# PROBABILISTIC GRAPHICAL MODELS FOR THE ANALYSIS AND SYNTHESIS OF MUSICAL AUDIO

MATTHEW DOUGLAS HOFFMAN

A DISSERTATION
PRESENTED TO THE FACULTY
OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE
BY THE DEPARTMENT OF
COMPUTER SCIENCE
ADVISER: PERRY RAYMOND COOK

NOVEMBER 2010

# Abstract

Content-based Music Information Retrieval (MIR) systems seek to automatically extract meaningful information from musical audio signals. This thesis applies new and existing generative probabilistic models to several content-based MIR tasks: timbral similarity estimation, semantic annotation and retrieval, and latent source discovery and separation.

In order to estimate how similar two songs sound to one another, we employ a Hierarchical Dirichlet Process (HDP) mixture model to discover a shared representation of the distribution of timbres in each song. Comparing songs under this shared representation yields better query-by-example retrieval quality and scalability than previous approaches.

To predict what tags are likely to apply to a song (e.g., "rap," "happy," or "driving music"), we develop the Codeword Bernoulli Average (CBA) model, a simple and fast mixture-of-experts model. Despite its simplicity, CBA performs at least as well as state-of-the-art approaches at automatically annotating songs and finding to what songs in a database a given tag most applies.

Finally, we address the problem of latent source discovery and separation by developing two Bayesian nonparametric models, the Shift-Invariant HDP and Gamma Process NMF. These models allow us to discover what sounds (e.g. bass drums, guitar chords, etc.) are present in a song or set of songs and to isolate or suppress individual source. These models' ability to decide how many latent sources are necessary to model the data is particularly valuable in this application, since it is impossible to guess a priori how many sounds will appear in a given song or set of songs.

Once they have been fit to data, probabilistic models can also be used to drive the synthesis of new musical audio, both for creative purposes and to qualitatively diagnose what information a model does and does not capture. We also adapt the SIHDP model to create new versions of input audio with arbitrary sample sets, for example, to create a sound file that matches a song as closely as possible by combining spoken text.

# Acknowledgements

I want to thank, first of all, my advisor Perry Cook and my co-advisor David Blei. Without their time, support, advice, instruction, and insight this dissertation could never have existed, and I would never have developed the skills I needed to write it. I also thank them for giving me the freedom to work on whatever projects I found exciting, and for their enthusiasm for those projects. I owe them both a tremendous debt of thanks.

I also want to thank the other members of my thesis committee, Ken Steiglitz, Adam Finkelstein, and Rob Schapire, for their helpful comments and suggestions.

Thanks also to Ge Wang, Ananya Misra, Jeff Bernstein, Rebecca Fiebrink, Jordan Boyd-Graber, Jonathan Chang, Sean Gerrish, Sam Gershman, Lauren Hannah, John Paisley, Gungor Polatkan, Chong Wang, Sonya Nikolova, Xiaojuan Ma, Zhe Wang, Kai Li, and the many other members of the Computer Science department who have provided me with support, friendship, and help at various times in various ways.

Without the inspiration of the faculty, staff, and graduate students at Columbia's Computer Music Center I might never have applied to graduate school in the first place. In particular, I want to thank Douglas Repetto, Brad Garton, Terry Pender, and Luke DuBois for getting me interested in the world of computational music research.

This work was supported by National Science Foundation grants 0101247, 0509447, 0745520, and 9984087, National Institutes of Health grant 5R44HL072534-03, Office of Naval Research grant 175-6343, New Jersey Center for Science and Technology grant 01-2042-007-22, the Kimberly and Frank H. Moss '71 Research Innovation Fund (Princeton School of Engineering and Applied Sciences), the E. L. Keyes, Jr. Emerson Electric Co. Faculty Award, Google Inc., Microsoft Corp., Interval Research, the Arial Foundation, and Lingraphicare. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of these institutions.

Finally, many thanks to my family: to my parents for their limitless love, support, wisdom, and good humor; to my grandparents and extended family for their kind wishes and confidence in me; and to my wife Maggie, for everything.

To my wife Maggie, who has made my time as a graduate student the happiest of my life.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

The past fifteen years have witnessed an explosion in the availability and prevalence of digital music stored on and accessed via computers. Many individuals have accumulated large libraries of songs, and companies such as Apple, Pandora, Amazon, Google, Spotify, Rhapsody, Napster, and Mog (to name only a few) manage libraries of many millions of recordings. The prevalence and size of digital music collections exposes new opportunities and challenges in organizing these collections. In some cases, these can be addressed without recourse to the music audio signals being organized. In others, content-based methods that analyze musical audio signals are necessary.

For example, digital jukebox applications such as iTunes allow users to search and organize their collections using metadata tags (e.g. artist name, genre, year of release, etc.). But such tags may be missing or inadequate, especially for so-called "long-tail" songs that have not achieved widespread popularity, and may not have received as much curatorial attention as more mainstream recordings have. An unsigned band that posts its home-recorded mp3s to its MySpace page (or a fan who tapes a Phish concert and uploads the result to the Internet Archive[1]) may have forgotten to fill in some of the available ID3 tags, for example.

Likewise, a company may want to recommend new songs to its users based on songs that the user has purchased or listened to in the past. Collaborative filtering methods like those used in the Netflix Prize competition [12] offer a potential solution to this problem, but cannot be used to recommend songs for which the company lacks sufficient user preference data. This again makes them unable to recommend long-tail songs (which almost by definition cannot have generated much user preference data), and limits their usefulness for smaller companies that have not yet have accumulated large databases of user preference data [22].

Such limitations of non-content-based approaches motivate the development of content-based Music Information Retrieval (MIR) methods that can extract and manipulate meaningful information from musical audio signals. This thesis presents several new techniques for analyzing musical audio signals, as well as ways of applying such techniques to the

[1] `http://www.archive.org`

problem of generating or transforming musical audio for creative purposes. A common thread through all of the techniques we present is the use of probabilistic graphical models, and particularly Bayesian nonparametric models, to analyze audio signals.

## 1.2 Probabilistic Graphical Modeling

A cursory examination of the proceedings of ISMIR, the International Conference on Music Information Retrieval [1], will show that machine learning plays a pivotal role in MIR. As in other pattern recognition domains [14], data-driven approaches have generally proven more effective than pure engineering approaches [2]. There has been, however, a (by no means universal) tendency within the MIR community to rely on well established machine learning techniques such as Gaussian Mixture Models (GMMs), Hidden Markov Models (HMMs), Support Vector Machines (SVMs), and boosting, using these algorithms as "black boxes" within larger systems rather than developing problem-specific machine learning algorithms.

Although well engineered feature extraction algorithms in conjunction with off-the-shelf learning methods can produce good results for many problems, we believe that better performance can be achieved by using learning methods that take into account the special structure of a problem. The framework of probabilistic graphical modeling [52] is well suited to this approach. The process of using probabilistic modeling to address a data analysis problem can be roughly broken into three steps:

1. Posit a probabilistic model of the data to be explained. This involves defining a family of joint probability distributions over all observable data and any hidden variables. This family will typically be indexed by a set of parameters—in Bayesian analyses these parameters are endowed with their own prior probability distributions indexed by a set of hyperparameters.

2. Derive an algorithm to infer settings of the parameters and/or hidden variables of interest that are consistent with the observable data. In fully Bayesian inference, this consists of reasoning about the posterior distribution over the parameters and hidden variables conditioned on the data and any hyperparameters. Maximum-likelihood inference algorithms instead try to find a single setting of the model parameters that maximizes the likelihood of the observed data under the model.

3. Use the inferences about the model parameters and hidden variables to address the problem under consideration.

In practice, this often becomes an iterative process where the results of an analysis may suggest alterations to the model or inference algorithm, difficulties in deriving an inference algorithm may motivate changes to the model, etc.

For example, say we want to partition some multidimensional real-valued data into $K$ clusters. We might

[2] Some notable exceptions do exist, for example the doctoral work of Anssi Klapuri [56]

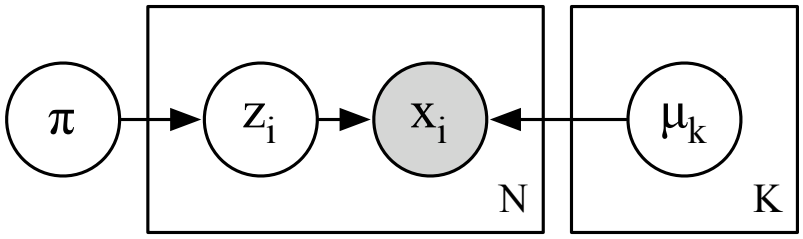


Figure 1.1: Graphical model representation of a mixture-of-Gaussians model.

1. Assume that the data came from a mixture of Gaussians model, where each observed vector of reals $\boldsymbol{x}_i$ is generated by

   (a) choosing a cluster index $z_i \in \{1, \ldots, K\}$ according to a multinomial distribution with weights $\boldsymbol{\pi}$, and

   (b) sampling the observed vector $\boldsymbol{x}_i$ from a Gaussian with mean $\boldsymbol{\mu}_{z_i}$ and diagonal covariance matrix $\sigma^2 I$.

   Here the model parameters to be inferred are the Gaussian means $\boldsymbol{\mu}$ and the mixture weights $\boldsymbol{\pi}$, and the hidden variables that need to be inferred are the cluster indices $\boldsymbol{z}$.

2. Use the Expectation-Maximization (EM) algorithm [25] to find a setting of $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$ that is a (local) maximum of the likelihood $p(\boldsymbol{x}|\boldsymbol{\pi}, \boldsymbol{\mu})$ of the observations $\boldsymbol{x}$, and infer the posterior $p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\pi}, \boldsymbol{\mu})$ over the cluster indices $\boldsymbol{z}$.

3. Assign each observation $\boldsymbol{x}_i$ to the cluster $k$ for which the posterior probability $p(z_i = k|\boldsymbol{x}_i, \boldsymbol{\pi}, \boldsymbol{\mu})$ is highest (or, if fractional cluster membership is permissable, use the posterior over the possible values of $z_i$ directly).

The term graphical modeling refers to the practice of representing the dependency structure of a model as a graph whose nodes denote random variables and whose edges denote dependencies between random variables. Shaded nodes denote observable variables, unshaded nodes denote hidden variables whose values must be inferred (or integrated out). A directed edge from a variable $x$ to a variable $y$ means that the joint probability $p(x, y)$ factorizes to $p(x)p(y|x)$. Undirected edges (which do not appear in the models described in this thesis) denote joint dependencies that cannot be factorized in this way. Replication of random variables is denoted by "plates," boxes that surround a group of random variables. The number in the corner of the plate denotes how many times the set of random variables is replicated. Figure 1.1 shows the graphical model for the mixture of Gaussians model described above with $N$ observations and $K$ mixture components.

The probabilistic graphical modeling framework has several advantages to recommend it:

**It is extremely flexible.** Almost any learning problem can be framed in terms of parameter inference. Probabilistic models give state-of-the-art performance in problems as diverse as regression, classification, clustering, factor analysis, collaborative filtering, etc.

**It provides a language for formally specifying the assumptions made by an algorithm, and for relaxing those assumptions as necessary.** For example, the clustering model discussed above assumes that all clusters should have the same size and shape, since each Gaussian has a fixed diagonal covariance matrix. Should this assumption prove problematic, it can be relaxed by changing the model to include a separate covariance matrix for each Gaussian.

**It provides a principled alternative to ad hoc algorithm design.** The core computational challenge in probabilistic modeling is the well defined (and well studied) problem of probabilistic inference, for which a large arsenal of generic techniques exists. If a model is correctly specified, and an appropriate inference algorithm can be applied, then the problem can be solved.

**It is modular and extensible.** Probabilistic graphical modeling provides a principled paradigm for extending and combining simple models to address the unique structure of a particular problem.

Our goal throughout this thesis will be to address problems in the analysis and synthesis of musical audio by building and fitting problem-appropriate probabilistic graphical models to audio data. We will generally avoid making strong assumptions about what or how many kinds of sounds are present in our data, what insights from music theory are applicable, etc. The motivation for this is twofold. First, models that make fewer assumptions are often easier to build, apply, and understand. Second, a priori assumptions are often wrong, and by relying on data rather than prior intuitions we can build models that are applicable to a wider range of datasets.

## 1.3 Contributions

Below we outline the contributions of this thesis. Since a number of different problems are addressed by these contributions, each of which has its own literature, more thorough reviews of prior work are postponed until the chapters devoted to each application.

### 1.3.1 Analysis Applications

This thesis presents new techniques to address several problems in content-based analysis of musical audio. These problems are summarized below.

**Timbral similarity.** The ability to automatically assess the similarity of one song to another has several applications in MIR. One basic MIR paradigm is query-by-example, in which a user has a query song and wants to find other songs that are similar to that song—clearly the ability of a query-by-example system to evaluate the similarity of two songs is critical to its success [71]. Audio similarity estimation also has applications to playlist

generation, where it can be used to ensure that a sequence of songs lacks jarring transitions [72]. Finally, one approach to the problem of music recommendation is to suggest songs to users that resemble songs they already like. Previous approaches to analyzing timbral similarity have either been overly restrictive or too expensive to scale to large music collections [71]. Furthermore, the most successful previous approaches suffered from the problem of "hubs," songs that are erroneously evaluted as very similar to all other songs in the corpus [8]. In chapter 3, we will present a method based on an expressive model-based representation that outperforms previous approaches in a query-by-example task, is more computationally efficient than previous methods, and does not suffer from the problem of hubs.

**Automatic tagging.** One way of organizing songs is by tag-based annotation, in which a song is labeled with a set of words that apply to that song. These tags can be used to sort songs [28], to facilitate semantic retrieval (in which a user types in a set of words that describe the kind of song he or she is looking for) [89], or as a high-level feature representation for other MIR tasks [10]. In chapter 4, we will present a simple, efficient model that predicts what tags will apply to a song based on its audio content. Despite its simplicity, this model nonetheless achieves results competitive with more complex and expensive approaches.

**Latent source discovery and separation.** Most music recordings feature two or more sound sources playing simultaneously. The challenge of Blind Monaural Source Separation (BMSS) is to separate such mixed recordings into their component source signals. Human auditory systems can do this easily in many cases [3], but BMSS remains a difficult problem for computers. BMSS has applications to automatic music transcription, mid-level music feature extraction [23], music production [39] and upmixing monaural or stereophonic recordings to "surround sound" multichannel audio formats [32]. In chapters 5 and 6, we will present two models that are able to learn what sound sources are present in a musical recording or set of musical recordings, when and how strongly they appear, and how many such latent sources are needed to explain the variation in the audio signal. The ability of these models to decide how many latent sources to posit is a significant advance over previous methods, which typically assume that the number of latent sources needed to explain the data is somehow known a priori.

### 1.3.2 Synthesis Applications

In addition to the analysis problems summarized above, we also present applications of probabilistic modeling to musical audio synthesis.

[3] Or so human beings intuitively assume. It may be that the human auditory system merely gives the *perception* of solving the problem of separation when it is in fact solving the problem of pattern recognition in the presence of confounding signals. The second problem may have been both easier to solve and of greater adaptive value to our ancestors.

**Model-based feature synthesis.** An interesting property of generative probabilistic models is their ability to generate new data once fit to a set of training data. In chapter 7, we explore the use of Hidden Markov Models (HMMs) to synthesize audio inspired by a training song or set of training songs. In order to turn the sequences of feature vectors generated by a model into audio, we use the paradigm of feature-based synthesis.

**Bayesian spectral matching.** The latent source model developed in chapter 5 learns both what sources are present in a recording and how to combine those sources to approximate that recording. By fixing a set of sources ahead of time, however, we can use a simplified version of the same modeling machinery to learn how to combine those sources to approximate arbitrary recordings. The result is a system that can automatically arrange arbitrary samples to approximate arbitrary audio signals. This can also be thought of as a form of cross-synthesis, where a sampler is automatically manipulated to approximate the spectral characteristics of a target sound. We describe this approach and some of its creative possibilities in chapter 8.

## 1.4 Related Publications

Most of the material in this dissertation is based on articles published in various conference proceedings. Chapters 3, 4, 5, and 6 are based on papers presented at ISMIR 2008 [44], ISMIR 2009 [47], DAFX 2009 [48], and ICML 2010 [49], respectively. Chapter 7 is based on papers presented at ICMC 2006 [46] and ICMC 2008 [45], and chapter 8 is based on a paper presented at ICMC 2009 [50].

# Chapter 2

# Background

In this chapter, we discuss some background knowledge and previous work helpful to understanding the rest of the thesis. We begin by reviewing some standard approaches to probabilistic inference, and then discuss some standard models used in the Bayesian nonparametrics literature.

## 2.1 Inference Techniques

As mentioned above, statistical inference is the central computational challenge in data analysis using probabilistic modeling. Say we have a model with some unknown parameters $\boldsymbol{\theta}$, some hidden variables $\boldsymbol{z}$, and some observed data $\boldsymbol{x}$ [1]. In Bayesian inference, the goal is to reason about the posterior distribution over the parameters and hidden variables conditioned on the data, which is given by Bayes' rule:

$$p(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{z})p(\boldsymbol{\theta}, \boldsymbol{z})}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{z})p(\boldsymbol{\theta}, \boldsymbol{z})}{\int_{\boldsymbol{\theta}} \int_{\boldsymbol{z}} p(\boldsymbol{x}|\boldsymbol{\theta}, \boldsymbol{z})p(\boldsymbol{\theta}, \boldsymbol{z}) d\boldsymbol{\theta} d\boldsymbol{z}}. \tag{2.1}$$

For computational (or philosophical) reasons, one may prefer to fit a simple point estimate of our model parameters according to the principle of maximum likelihood (ML) or maximum a posteriori likelihood (MAP):

$$\boldsymbol{\theta}^{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{x}|\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}) d\boldsymbol{z}; \tag{2.2}$$

$$\boldsymbol{\theta}^{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\boldsymbol{x}) = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{x}, \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \int_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta}) d\boldsymbol{z}. \tag{2.3}$$

Except in very simple models, closed-form solutions to inference problems are rarely available. Fully Bayesian inference is particularly challenging in that computing the normalizing constant $p(\boldsymbol{x})$ requires computing the integral in the denominator of equation 2.1, which is often intractable. Approximate numerical inference algorithms can nonetheless

[1] Note that the distinction between parameters and hidden variables is somewhat arbitrary, particularly in Bayesian inference. We will usually refer to an unobserved variable as a parameter if its dimensionality does not increase with the number of observations, and a hidden variable otherwise.

produce results that are sufficient for the problem at hand. In this thesis we will use two main families of inference strategies: deterministic methods using the framework of variational inference, and Markov Chain Monte Carlo (MCMC) methods that produce samples from a posterior distribution. A brief overview of these approaches follows—more complete introductions and derivations can be found in [94] (for variational inference) and [67] (for MCMC).

### 2.1.1 Variational Inference

Variational inference [53] is a deterministic approach to inference that frames the inference problem as one of maximizing a lower bound. It includes the classic Expectation-Maximization (EM) algorithm [25] as a special case [69]. We will begin by deriving the EM algorithm in the variational framework, then show how variational methods can be applied to approximate Bayesian inference.

When doing maximum-likelihood estimation of a set of parameters $\boldsymbol{\theta}$ for a probabilistic model with hidden variables $\boldsymbol{z}$ and observed data $\boldsymbol{x}$, it is often easier to optimize the so-called complete log-likelihood $\log p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})$ than to optimize the marginal likelihood $\log p(\boldsymbol{x}|\boldsymbol{\theta}) = \log \int_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}) d\boldsymbol{z}$ due to the intractability of integrating over all possible settings of the hidden data. By Jensen's inequality, which states that any concave function (such as the logarithm) of an expectation is greater than or equal to the expectation of that function, we can write

$$\begin{aligned}
\log p(\boldsymbol{x}|\boldsymbol{\theta}) &= \log \int_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}) d\boldsymbol{z} \\
&= \log \int_{\boldsymbol{z}} \frac{q(\boldsymbol{z})}{q(\boldsymbol{z})} p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}) d\boldsymbol{z} \\
&= \log \mathbb{E}_q \left[ \frac{p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})}{q(\boldsymbol{z})} \right] \\
&\geq \mathbb{E}_q \left[ \log \frac{p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})}{q(\boldsymbol{z})} \right] \\
&= \mathbb{E}_q[\log p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})] - \mathbb{E}_q[\log q(\boldsymbol{z})],
\end{aligned} \tag{2.4}$$

which holds for any distribution $q(\boldsymbol{z})$. An application of the chain rule $p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta}) = p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{x}|\boldsymbol{\theta})$ and little more algebraic manipulation show that the slack in the bound is given by

$$\begin{aligned}
&\log p(\boldsymbol{x}|\boldsymbol{\theta}) - \mathbb{E}_q[\log p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})] + \mathbb{E}_q[\log q(\boldsymbol{z})] \\
&= \log p(\boldsymbol{x}|\boldsymbol{\theta}) - \mathbb{E}_q[\log p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta})] - \mathbb{E}_q[\log p(\boldsymbol{x}|\boldsymbol{\theta})] + \mathbb{E}_q[\log q(\boldsymbol{z})] \\
&= \mathbb{E}_q \left[ \log \frac{q(z)}{p(z|\boldsymbol{x}, \boldsymbol{\theta})} \right],
\end{aligned} \tag{2.5}$$

which is the Kullback-Leibler divergence (KLD) between $q(\boldsymbol{z})$ and the posterior distribution $p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta})$. Since this KLD can be driven to zero by setting $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta})$, the bound can be tightened perfectly as long as $p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta})$ has an analytic form (as it does for

many models). This is the "E" step in the EM algorithm. Holding $q$ fixed, $\boldsymbol{\theta}$ can then be set to maximize $\mathbb{E}_q[\log p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})]$; doing so constitutes the "M" step in the EM algorithm. Iterating between E and M steps will monotonically increase the marginal log-probability $\log p(\boldsymbol{x}|\boldsymbol{\theta})$, ultimately reaching a local maximum.

The same basic approach can likewise be applied to MAP estimation. The only difference is the addition of a prior term $\log p(\boldsymbol{\theta})$:

$$\log p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \geq \mathbb{E}_q[\log p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})] + \log p(\boldsymbol{\theta}) - \mathbb{E}_q[\log q(\boldsymbol{z})]. \tag{2.6}$$

This additional term must be accounted for during the M step.

Finally, variational inference can also be used to develop approximate Bayesian inference algorithms. The core idea is to replace the true posterior $p(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{x})$ with an approximate posterior $q(\boldsymbol{\theta}, \boldsymbol{z})$, and optimize $q(\boldsymbol{\theta}, \boldsymbol{z})$ to be as close as possible (in KL divergence) to $p(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{x})$. This can be done by lower bounding the (often uncomputable) marginal probability $\log p(\boldsymbol{x})$:

$$\begin{aligned}
\log p(\boldsymbol{x}) &= \log \int_{\boldsymbol{z},\boldsymbol{\theta}} p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta}) d\boldsymbol{z} d\boldsymbol{\theta} \\
&= \log \int_{\boldsymbol{z},\boldsymbol{\theta}} \frac{q(\boldsymbol{z}, \boldsymbol{\theta})}{q(\boldsymbol{z}, \boldsymbol{\theta})} p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta}) d\boldsymbol{z} d\boldsymbol{\theta} \\
&= \log \mathbb{E}_q \left[ \frac{p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta})}{q(\boldsymbol{z}, \boldsymbol{\theta})} \right] \\
&\geq \mathbb{E}_q \left[ \log \frac{p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta})}{q(\boldsymbol{z}, \boldsymbol{\theta})} \right] \\
&= \mathbb{E}_q[\log p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta})] - \mathbb{E}_q[\log q(\boldsymbol{z}, \boldsymbol{\theta})].
\end{aligned} \tag{2.7}$$

Essentially, we are treating the parameters $\boldsymbol{\theta}$ as being no different from any other hidden variable and proceeding as above. Optimizing this lower bound is equivalent to minimizing the KL divergence between $q(\boldsymbol{z}, \boldsymbol{\theta})$ and the posterior of interest, $p(\boldsymbol{z}, \boldsymbol{\theta}|\boldsymbol{x})$.

So far, we have said little about the form $q$ should take. For simple models, it may be possible to set $q$ equal to the posterior analytically. For hierarchical Bayesian models, however, this is not usually possible. The simplest, and most common, approach is to instead give $q$ a fully factorized form—that is, to make all variables independent under $q$. This approach, called mean-field [2] variational inference, generally makes it impossible to perfectly match $q$ to the true posterior, but can also dramatically simplify optimization. For example, if we endowed the mixture-of-Gaussians model described above with priors on the parameters $\boldsymbol{\pi}$ and $\boldsymbol{\mu}$, then the mean-field variational bound for that model would factorize to

$$\begin{aligned}
\log p(\boldsymbol{x}) \geq \sum_i \mathbb{E}_q[\log p(\boldsymbol{x}_i|z_i, \boldsymbol{\mu})] + \mathbb{E}_q[\log p(z_i|\boldsymbol{\pi})] \\
+ \mathbb{E}_q[\log p(\boldsymbol{\pi})] - \mathbb{E}_q[\log q(\boldsymbol{\pi})] + \sum_k \mathbb{E}_q[\log p(\boldsymbol{\mu}_k)] - \mathbb{E}_q[\log q(\boldsymbol{\mu}_k)].
\end{aligned} \tag{2.8}$$

[2] The term "mean-field" is a reference to the mean field approximation of statistical mechanics.

This bound can then be optimized by a coordinate ascent algorithm directly analogous to EM, iteratively setting $\log q(\boldsymbol{z}) = \mathbb{E}_q[\log p(\boldsymbol{z}|\boldsymbol{\pi}, \boldsymbol{x})]$, $\log q(\boldsymbol{\pi}) = \mathbb{E}_q[\log p(\boldsymbol{\pi}|\boldsymbol{z})]$, and $\log q(\boldsymbol{\mu}) = \mathbb{E}_q[\log p(\boldsymbol{\mu}|\boldsymbol{z}, \boldsymbol{x})]$. If the priors on $\boldsymbol{\pi}$ and $\boldsymbol{\mu}$ are conjugate to the multinomial and normal distributions respectively, i.e. if the conditional probabilities $p(\boldsymbol{\mu}|\boldsymbol{z}, \boldsymbol{x})$ and $p(\boldsymbol{\pi}|\boldsymbol{z})$ have the same form as the priors $p(\boldsymbol{\mu})$ and $p(\boldsymbol{\pi})$, then these updates can be done analytically.

### 2.1.2 Markov Chain Monte Carlo Inference

Often in Bayesian models, the joint distribution $p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta})$ over parameters $\boldsymbol{\theta}$, hidden data $\boldsymbol{z}$, and observable data $\boldsymbol{\theta}$ has a convenient form but the marginal likelihood of the data $p(\boldsymbol{x}) = \int_{\boldsymbol{z},\boldsymbol{\theta}} p(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta}) d\boldsymbol{z} d\boldsymbol{\theta}$ is intractable to compute. $p(\boldsymbol{x})$ is needed to normalize the posterior probability $p(\boldsymbol{z}, \boldsymbol{\theta}|\boldsymbol{x})$, but is constant with respect to the parameters and hidden data. When this is the case, Markov Chain Monte Carlo (MCMC) methods can be used to sample from a Markov chain whose stationary distribution is the true posterior [67]. A set of samples from this Markov chain can then be used to approximate the posterior distribution of interest.

MCMC, like variational inference, has its roots in the statistical physics literature, beginning with the classic paper of Metropolis et al. [64]. One of the simplest MCMC algorithms used for Bayesian inference is Gibbs sampling [37], which proceeds by repeatedly resampling the value of each parameter conditioned on each other parameter. In the example of the mixture-of-Gaussians model above, one might alternate between sampling $\boldsymbol{\pi}$ from $p(\boldsymbol{\pi}|\boldsymbol{z})$, sampling each $\boldsymbol{\mu}_k$ from $p(\boldsymbol{\mu}_k|\boldsymbol{x}, \boldsymbol{z})$, and sampling each $z_i$ from $p(z_i|\boldsymbol{x}_i, \boldsymbol{\mu})$. As in mean-field variational inference, these conditional distributions have an analytic form if conjugate priors are chosen for $p(\boldsymbol{\pi})$ and $p(\boldsymbol{\mu})$. After a "burn-in" period during which the Markov chain converges to a region of high probability mass, samples from the Markov chain will almost surely have the same distribution as the (intractable) posterior of interest. In practice, the output of the Markov chain should be subsampled (e.g. keeping only every 10th sample) to ensure that the samples that are used to approximate the posterior are uncorrelated.

Although there are strong asymptotic guarantees that a properly designed MCMC sampler will have the same stationary distribution as the posterior of interest, in practice Gibbs samplers that are run for a finite amount of time usually only find a single mode of the posterior and explore the space around it. This is due to the extremely low probability of going from a region of high probability mass to a region of low probability mass on the way to another distant region of high probability mass. The problem is analogous to the problem of getting stuck in local optima when optimizing non-convex objective functions (such as variational bounds). Nonetheless, these local modes are often good enough to solve practical problems.

## 2.2 Bayesian Nonparametric Mixture Modeling

Many models have some order parameter $K$ that must be set a priori. For example, Gaussian mixture models assume that each element of a set of observations was drawn from one

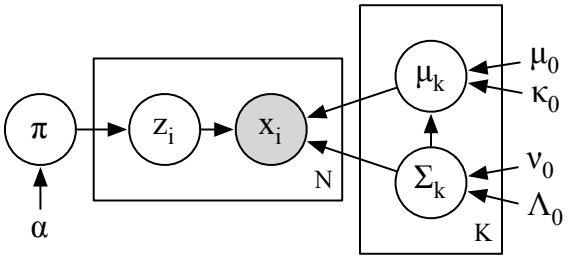


Figure 2.1: Graphical model for a Bayesian GMM with a full set of covariance matrices $\boldsymbol{\Sigma}$.

of $K$ Gaussian mixture components. Bayesian nonparametric models sidestep the problem of how to set $K$, typically by assuming the existence of an *infinite* number of latent variables, while placing a prior on the model that only expects a finite number of these latent variables to have any association with the observed data.

Several of the applications in this dissertation make use of the Hierarchical Dirichlet Process (HDP) [85], a particularly powerful Bayesian nonparametric model that extends the Dirichlet Process Mixture Model (DPMM) [6], a standby of Bayesian nonparametric modeling. In this section, we will review traditional mixture modeling from a Bayesian perspective, show how the DP can be used to extend Bayesian mixture models to the setting where the number of mixture components is unknown a priori, and then describe how the HDP extends the DP to model grouped data.

### 2.2.1 Bayesian Mixture Models

Bayesian mixture models often assume a generative process of the form

$$\boldsymbol{\phi}_k \sim H; \quad \boldsymbol{\pi} \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_K); \quad z_i \sim \text{Multinomial}(\boldsymbol{\pi}); \quad \boldsymbol{x}_i \sim f(\boldsymbol{\phi}_{z_i}), \quad (2.9)$$

where $H$ defines a prior distribution over a set of $K$ mixture component parameters $\boldsymbol{\phi}$, $f(\boldsymbol{\phi}_k)$ is the probability density or mass function associated with component $k$, $\boldsymbol{\pi}$ is a vector of nonnegative mixture weights summing to 1, $z_i \in \{1, \ldots, K\}$ is an indicator saying what mixture component is responsible for the $i$th observation, and $\boldsymbol{x}_i$ is the $i$th observation. For a Gaussian Mixture Model (GMM) with full covariance matrices $\boldsymbol{\Sigma}_k$, $\boldsymbol{\phi}_k = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ and $f(\boldsymbol{x}_i; \boldsymbol{\phi}_{z_i})$ would be the normal Probability Density Function (PDF) $\mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_{z_i}, \boldsymbol{\Sigma}_{z_i})$. A natural choice for $H$ in this case would be the normal-inverse-Wishart distribution, which is conjugate to the multivariate normal distribution with unknown mean and covariance [36] and can be written as a two-stage generative process:

$$\boldsymbol{\Sigma} \sim \text{Inverse-Wishart}_{\nu_0}(\boldsymbol{\Lambda}_0^{-1}); \quad \boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}/\kappa_0), \quad (2.10)$$

where $\nu_0$, $\kappa_0$, $\boldsymbol{\mu}_0$, and $\boldsymbol{\Lambda}_0$ are parameters to the normal-inverse-Wishart distribution. The graphical model for the GMM is shown in figure 2.1. It differs from the GMM discussed in the introduction in that the Gaussians are given full covariance matrices, and all parameters are endowed with priors controlled by hyperparameters such as $\boldsymbol{\alpha}$.

The Dirichlet distribution, which has PDF

$$\text{Dirichlet}(\boldsymbol{\pi}; \boldsymbol{\alpha}) = \frac{\Gamma(\sum_k \alpha_k)}{\sum_k \Gamma(\alpha_k)} \exp\left\{ \sum_k (\alpha_k - 1) \log \pi_k \right\}, \tag{2.11}$$

is the conjugate prior for the multinomial distribution, which means that given the prior parameter $\boldsymbol{\alpha}$ and a set of observed values for some indicator variables $\boldsymbol{z}$, the posterior $p(\boldsymbol{\pi}|\boldsymbol{\alpha}, \boldsymbol{z})$ is also a Dirichlet distribution. Conjugate priors are defined by this property in general—if the parameter to a distribution is given a conjugate prior, then the posterior distribution over that parameter conditioned on observed data will be in the same family as the prior. Using conjugate priors dramatically simplifies Bayesian inference, since the posterior distribution of each parameter conditioned on all other variables in the model is guaranteed to have an analytic form if that parameter is endowed with a conjugate prior.

If we denote the sum of the elements of $\boldsymbol{\alpha}$ as $\bar{\alpha} = \sum_k \alpha_k$, we can think of $\bar{\alpha}$ as controlling the sparsity of the Dirichlet prior on $\boldsymbol{\pi}$. Larger values of $\bar{\alpha}$ will give more weight to less sparse vectors, while smaller values of $\bar{\alpha}$ will give more weight to sparser vectors. As $\bar{\alpha} \to 0$ for a fixed dimensionality $K$, the probability of more than one element of $\boldsymbol{\pi}$ being significantly greater than 0 goes to 0, while as $\bar{\alpha} \to \infty$, $\pi_k = \alpha_k / \bar{\alpha}$ for each element $k$ of $\boldsymbol{\pi}$ with probability 1. In the next section, we will consider what happens when $K$ grows arbitrarily large while $\bar{\alpha}$ remains constant.

### 2.2.2 Dirichlet Process Mixture Models

As suggested above, a nagging issue in mixture modeling is model order selection, i.e., choosing the number of components $K$ with which to explain the data. Bayesian nonparametric statistics focuses on models such as the Dirichlet Process Mixture Model (DPMM) that sidestep this issue. Where a standard mixture model assumes the existence of $K$ mixture components, the DPMM [6, 31] assumes the existence of a countably infinite set of mixture components, only a finite subset of which are used to explain the observations.

There are a number of constructions of the DPMM. One is by considering what happens to the finite mixture model

$$\boldsymbol{\phi}_k \sim H; \quad \boldsymbol{\pi} \sim \text{Dirichlet}(\alpha/K, \ldots, \alpha/K); \quad z_t \sim \text{Multinomial}(\boldsymbol{\pi}); \quad \boldsymbol{y}_t \sim f(\boldsymbol{\phi}_k) \tag{2.12}$$

as $K \to \infty$. As the number of mixture components grows larger and larger, the Dirichlet prior on $\boldsymbol{\pi}$ grows sparser and sparser to compensate. This phenomenon is illustrated in figure 2.2 for $\alpha = 2$; as $K$ increases, the number of elements of $\boldsymbol{\pi}$ significantly greater than 0 remains within the same range. Even when $K$ goes to infinity, it turns out that for any $\epsilon \in (0, 1]$, there exists a finite set $S$ of indices $s \in \mathbb{N}$ such that $\sum_{s \in S} \pi_s > 1 - \epsilon$ with probability 1 [55]. That is, only a finite number of elements of $\boldsymbol{\pi}$ are needed to account for virtually all of the mass in $\boldsymbol{\pi}$.

Since the number of elements of $\boldsymbol{\pi}$ significantly greater than zero is finite with probability 1 in the prior, it must likewise be finite with probability one in the posterior after conditioning on a finite set of observations $\boldsymbol{y}$. Obviously, it is computationally impossible to explicitly represent a vector $\boldsymbol{\pi}$ of infinite dimensionality. It is, however, possible to inte-

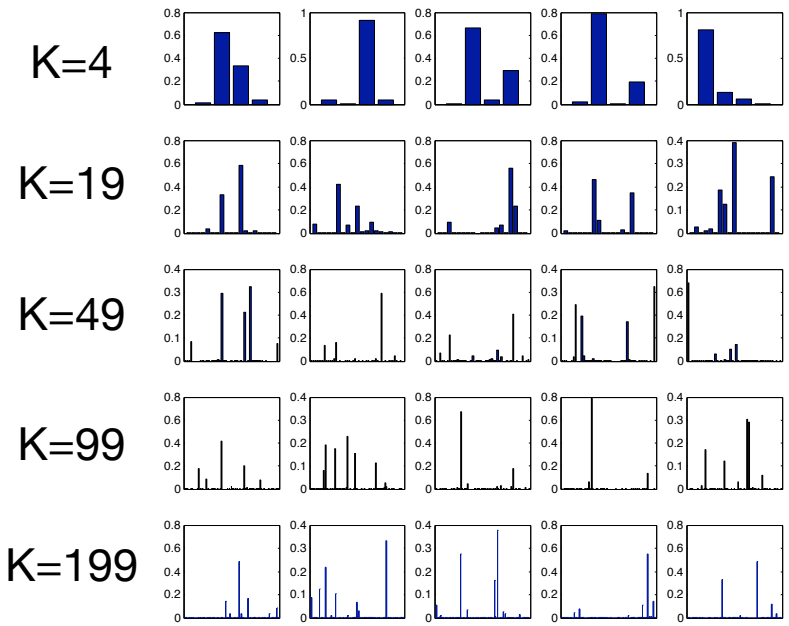


Figure 2.2: Five vectors drawn from a Dirichlet$(2/K, \ldots, 2/K)$ distribution for various dimensionalities $K$.

grate out $\boldsymbol{\pi}$ and perform posterior inference by representing explicitly only the indicators $\boldsymbol{z}$ and those emission parameters $\boldsymbol{\phi}$ associated with observed data via the indicators $\boldsymbol{z}$ using the Chinese Restaurant Process (CRP) representation of the DP [68].

In the CRP, we imagine a Chinese restaurant with an infinite number of communal tables (each with an infinite number of seats) and a positive scalar hyperparameter $\alpha$. The restaurant is initially empty. The first customer sits at the first table and orders a dish that will be shared by all future customers sitting at that table. The second customer enters and decides either to sit at the first table with probability $\frac{1}{1+\alpha}$ or at an empty table with probability $\frac{\alpha}{1+\alpha}$. When sitting at an empty table a customer orders a new dish, when joining an occupied table a customer eats the dish being eaten by the other customers at that table. This process continues for each new customer, with the $t$th customer choosing either to sit at a new table with probability $\frac{\alpha}{\alpha+t-1}$ or at the $k$th existing table with probability $\frac{n_k}{\alpha+t-1}$, where $n_k$ is the number of other customers already sitting at table $k$. Notice that popular tables become more popular, and that as more customers come in they become less and less likely to sit down at a new table.

We obtain a DPMM from a CRP as follows. The "dishes" in the CRP correspond to probability density functions, and the process of "ordering" a dish $k$ corresponds to drawing the parameters $\boldsymbol{\phi}_k$ to a PDF from a prior distribution $H$ over those parameters. (For example, each dish $\boldsymbol{\phi}_k$ can be a Gaussian with parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\} = \boldsymbol{\phi}_k \sim H$.) The process of a customer $t$ choosing a table $z_t$ corresponds to choosing a distribution $\boldsymbol{\phi}_{z_t}$ from which to draw an observation $\boldsymbol{y}_t$ (in our case, a feature vector). Since customers in

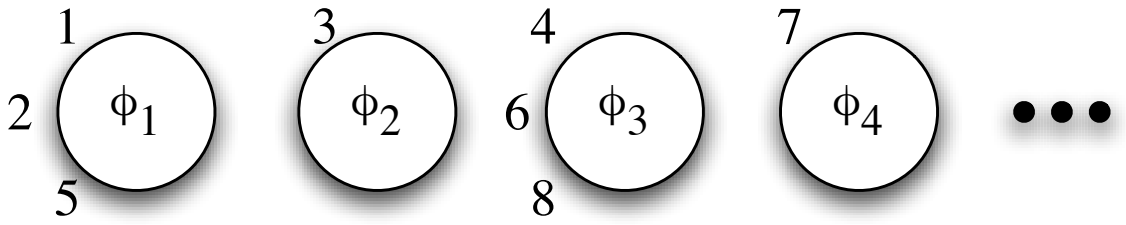


Figure 2.3: Four tables and eight customers in a Chinese Restaurant Process (CRP). In this example, the 1st, 3rd, 4th, and 7th customers all sat at an empty table, whereas the 2nd, 5th, 6th, and 8th sat at existing tables. The 9th customer will sit at table 1, 2, 3, or 4 with probabilities $\frac{3}{8+\alpha}$, $\frac{1}{8+\alpha}$, $\frac{3}{8+\alpha}$, and $\frac{1}{8+\alpha}$ respectively, or will sit at a new table with probability $\frac{\alpha}{8+\alpha}$

the CRP tend to sit at tables with many other customers, the DPMM tends to draw points from the same mixture components again and again even though it has an infinite number of mixture components to choose from. The process of drawing from a CRP is illustrated in figure 2.3.

Although we have described the CRP as a sequential process, in fact data under a CRP are exchangeable—the probability of a seating plan under the CRP is the same regardless of the order in which the customers sat down. This allows us to think of the CRP as defining an implicit prior on infinite multinomial distributions over mixture components. It turns out that a draw from a CRP is equivalent (modulo a permutation of the label identities) to a draw from the process

$$\boldsymbol{\pi} \sim \text{Dirichlet}(\alpha/K, \ldots, \alpha/K); \quad z_t \sim \text{Multinomial}(\boldsymbol{\pi}) \tag{2.13}$$

when $K \to \infty$.

Analysis under a DPMM involves inferring the posterior distribution over its latent parameters conditioned on the data. This provides both a partition of the data (feature vectors) into an unknown number of clusters (the number of tables) and the identities of the parameters $\boldsymbol{\phi}$ (the parameters to the mixture components). The posterior distribution $p(\boldsymbol{\phi}, \boldsymbol{z}|\boldsymbol{y}, \alpha, H)$ of the set of mixture component parameters $\boldsymbol{\phi}$ and the cluster labels $\boldsymbol{z}$ conditioned on the data $\boldsymbol{y}$ can be inferred using Markov Chain Monte Carlo (MCMC) methods such as Gibbs sampling [68]. For simple data, there will be relatively few unique cluster labels in $\boldsymbol{z}$, but more clusters will be necessary to explain more complex data.

### 2.2.3 The Hierarchical Dirichlet Process

The Hierarchical Dirichlet Process (HDP) [85] is a model of *grouped data*, which is more appropriate than the DPMM when observations are presented in nonhomogeneous groups. (For example, the HDP can be used to analyze text corpora where observed words are grouped into documents. In chapter 3 the groups will be sets of feature vectors grouped by the songs they were extracted from.) Rather than model each group using independent DPMMs (which would make it difficult to compare groups, and impossible to share statistical

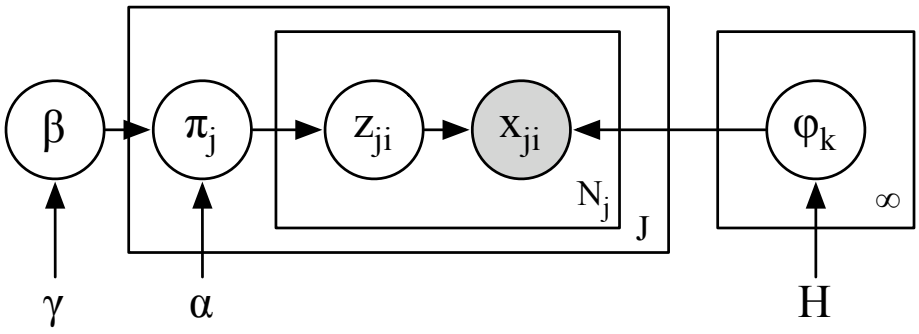


Figure 2.4: Graphical model for the HDP.

strength between groups) or model all groups using a single DPMM (which would restrict the model's ability to model differences between individual groups), the HDP lets us model each group of observations as being generated using a group-specific mixture of a globally shared set of mixture components.

The generative process underlying the HDP can be understood with the Chinese Restaurant Franchise (CRF) metaphor. The CRF takes two hyperparameters $\alpha$ and $\gamma$. Each group (e.g., a song) $j$ has its own CRP with hyperparameter $\alpha$, and each feature vector $\boldsymbol{y}_{jt}$ is assigned to a group-level table $z_{jt}$ according to that CRP. When a customer in a group-level CRP sits down at a new table, it chooses a dish for that table from a global CRP (with hyperparameter $\gamma$) shared by all groups—that is, it either chooses a dish that is already being served at some number $m$ of other tables with probability proportional to $m$, or it chooses a new dish with probability proportional to $\gamma$. The process of drawing from a CRF is illustrated in figure 2.5.

Just as the DPMM can be thought of as the infinite limit of a finite Bayesian mixture model, the HDP can also be thought of as the infinite limit of a finite mixed-membership mixture model given by:

$$\begin{aligned} \boldsymbol{\phi}_k \sim H; \quad \boldsymbol{\beta} \sim \text{Dirichlet}(\gamma/K, \ldots, \gamma/K); \\ \boldsymbol{\pi}_j \sim \text{Dirichlet}(\alpha\boldsymbol{\beta}); \quad z_{ji} \sim \text{Multinomial}(\boldsymbol{\pi}_j); \quad \boldsymbol{y}_{ji} \sim f(\boldsymbol{\phi}_{z_{ji}}) \end{aligned} \tag{2.14}$$

when $K \to \infty$. This representation complements the CRF representation. A graphical model for the HDP is given in figure 2.4. Note that the only group-specific parameters that are learned in the HDP are the mixture weight vectors $\boldsymbol{\pi}$. These weight vectors give a compact summary of the observed data for each group.

For a more complete exposition of the HDP, including details of how to infer the posteriors for its parameters conditioned on data, see [85].

### 2.2.4 Stick-Breaking construction

Another construction of the DP exists, and is worth mentioning here mostly for notational convenience. The stick-breaking construction of the DP [77] gives an constructive definition of the infinite Dirichlet prior defined implicitly above as the limit of

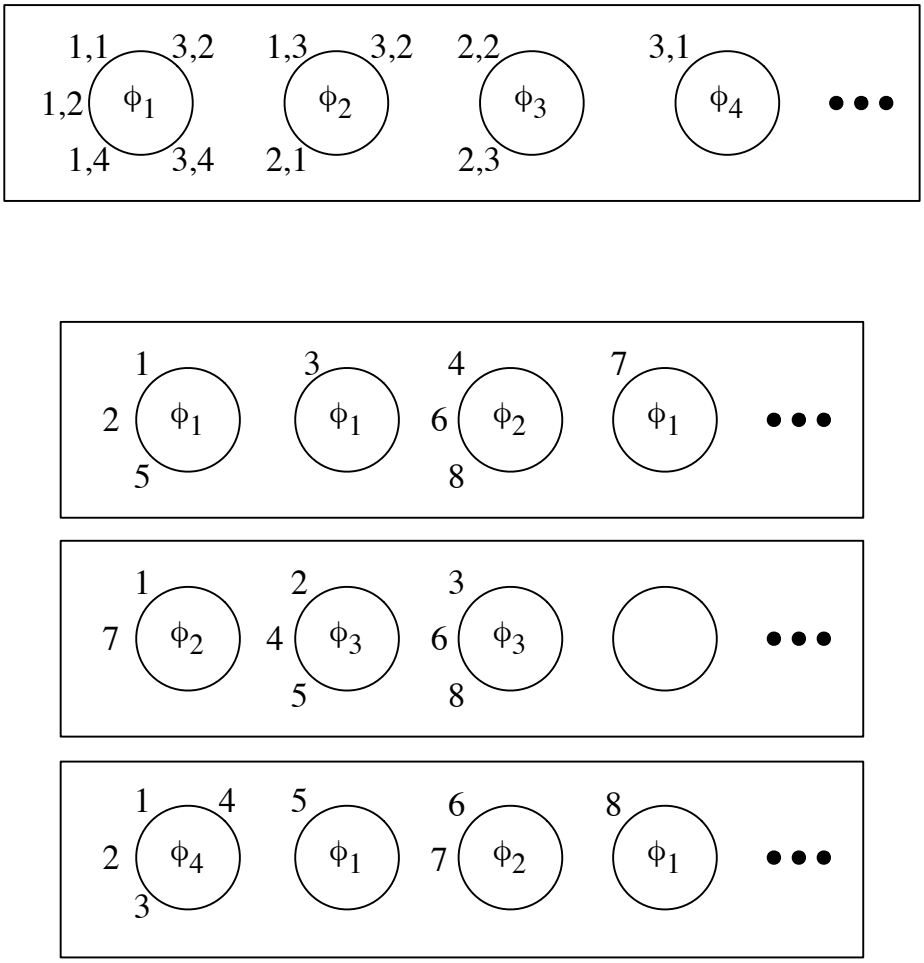


Figure 2.5: Chinese Restaurant Franchise (CRF) for three groups of eight observations each. Below are three CRPs (corresponding to the three groups), and above is the global CRP from which the CRPs get their dishes. Each customer $j, i$ sitting at a table in the global CRP corresponds to table $i$ in restaurant $j$, and customer $j, i$'s table membership in the global CRP determines the dish that is served at table $i$ in restaurant $j$. If a new customer coming into a restaurant $j$ sits down at a new table, then the dish for that table will be $\phi_1$, $\phi_2$, $\phi_3$, or $\phi_4$ with probability $\frac{5}{\gamma+11}$, $\frac{3}{\gamma+11}$, $\frac{2}{\gamma+11}$, or $\frac{1}{\gamma+11}$ respectively, or a new dish with probability $\frac{\gamma}{\gamma+11}$.

Dirichlet$(\alpha/K, \ldots, \alpha/K)$ as $K \to \infty$. If

$$V_k \sim \text{Beta}(1, \alpha); \quad \pi_k = V_k \prod_{i=1}^{k-1} (1 - V_k) \tag{2.15}$$

for $k = 1, \ldots, \infty$, then the infinite vector $\boldsymbol{\pi}$ is a size-biased permutation of the infinite vector of mixture weights for a DP.

Following convention, we will use $\boldsymbol{\pi} \sim \text{GEM}(\alpha)$ to denote the process defined above. Later in this dissertation, it will be convenient to use this notation in defining models'

generative processes—we will use this notation somewhat loosely in that we will generally ignore the size-biased nature of the stick-breaking prior.

# Chapter 3

# Timbral Similarity Estimation Based on the Hierarchical Dirichlet Process

## 3.1 Introduction and Previous Work

The first problem we address in this dissertation is automatic timbral similarity estimation, the problem of estimating how similar two pieces of recorded music sound to one another. Our technique is based on the hierarchical Dirichlet process, a flexible Bayesian model for uncovering latent structure in high-dimensional data.

One approach to computing the timbral similarity of two songs is to train a single Gaussian or a Gaussian Mixture Model (GMM) on the Mel-Frequency Cepstral Coefficient (MFCC) feature vectors for each song and compute (for the single Gaussian) or approximate (for the GMM) the Kullback-Leibler (K-L) divergence between the two models [8]. The basic single Gaussian approach with full covariance matrix ("G1" [71]) has been successful, forming the core of top-ranked entries to the MIREX similarity evaluation task for years [2].

Although MFCC data are not normally distributed within songs, using a richer model such as the GMM to more accurately represent their true distribution provides little or no improvement in numerous studies [71, 51, 8]. This suggests that a "glass ceiling" has been reached for this type of representation. Moreover, the computational cost of the Monte Carlo estimation procedure involved in comparing two GMMs is orders of magnitude more than that incurred by computing the K-L divergence between two single Gaussians exactly. This is a very significant issue if we want to compute similarity matrices for large sets of songs. Since the number of comparisons between models that must be done grows quadratically with the number of songs, the cost of model comparison quickly dominates the cost of model fitting.

Another approach [10] produced results statistically indistinguishable from the other top algorithms in MIREX 2007 by using a mid-level semantic feature representation to compute similarity. Using painstakingly human-labeled data, Barrington et al. trained GMMs to estimate the posterior likelihood that a song was best characterized by each of 146 words. These models then produced a vector for each test song defining a multinomial

distribution over the 146 semantic concepts. To compute the dissimilarity of two songs, the K-L divergence between these multinomial distributions for the songs was computed.

The success of this method suggests that alternative statistical representations of songs are worth exploring. Rather than take a supervised approach requiring expensive hand-labeled data, we make use of the Hierarchical Dirichlet Process (HDP), which automatically discovers latent structure within and across groups of data (songs, in our case). This latent structure generates a compact alternative representation of each song, and the model provides a natural and efficient way of comparing songs using K-L divergence.

## 3.2 HDP-Based Similarity Using Latent Features

Recall from chapter 2 that the hierarchical Dirichlet process (HDP) is an extension of the Dirichlet process (DP) designed to model grouped data. Here, our data will be sets of MFCC feature vectors grouped by the song that they came from. Using the HDP, we will infer the shared latent structure in these MFCC vectors, and represent each song in terms of that latent structure.

We will assume the following generative process is responsible for generating our data:

$$\begin{aligned} \boldsymbol{\phi}_k = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\} \sim \mathcal{NIW}(\kappa_0, \nu_0, \boldsymbol{\Lambda}_0, \boldsymbol{\mu}_0); \quad \boldsymbol{\beta} \sim \text{GEM}(\gamma); \\ \boldsymbol{\pi}_j \sim \text{Dirichlet}(\alpha\boldsymbol{\beta}); \quad z_{ji} \sim \text{Multinomial}(\boldsymbol{\pi}_j); \quad \boldsymbol{y}_{ji} \sim \mathcal{N}(\boldsymbol{\mu}_{z_{ji}}, \boldsymbol{\Sigma}_{z_{ji}}), \end{aligned} \tag{3.1}$$

where $\boldsymbol{y}_{ji}$ is the $i$th MFCC feature vector in song $j$, $z_{ji}$ is the index of the latent mixture component used to generate $\boldsymbol{y}_{ji}$, $\boldsymbol{\pi}_j$ is the (infinite) vector of mixture weights over components for song $j$, $\boldsymbol{\beta}$ is the (infinite) vector of global mixture weights ensuring that components are shared between songs, and $\boldsymbol{\phi}_k$ denotes the set of mean and covariance parameters for latent mixture component $k$. $\mathcal{N}$ denotes the normal distribution, $\mathcal{NIW}$ denotes the normal-inverse-Wishart distribution, and GEM denotes the stick-breaking prior over infinite vectors that sum to 1.

We use the direct assignment Gibbs sampler derived in [85] to approximate the posterior distribution over the cluster assignments $\boldsymbol{z}$ and the global mixture weight vector $\boldsymbol{\beta}$, integrating out all other parameters to speed convergence. To represent the infinite vector $\boldsymbol{\beta}$, we only need to explicitly represent the weights for those mixture components actually associated with an observation—i.e. we only represent $\beta_k$ explicitly if $z_{ji} = k$ for some $j, i$. The remainder of the probability mass in $\boldsymbol{\beta}$ is assigned to unused latent components, which are indistinguishable from one another since no data is associated with them.

### 3.2.1 Representing Songs Using the HDP

The mixture components parameterized by $\boldsymbol{\phi}$ capture the latent structure in the feature data, and the mixture proportion vectors $\boldsymbol{\pi}$ express the feature data for each song in terms of that latent structure. $\boldsymbol{\phi}$ and $\boldsymbol{\pi}_j$ together can describe the empirical distribution of feature vectors for a song $j$ as richly as a GMM can, but the HDP does not require that we choose a fixed value of $K$, and represents the songs in a more compact way.

To compute the dissimilarity between two songs $i$ and $j$ given $\boldsymbol{z}$ and $\boldsymbol{\pi}$, we can compute the symmetrized Kullback-Leibler (KL) divergence between the posterior distributions $p(\boldsymbol{\pi}_i|\boldsymbol{\beta}, \boldsymbol{z})$ and $p(\boldsymbol{\pi}_j|\boldsymbol{\beta}, \boldsymbol{z})$, which have the form

$$p(\boldsymbol{\pi}_i|\boldsymbol{\beta}, \boldsymbol{z}) = \text{Dirichlet}(\boldsymbol{\pi}_i; \beta_1 + n_{j1}, \ldots, \beta_K + n_{jK}, \beta_{K+1}), \tag{3.2}$$

where $K$ is the number of components for which some $z_{ji} = k$, and $\beta_{K+1}$ is the probability mass associated with unseen components in the (infinite) global mixture weight vector.

This allows us to compare two songs in terms of the latent structure of their feature data, rather than directly comparing their distributions over the low-level features as the G1 algorithm and GMM-based algorithms do. The KL divergence between these two posteriors can be efficiently computed. The KL divergence between two Dirichlet distributions with parameters $v$ and $w$ each of length $K$ is:

$$D(\text{Dirichlet}(v)||\text{Dirichlet}(w)) = \log \frac{\Gamma(\sum v)}{\Gamma(\sum w)} + \sum_{s=1}^{K} \frac{\log(\Gamma(w_s))}{\log(\Gamma(v_s))} + \sum_{s=1}^{K}((v_s - w_s)(\Psi(v_s) - \Psi(\sum v))$$

where $\Gamma(x)$ is the gamma function, $\Psi(x)$ is the digamma function (the first derivative of the log gamma function), and $\sum v$ and $\sum w$ denote the sum of the K elements of $v$ and $w$ respectively.

For moderate numbers of mixture components, this is less expensive to compute than the KL divergence between two high-dimensional multivariate Gaussian densities. It can be sped up further by computing the gamma and digamma terms offline for each song.

To estimate the posterior over $\boldsymbol{\pi}$, we used the sample of $\boldsymbol{z}$ and $\boldsymbol{\beta}$ from the Gibbs sampler that had the highest posterior probability conditioned on the observed data.

### 3.2.2 Generalizing to New Songs

It is important that our approach be scalable to new songs not seen during training. Once we have inferred the global mixture weights $\boldsymbol{\beta}$ and the mixture component parameters $\boldsymbol{\phi}$, we can infer the posterior distribution over the mixture proportions $\boldsymbol{\pi}_{J+1}$ for a new song $J + 1$ conditioned on $\boldsymbol{\beta}$, $\boldsymbol{\phi}$, and the new data $by_{J+1}$ using the same Gibbs sampling techniques originally used to fit the model, holding all other parameters constant.

## 3.3 Evaluation

In this section we describe the experiments we performed to evaluate our approach against G1, GK (the analogous algorithm for K-component GMMs), and a simplified approach based on Vector Quantization (VQ).

### 3.3.1 South by Southwest Dataset

We test our approach on a dataset that we compiled from the South by Southwest (SXSW) 2007 and 2008 festivals' freely distributed "artist showcase" mp3s [3]. We selected a set of up to twenty mp3s (all by different artists to avoid biasing the results) for seven genres: country, electronic, hip-hop, jazz, metal, punk, and rock. Songs that we felt were unrepresentative of their genre were removed or replaced prior to any quantitative evaluations. There were fewer than 20 usable songs available for country (12), jazz (14), and metal (15), so those genres are slightly underrepresented. There are a total of 121 songs in the dataset.

### 3.3.2 Features

All models were trained on the same sets of feature vectors, which for each frame consisted of 13 MFCCs (extracted using jAudio [62]) combined with 26 delta features computed by subtracting the MFCCs for frame $t$ from those at frame $t-1$ and $t-2$, for a total of 39 dimensions. Each frame was approximately 23 ms long, or 512 samples at the files' sampling rate of 22050 Hz, with a hop size of 512 samples (no overlap). 1000 feature vectors were extracted from the middle of each song.

### 3.3.3 Models Evaluated

#### G1

As described above, G1 models each song's distribution over feature vectors with a single multivariate Gaussian distribution with full covariance matrix. Models are compared using the symmetrized KL divergence between these Gaussians.

#### $K$-component GMMs

We train $K$-component GMMs for each song using the EM algorithm [25]. The symmetrized KL divergence between models is approximated by drawing 1000 synthetic feature vectors from the trained models and evaluating their log likelihoods under both models [8]. This approach is evaluated for $K = 5$, 10, 20, and 30.

#### VQ Codebook

This algorithm is meant to be a simple approximation to the HDP method we outlined above. First, we cluster all of the feature vectors for all songs into $K$ groups using the k-means algorithm, renormalizing the data so that all dimensions have unit standard deviation. This defines a codebook of $K$ cluster centers that identifies every feature vector with the cluster center to which it is closest in Euclidean space. For each song $j$, we compute the vector $\boldsymbol{\pi}_{j,1\ldots K}$ of the relative frequencies of each cluster label. Each $\boldsymbol{\pi}_{j,1\ldots K}$ defines a multinomial distribution over clusters, and we compute the distance between songs as the symmetrized KL divergence between these multinomial distributions (smoothed by a factor of $10^{-5}$ to prevent numerical issues).

| G1 | G5 | G10 | G20 | G30 | VQ5 | VQ10 | VQ30 | VQ50 | VQ100 | HDP |
|---|---|---|---|---|---|---|---|---|---|---|
| 13.24 | 829 | 1487 | 2786 | 4072 | 0.58 | 0.59 | 0.63 | 0.686 | 0.85 | **0.25** |

Table 3.1: Time in seconds required to compute a 121x121 distance matrix for G1, GMM-based ($K = 5, 10, 20, 30$), VQ-based ($K = 5, 10, 30, 50, 100$), and HDP-based algorithms.

This algorithm, like our HDP-based method, represents each song as a multinomial distribution over latent cluster identities discovered using an unsupervised algorithm, and lets us see how a much simpler algorithm that uses similar ideas performs compared with the HDP. Within weeks of our first presenting this VQ-based approach at ISMIR 2008, a very similar approach was proposed at DAFX 2008 [78].

**HDP**

We fit the HDP to all of the data using the direct assignment Gibbs sampler from [85], inferring, inferring posterior distributions over $\boldsymbol{\pi}_j$ for each song $j$ and computing the dissimilarity between two songs $i$ and $j$ as the symmetrized KL divergence between the posteriors over $\pi_i$ and $\pi_j$. We place vague gamma priors on $\alpha$ and $\gamma$ [85]:

$$\alpha \sim \text{Gamma}(1, 0.1), \quad \gamma \sim \text{Gamma}(1, 0.1) \tag{3.3}$$

and learn them during inference. For the prior $H$ over $\boldsymbol{\phi}$, we use the normal-inverse-Wishart distribution [36] with parameters $\kappa_0 = 2$, $\nu_0 = 41$ (the number of dimensions plus two), and $\mu_0 = \bar{y}$ (the mean of all feature vectors across songs). The normal-inverse-Wishart matrix parameter $\Lambda_0$ was chosen by averaging the covariance matrices from 100 clusters of feature vectors, each of which was obtained by choosing a feature vector at random and choosing the 24,200 feature vectors closest to it under a Euclidean distance metric. (The number 24,200 was chosen because it was $1/5$ of the total number of points.) The goal of this process is to choose a matrix $\Lambda_0$ that resembles the covariance matrix of fairly large cluster of points, encouraging the model to find similarly shaped clusters. Using smaller (larger) clusters to choose $\Lambda_0$ would result in the model creating more (fewer) latent topics to explain the data.

### 3.3.4 Experiments

Since human-labeled ground truth similarity data is inherently expensive and difficult to acquire, we follow previous researchers [8, 71, 92] in using genre as a proxy for similarity. We assume that all songs labeled with the same genre are "similar," which allows us to use evaluation metrics from the information retrieval literature. We first compute a full 121x121 distance matrix between all songs using each algorithm. For each query song $s_q$, each other song $s_i$ is given a rank $r_{q,i}$ based on its similarity to $s_q$. The quality of this ranking, i.e. how well it does at ranking songs of the same genre as $s_q$ more similar than songs of different genres, is summarized using R-Precision (RP), Average Precision (AP), and the Area Under the ROC Curve (AUC), which are standard metrics from the

| | G1 | G5 | G10 | G20 | G30 |
|---|---|---|---|---|---|
| RP | 0.3254 | 0.3190 | 0.3287 | 0.3144 | 0.3146 |
| AP | 0.3850 | 0.3761 | 0.3746 | 0.3721 | 0.3706 |
| AUC | 0.6723 | 0.6712 | 0.6687 | 0.6679 | 0.6661 |

| | VQ5 | VQ10 | VQ30 | VQ50 | VQ100 | HDP |
|---|---|---|---|---|---|---|
| RP | 0.2659 | 0.2997 | 0.3191 | 0.340 | 0.3313 | **0.3495** |
| AP | 0.3171 | 0.3546 | 0.3850 | 0.3989 | 0.3910 | **0.3995** |
| AUC | 0.6513 | 0.6675 | 0.6846 | 0.6893 | 0.6758 | **0.7002** |

Table 3.2: Three measures of retrieval quality: mean R-Precision (RP), mean Average Precision (AP), and mean Area Under ROC Curve (AUC) for G1, GMM-based ($K = 5, 10, 20, 30$), VQ-based ($K = 5, 10, 30, 50, 100$), and HDP-based algorithms on the large SXSW dataset.

| | G1 | HDP |
|---|---|---|
| RP | 0.5486 | **0.6000** |
| AP | 0.6807 | **0.7154** |
| AUC | 0.8419 | **0.8983** |

Table 3.3: Mean R-Precision (RP), mean Average Precision (AP), and mean Area Under ROC Curve (AUC) for G1 and our HDP-based algorithm on the smaller dataset.

information retrieval literature [61]. All experiments were conducted on a MacBook Pro with a 2.0 GHz Intel Core Duo processor and 2 GB of RAM. All models were implemented in MATLAB.

#### Testing on Additional Data

To test our HDP-based method's ability to generalize to unseen data using the method in section 3.2.2, we use the HDP trained on the large SXSW set to compute a similarity matrix on a smaller set consisting of 5 artist-filtered songs per genre (35 in all) by artists not in the training set. The electronic, punk, rap, and rock songs came from the SXSW artist showcase collection, and the country, jazz, and metal songs came from a dataset previously used by George Tzanetakis [91]. We also compute a similarity matrix on this dataset using G1, and compare the RP, AP, and AUC metrics for retrieval quality obtained using both algorithms.

## 3.4 Results

Tables 3.1, 3.2, and 3.3 summarize the results of our experiments. The best results in each row are in bold.

The amount of time required to compute the distance matrices for the GMMs was, as expected, enormous by comparison to the other models. The cost of computing the KL

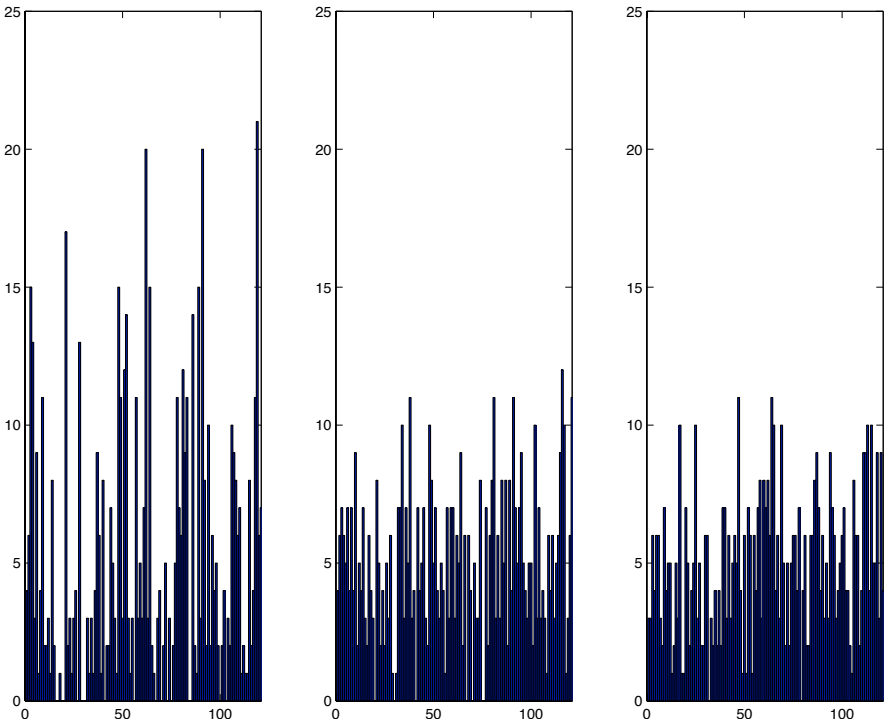


Figure 3.1: Histograms of how often each song is ranked in the top five of another song's similarity list for similarity matrices obtained using G1 (left), the HDP (center), and by choosing distances at random (right).

divergence for the VQ-based and HDP-based models was more than an order of magnitude lower even than the cost of computing the KL divergence between single Gaussians. This is due to the costs involved in working with the Gaussians' 39-by-39-dimensional covariance matrices.

The HDP performed better than the other models for all three standard information retrieval metrics, although the VQ model with $K = 50$ was a very close second. None of the GMMs outperformed G1.

The results in table 3.3 show that the HDP-based approach does generalize well to new songs, showing that the algorithm can be scaled up efficiently to databases of many songs.

### 3.4.1 Similarity Hubs

The G1 and GK approaches are known to produce "hubs" [8]—an undesirable phenomenon where certain songs are found to be similar to many other songs. The hub phenomenon is a potentially serious concern, since it can result in very bad matches being selected as similar to a query song.

Our HDP-based approach does not suffer from this problem. Figure 3.1 shows how often each song is ranked in the top five of another song's similarity list for similarity matrices obtained from G1, the HDP, and choosing distances at random. The randomly generated histogram shows the sort of distribution of hubs one would expect to see due to chance in a dataset of this size. The HDP's histogram closely resembles the random one, indicating an absence of abnormal hubs. G1's histogram, by contrast, shows more severe and more numerous hubs than the other two histograms.

## 3.5 Discussion

We developed a new method for assessing the similarity between songs. Our HDP-based approach outperformed the G1 algorithm, can compute large distance matrices efficiently, and does not suffer from the "hub" problem where some songs are found to be similar to all other songs. Since our approach does not have access to any information about temporal structure beyond that provided by the MFCC deltas (about 69 ms in total), we expect that combining the distances it provides with fluctuation patterns or some similar feature set would provide an improvement in similarity performance, as it does for the G1C algorithm [71].

# Chapter 4

# Codeword Bernoulli Averaging: A Model for Autotagging Songs

It has been said that talking about music is like dancing about architecture[1], but people nonetheless use words to describe music. In this chapter we will present a simple system that addresses tag prediction from audio—the problem of predicting what words people would be likely to use to describe a song.

Two direct applications of tag prediction are semantic annotation and retrieval. If we have an estimate of the probability that a tag applies to a song, then we can say what words in our vocabulary of tags best describe a given song (automatically annotating it) and what songs in our database a given word best describes (allowing us to retrieve songs from a text query).

We present the Codeword Bernoulli Average (CBA) model, a probabilistic model that attempts to predict the probability that a tag applies to a song based on a vector-quantized (VQ) representation of that song's audio. Our CBA-based approach to tag prediction

- Is easy to implement using a simple EM algorithm.
- Is fast to train.
- Makes predictions efficiently on unseen data.
- Performs as well as or better than previous state-of-the-art approaches.

We by no means claim that CBA is the last word in autotagging. CBA was designed to get maximum performance with minimum complexity, and in recent MIREX competitions it has been outperformed by more sophisticated (and complex) methods [2]. Nonetheless, it remains useful as a fast and simple baseline method, as seen in several recent ISMIR papers (e.g. [24, 65]).

[1] Attribution for this quip is difficult to track down—Elvis Costello, Martin Mull, and Thelonius Monk have all been advanced as possible sources. For a full discussion see `http://www.pacifier.com/~ascott/they/tamildaa.htm`

## 4.1 Data and Representation

### 4.1.1 The CAL500 data set

We train and test our method on the CAL500 dataset [88, 89]. CAL500 is a corpus of 500 tracks of Western popular music, each of which has been manually annotated by at least three human labelers. We used the "hard" annotations provided with CAL500, which give binary values $y_{jw} \in \{0, 1\}$ for all songs $j$ and tags $w$. $y_{jw} = 1$ indicates that tag $w$ applies to song $j$, $y_{jw} = 0$ indicates that it does not.

CAL500 is distributed with a set of 10,000 39-dimensional Mel-Frequency Cepstral Coefficient Delta (delta-MFCC) feature vectors for each song. Each delta-MFCC vector summarizes the timbral evolution of three successive 23ms windows of a song. CAL500 provides these feature vectors in a random order, so no temporal information beyond a 69ms timescale is available.

Our goals are to use these features to predict which tags apply to a given song and which songs are characterized by a given tag. The first task yields an automatic annotation system, the second yields a semantic retrieval system.

### 4.1.2 A vector-quantized representation

Rather than work directly with the delta-MFCC feature representation, we first vector quantize all of the feature vectors in the corpus, ignoring for the moment what feature vectors came from what songs. We:

1. Normalize the feature vectors so that they have mean 0 and standard deviation 1 in each dimension.

2. Run the k-means algorithm [59] on a subset of randomly selected feature vectors to find a set of $K$ cluster centroids.

3. For each normalized feature vector $f_{ji}$ in song $j$, assign that feature vector to the cluster $k_{ji}$ with the smallest squared Euclidean distance to $f_{ji}$.

This vector quantization procedure allows us to represent each song $j$ as a vector $\boldsymbol{n}_j$ of counts of a discrete set of codewords:

$$n_{jk} = \sum_{i=1}^{N_j} 1(k_{ji} = k) \tag{4.1}$$

where $n_{jk}$ is the number of feature vectors assigned to codeword $k$, $N_j$ is the total number of feature vectors in song $j$, and $1(a = b)$ is a function returning 1 if $a = b$ and 0 if $a \neq b$.

This discrete "bag-of-codewords" representation is less rich than the original continuous feature vector representation. However, as seen in the previous chapter, it yields competitive results for the problem of timbral similarity. VQ codebook-based representations have also produced state-of-the-art performance in image annotation and retrieval systems [95].

## 4.2 The Codeword Bernoulli Average model

In order to predict what tags will apply to a song and what songs are characterized by a tag, we developed the Codeword Bernoulli Average model (CBA). CBA models the conditional probability of a tag $w$ appearing in a song $j$ conditioned on the empirical distribution $\boldsymbol{n}_j$ of codewords extracted from that song. One we have estimated CBA's hidden parameters from our training data, we will be able to quickly estimate this conditional probability for new songs.

### 4.2.1 Related work

One class of approaches treats audio tag prediction as a set of binary classification problems to which variants of standard classifiers such as the Support Vector Machine (SVM) [60, 87] or AdaBoost [13] can be applied. Once a set of classifiers has been trained, the classifiers attempt to predict whether or not each tag applies to previously unseen songs. These predictions come with confidence scores that can be used to rank songs by relevance to a given tag (for retrieval), or to rank tags by relevance to a given song (for annotation).

Classifiers like SVMs or AdaBoost focus on binary classification accuracy rather than directly optimizing the continuous confidence scores that are used for retrieval tasks—this goal leads them to focus on the examples that are most difficult to classify and largely ignore those for which their confidence scores are reasonably high. In a retrieval system that ranks results by confidence scores, however, these difficult examples near the margin of the classification boundary are less likely to be returned than easier examples, suggesting that these classifiers may be optimizing the wrong objective for the task.

Another approach is to fit a generative probabilistic model such as a Gaussian Mixture Model (GMM) for each tag to the audio feature data for all of the songs manifesting that tag [89]. The posterior likelihood $p(\text{tag}|\text{audio})$ of the feature data for a new song being generated from the model for a particular tag is then used to estimate the relevance of that tag to that song (and vice versa). Although this model tells us how to generate the audio feature data for a song conditioned on a *single* tag, it does not define a generative process for songs with *multiple* tags, and so heuristics are necessary to estimate the posterior likelihood of a set of tags.

Rather than assuming that the audio for a song depends on the tags associated with that song, we will assume that the tags depend on the audio data. This will yield a probabilistic model with a discriminative flavor, and a more coherent generative process than that in [89].

### 4.2.2 Generative process

CBA models a collection of binary random variables $\boldsymbol{y}$, with $y_{jw} \in \{0, 1\}$ determining whether or not tag $w$ applies to song $j$. These variables are generated in two steps. First, a codeword $z_{jw} \in \{1, \ldots, K\}$ is selected with probability proportional to the number of

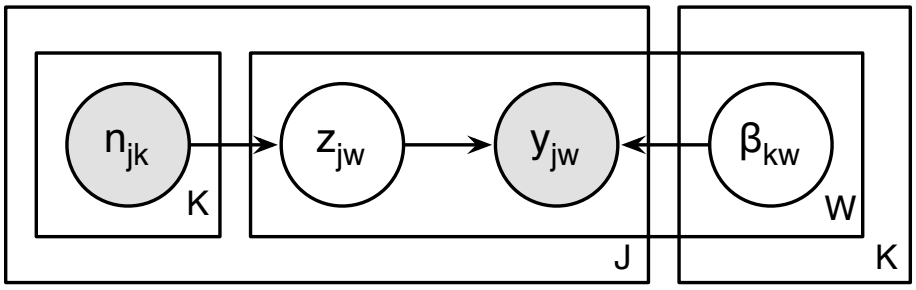


Figure 4.1: Graphical model representation of CBA.

times $n_{jk}$ that that codeword appears in song $j$'s feature data:

$$p(z_{jw} = k|\boldsymbol{n}_j, N_j) = \frac{n_{jk}}{N_j} \tag{4.2}$$

Then a value for $y_{jw}$ is chosen from a Bernoulli distribution with parameter $\beta_{kw}$:

$$\begin{aligned} p(y_{jw} = 1|z_{jw}, \boldsymbol{\beta}) &= \beta_{z_{jw}w} \\ p(y_{jw} = 0|z_{jw}, \boldsymbol{\beta}) &= 1 - \beta_{z_{jw}w} \end{aligned} \tag{4.3}$$

The full joint distribution over $\boldsymbol{z}$ and $\boldsymbol{y}$ conditioned on the observed counts of codewords $\boldsymbol{n}$ is:

$$p(\boldsymbol{z}, \boldsymbol{y}|\boldsymbol{n}) = \prod_w \prod_j \frac{n_{jz_{jw}}}{N_j} \beta_{z_{jw}w}^{y_{jw}} (1 - \beta_{z_{jw}w})^{(1-y_{jw})}. \tag{4.4}$$

The random variables in CBA and their dependencies are summarized in figure 4.1.

### 4.2.3 Inference using expectation-maximization

We fit CBA with maximum-likelihood (ML) estimation. Our goal is to estimate a set of values for our Bernoulli parameters $\boldsymbol{\beta}$ that will maximize the likelihood $p(\boldsymbol{y}|\boldsymbol{n}, \boldsymbol{\beta})$ of the observed tags $\boldsymbol{y}$ conditioned on the VQ codeword counts $\boldsymbol{n}$ and the parameters $\boldsymbol{\beta}$. Analytic ML estimates for $\boldsymbol{\beta}$ are not available because of the latent variables $\boldsymbol{z}$. We use the Expectation-Maximization (EM) algorithm to do maximum-likelihood estimation in the presence of the latent variables $\boldsymbol{z}$ [25].

Each iteration of EM operates in two steps. In the expectation ("E") step, we compute the posterior of the latent variables $\boldsymbol{z}$ given our current estimates for the parameters $\boldsymbol{\beta}$. We define a set of expectation variables $h_{jwk}$ corresponding to the posterior $p(z_{jw} = k|\boldsymbol{n}, \boldsymbol{y}, \boldsymbol{\beta})$:

$$h_{jwk} = p(z_{jw} = k|\boldsymbol{n}, \boldsymbol{y}, \boldsymbol{\beta}) \tag{4.5}$$

$$= \frac{p(y_{jw}|z_{jw} = k, \boldsymbol{\beta}) p(z_{jw} = k|\boldsymbol{n})}{p(y_{jw}|\boldsymbol{n}, \boldsymbol{\beta})} \tag{4.6}$$

$$= \begin{cases} \frac{n_{jk}\beta_{kw}}{\sum_{i=1}^{K} n_{ji}\beta_{iw}} & \text{if } y_{jw} = 1 \\ \frac{n_{jk}(1-\beta_{kw})}{\sum_{i=1}^{K} n_{ji}(1-\beta_{iw})} & \text{if } y_{jw} = 0 \end{cases} \tag{4.7}$$

In the maximization ("M") step, we find maximum-likelihood estimates of the parameters $\boldsymbol{\beta}$ given the expected posterior sufficient statistics:

$$\begin{aligned} \beta_{kw} &\leftarrow \mathbb{E}[y_{jw}|z_{jw} = k, \boldsymbol{h}] && (4.8) \\ &= \frac{\sum_j p(z_{jw} = k|\boldsymbol{h}) y_{jw}}{\sum_j p(z_{jw} = k|\boldsymbol{h})} && (4.9) \\ &= \frac{\sum_j h_{jwk} y_{jw}}{\sum_j h_{jwk}} && (4.10) \end{aligned}$$

By iterating between computing $\boldsymbol{h}$ (using equation 4.7) and updating $\boldsymbol{\beta}$ (using equation 4.10), we find a set of values for $\boldsymbol{\beta}$ that maximize the likelihood of the training data under the CBA model. Since the log-likelihood function

$$\log p(\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{n}) = \sum_j \sum_w y_{jw} \log(\textstyle\sum_k n_{jk}\beta_{kw}) + (1 - y_{jw}) \log(\textstyle\sum_k n_{jk}(1 - \beta_{kw})) - \log(N_j) \tag{4.11}$$

is concave in $\boldsymbol{\beta}$, we are assured of finding a global maximum [2].

### 4.2.4 Generalizing to new songs

Once we have inferred a set of Bernoulli parameters $\boldsymbol{\beta}$ from our training dataset, we can use them to infer the probability that a tag $w$ will apply to a previously unseen song $j$ based on the counts $\boldsymbol{n}_j$ of codewords for that song:

$$\begin{aligned} p(y_{jw}|\boldsymbol{n}_j, \boldsymbol{\beta}) &= \sum_k p(z_{jw} = k|\boldsymbol{n}_j) p(y_{jw}|z_{jw} = k) \\ p(y_{jw} = 1|\boldsymbol{n}_j, \boldsymbol{\beta}) &= \frac{1}{N_j} \sum_k n_{jk}\beta_{kw} && (4.12) \end{aligned}$$

As a shorthand, we will refer to our inferred value of $p(y_{jw} = 1|\boldsymbol{n}_j, \boldsymbol{\beta})$ as $s_{jw}$.

Once we have inferred $s_{jw}$ for all of our songs and tags, we can use these inferred probabilities both to retrieve the songs with the highest probability of having a particular tag and to annotate each song with a subset of our vocabulary of tags. In a retrieval system, we return the songs in descending order of $s_{jw}$. To do automatic tagging, we could annotate each song with the $M$ most likely tags for that song. However, this may lead to our annotating many songs with common, uninformative tags such as "Not Bizarre/Weird" and a lack of diversity in our annotations. To compensate for this, we use a simple heuristic: we introduce a "diversity factor" $d$ and discount each $s_{jw}$ by $d$ times the mean of the estimated probabilities $s_{\cdot w}$. A higher value of $d$ will make less common tags more likely to appear in annotations, which may lead to less accurate but more informative annotations. The diversity factor $d$ has no impact on retrieval.

The cost of computing each $s_{jw}$ using equation 4.12 is linear in the number of codewords $K$, and the cost of vector quantizing new songs' feature data using the previously

[2]

| Model | Precision | Recall | F-Score |
|---|---|---|---|
| UpperBnd | 0.712 (0.007) | 0.375 (0.006) | 0.491 |
| Random | 0.144 (0.004) | 0.064 (0.002) | 0.089 |
| MixHier | 0.265 (0.007) | 0.158 (0.006) | 0.198 |
| Autotag (MFCC) | 0.281 | 0.131 | 0.179 |
| Autotag (afeats exp.) | **0.312** | 0.153 | 0.205 |
| CBA $K = 5$ | 0.198 (0.006) | 0.107 (0.005) | 0.139 |
| CBA $K = 10$ | 0.214 (0.006) | 0.111 (0.006) | 0.146 |
| CBA $K = 25$ | 0.247 (0.007) | 0.134 (0.007) | 0.174 |
| CBA $K = 50$ | 0.257 (0.009) | 0.145 (0.007) | 0.185 |
| CBA $K = 100$ | 0.263 (0.007) | 0.149 (0.004) | 0.190 |
| CBA $K = 250$ | 0.279 (0.007) | 0.153 (0.005) | 0.198 |
| CBA $K = 500$ | 0.286 (0.005) | **0.162 (0.004)** | **0.207** |
| CBA $K = 1000$ | 0.283 (0.008) | 0.161 (0.006) | 0.205 |
| CBA $K = 2500$ | 0.282 (0.006) | **0.162 (0.004)** | 0.206 |

| Model | AP | AROC |
|---|---|---|
| UpperBnd | 1 | 1 |
| Random | 0.231 (0.004) | 0.503 (0.004) |
| MixHier | 0.390 (0.004) | 0.710 (0.004) |
| Autotag (MFCC) | 0.305 | 0.678 |
| Autotag (afeats exp.) | 0.385 | 0.674 |
| CBA $K = 5$ | 0.328 (0.009) | 0.707 (0.007) |
| CBA $K = 10$ | 0.336 (0.007) | 0.715 (0.007) |
| CBA $K = 25$ | 0.352 (0.008) | 0.734 (0.008) |
| CBA $K = 50$ | 0.366 (0.009) | 0.746 (0.008) |
| CBA $K = 100$ | 0.372 (0.007) | 0.748 (0.008) |
| CBA $K = 250$ | 0.385 (0.007) | 0.760 (0.007) |
| CBA $K = 500$ | 0.390 (0.008) | 0.759 (0.007) |
| CBA $K = 1000$ | 0.393 (0.008) | 0.764 (0.006) |
| CBA $K = 2500$) | **0.394 (0.008)** | **0.765 (0.007)** |

Table 4.1: Summary of the performance of CBA (with a variety of VQ codebook sizes $K$), a mixture-of-Gaussians model (MixHier), and an AdaBoost-based model (Autotag) on an annotation task (evaluated using precision, recall, and F-score) and a retrieval task (evaluated using average precision (AP) and area under the receiver-operator curve (AROC)). Autotag (MFCC) used the same Delta-MFCC feature vectors and training set size of 450 songs as CBA and MixHier. Autotag (afeats exp.) used a larger set of features and a larger set of training songs. UpperBnd uses the optimal labeling for each evaluation metric, and shows the upper limit on what any system can achieve. Random is a baseline that annotates and ranks songs randomly.

computed centroids obtained using k-means is linear in the number of features, the number

of codewords $K$, and the length of the song. For practical values of $K$, the total cost of estimating the probability that a tag applies to a song is comparable to the cost of feature extraction. Our approach can therefore tag new songs efficiently, an important feature for large commercial music databases.

## 4.3 Evaluation

We evaluated our model's performance on an annotation task and a retrieval task using the CAL500 data set. We compare our results on these tasks with two other sets of published results for these tasks on this corpus: those obtained by Turnbull et al. using mixture hierarchies estimation to learn the parameters to a set of mixture-of-Gaussians models [89], and those obtained by Bertin-Mahieux et al. using a discriminative approach based on the AdaBoost algorithm [13]. In the 2008 MIREX audio tag classification task, the approach in [89] was ranked either first or second according to all metrics measuring annotation or retrieval performance [2].

### 4.3.1 Annotation task

To evaluate our model's ability to automatically tag unlabeled songs, we measured its average per-word precision and recall on held-out data using tenfold cross-validation.

First, we vector quantized our data using k-means. We tested VQ codebook sizes from $K = 5$ to $K = 2500$. After finding a set of $K$ centroids using k-means on a randomly chosen subset of 125,000 of the Delta-MFCC vectors (250 feature vectors per song), we labeled each Delta-MFCC vector in each song with the index of the cluster centroid whose squared Euclidean distance to that vector was smallest. Each song $j$ was then represented as a $K$-dimensional vector $\boldsymbol{n}_j$, with $n_{jk}$ giving the number of times label $k$ appeared in song $j$, as described in equation 4.1.

We ran a tenfold cross-validation experiment modeled after the experiments in [89]. We split our data into 10 disjoint 50-song test sets at random, and for each test set

1. We iterated the EM algorithm described in section 4.2.3 on the remaining 450 songs to estimate the parameters $\boldsymbol{\beta}$. We stopped iterating once the negative log-likelihood of the training labels conditioned on $\boldsymbol{\beta}$ and $\boldsymbol{n}$ decreased by less than 0.1% per iteration.

2. Using equation 4.12, for each tag $w$ and each song $j$ in the test set we estimated $p(y_{jw}|\boldsymbol{n}_j, \boldsymbol{\beta})$, the probability of song $j$ being characterized by tag $w$ conditioned on $\boldsymbol{\beta}$ and the vector quantized feature data $\boldsymbol{n}_j$.

3. We subtracted $d = 1.25$ times the average conditional probability of tag $w$ from our estimate of $p(y_{jw}|\boldsymbol{n}_j, \boldsymbol{\beta})$ for each song $j$ to get a score $s_{jw}$ for each song.

4. We annotated each song $j$ with the ten tags with the highest scores $s_{jw}$.

To evaluate our system's annotation performance, we computed the average per-word precision, recall, and F-score. Per-word recall is defined as the average fraction of songs

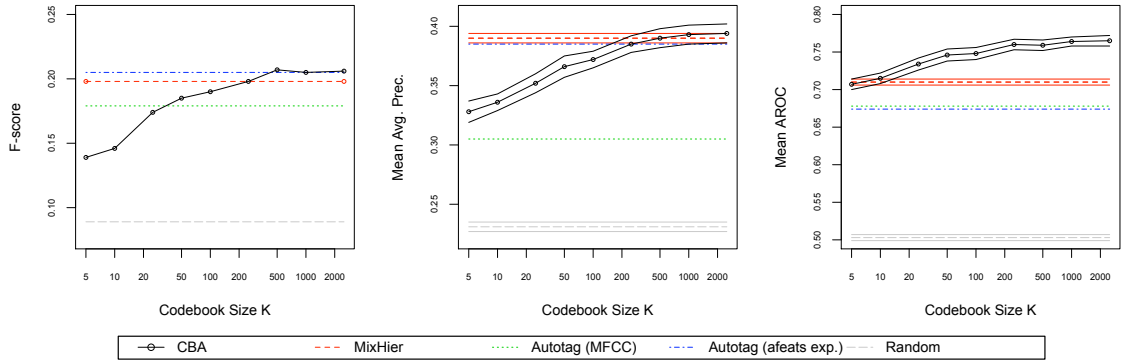


Figure 4.2: Visual comparison of the performance of several models evaluated using F-score, mean average precision, and area under receiver-operator curve (AROC).

actually labeled $w$ that our model annotates with label $w$. Per-word precision is defined as the average fraction of songs that our model annotates with label $w$ that are actually labeled $w$. F-score is the harmonic mean of precision and recall, and is one metric of overall annotation performance.

Following [89], when our model does not annotate any songs with a label $w$ we set the precision for that word to be the empirical probability that a word in the dataset is labeled $w$. This is the expected per-word precision for $w$ if we annotate all songs randomly. If no songs in a test set are labeled $w$, then per-word precision and recall for $w$ are undefined, so we ignore these words in our evaluation.

### 4.3.2 Retrieval task

To evaluate our system's retrieval performance, for each tag $w$ we ranked each song $j$ in the test set by the probability our model estimated of tag $w$ applying to song $j$. We evaluated the average precision (AP) and area under the receiver-operator curve (AROC) for each ranking. AP is defined as the average of the precisions at each possible level of recall, and AROC is defined as the area under a curve plotting the percentage of true positives returned against the percentage of false positives returned. As in the annotation task, if no songs in a test set are labeled $w$ then AP and AROC are undefined for that label, and we exclude it from our evaluation for that fold of cross-validation.

### 4.3.3 Annotation and retrieval results

Table 4.1 and figure 4.2 compare our CBA model's average performance under the five metrics described above with other published results on the same dataset. MixHier is Turnbull et al.'s system based on a mixture-of-Gaussians model [89], Autotag (MFCC) is Bertin-Mahieux's AdaBoost-based system using the same Delta-MFCC feature vectors as our model, and Autotag (afeats exp.) is Bertin-Mahieux's system trained using additional features and training data [13]. Random is a random baseline that retrieves songs in a random order and annotates songs randomly based on tags' empirical frequencies. Up-

perBnd shows the best performance possible under each metric. Random and UpperBnd were computed by Turnbull et al., and give a sense of the possible range for each metric.

We tested our model using a variety of codebook sizes $K$ from 5 to 2500. Cross-validation performance improves as the codebook size increases until $K = 500$, at which point it levels off. Our model's performance does not depend strongly on fine tuning $K$, at least within a range of $500 \leq K \leq 2500$.

When using a codebook size of at least 500, our CBA model does at least as well as MixHier and Autotag under every metric except precision. Autotag gets significantly higher precision than CBA when it uses additional training data and features, but not when it uses the same features and training set as CBA.

Tables 4.2 and 4.3 give examples of annotations and retrieval results given by our model during cross-validation.

### 4.3.4 Computational cost

We measured how long it took to estimate the parameters to CBA and to generalize to new songs. All experiments were conducted on one core of a server with a 2.2 GHz AMD Opteron 275 CPU and 16 GB of RAM running CentOS Linux.

Using a MATLAB implementation of the EM algorithm described in 4.2.3, it took 84.6 seconds to estimate CBA's parameters from 450 training songs vector-quantized using a 500-cluster codebook. In experiments with other codebook sizes $K$ the training time scaled linearly with $K$. Once $\boldsymbol{\beta}$ had been estimated, it took less than a tenth of a millisecond to predict the probabilities of 174 labels for a new song.

We found that the vector-quantization process was the most expensive part of training and applying CBA. Finding a set of 500 cluster centroids from 125,000 39-dimensional Delta-MFCC vectors using a C++ implementation of k-means took 479 seconds, and finding the closest of 500 cluster centroids to the 10,000 feature vectors in a song took 0.454 seconds. Both of these figures scaled linearly with the size of the VQ codebook in other experiments.

## 4.4 Discussion

We introduced the Codeword Bernoulli Average model, which predicts the probability that a tag will apply to a song based on counts of vector-quantized feature data extracted from that song. Our model is simple to implement, fast to train, generalizes to new songs efficiently, and yields state-of-the-art performance on annotation and semantic retrieval tasks.

| *Give it Away*<br>the Red Hot Chili Peppers | *Fly Me to the Moon*<br>Frank Sinatra |
|---|---|
| Usage—At a party | Calming/Soothing |
| Heavy Beat | NOT—Fast Tempo |
| Drum Machine | NOT—High Energy |
| Rapping | Laid-back/Mellow |
| Very Danceable | Tender/Soft |
| Genre—Hip Hop/Rap | NOT—Arousing/Awakening |
| Genre (Best)—Hip Hop/Rap | Usage—Going to sleep |
| Texture Synthesized | Usage—Romancing |
| Arousing/Awakening | NOT—Powerful/Strong |
| Exciting/Thrilling | Sad |
| *Blue Monday*<br>New Order | *Becoming*<br>Pantera |
| Very Danceable | NOT—Calming/Soothing |
| Usage—At a party | NOT—Tender/Soft |
| Heavy Beat | NOT—Laid-back/Mellow |
| Arousing/Awakening | Bass |
| Fast Tempo | Genre—Alternative |
| Drum Machine | Exciting/Thrilling |
| Texture Synthesized | Electric Guitar (distorted) |
| Sequencer | Genre—Rock |
| Genre—Hip Hop/Rap | Texture Electric |
| Synthesizer | High Energy |

Table 4.2: Examples of semantic annotation from the CAL500 data set showing the top 10 words associated by our model with the songs *Give it Away, Fly Me to the Moon, Blue Monday,* and *Becoming*.

| Query | Top 5 Retrieved Songs |
|---|---|
| Tender/Soft | John Lennon—Imagine<br>Shira Kammen—Music of Waters<br>Crosby Stills and Nash—Guinnevere<br>Jewel—Enter From the East<br>Yakshi—Chandra |
| Hip Hop | Tim Rayborn—Yedi Tekrar<br>Solace—Laz 7 8<br>Eminem—My Fault<br>Sir Mix-a-Lot—Baby Got Back<br>2-Pac—Trapped |
| Piano | Robert Johnson—Sweet Home Chicago<br>Shira Kammen—Music of Waters<br>Miles Davis—Blue in Green<br>Guns n' Roses—November Rain<br>Charlie Parker—Ornithology |
| Exercising | Tim Rayborn—Yedi Tekrar<br>Monoide—Golden Key<br>Introspekt—TBD<br>Belief Systems—Skunk Werks<br>Solace—Laz 7 8 |
| Screaming | Nova Express—I'm Alive<br>Rocket City Riot—Mine Tonite<br>Seismic Anamoly—Wreckinball<br>Pizzle—What's Wrong With My Footman<br>Jackalopes—Rotgut |

Table 4.3: Examples of semantic retrieval from the CAL500 data set. The left column shows a query word, and the right column shows the five songs in the dataset judged by our system to best match that word.

# Chapter 5

# Nonparametric Latent Source Discovery Part I: The SI-HDP

In previous chapters, we used feature-based representations of the audio signal. Although this approach makes modeling simpler, it can only model the qualities of the mixed audio signal, not of individual sounds that occur simultaneously. In this chapter, we will present the Shift-Invariant Hierarchical Dirichlet Process (SIHDP), a generative model that moves beyond this approach and allows us to represent songs in terms of the instruments and other sounds that generated them. We will be able to decompose an audio spectrogram into a set of short spectrograms corresponding to latent sources present in the audio signal, a set of global activation levels for those sources, and a set of time-varying activation levels for those sources. This representation will yield a higher-level representation of a set of songs, and also allow us to isolate or suppress the energy in a recording associated with one or another latent source.

The same instruments tend to appear in multiple recordings in different combinations, and without hand-generated metadata there is no way of knowing a priori how many or which sources will appear in a given recording. This suggests that a model based on the Hierarchical Dirichlet Process (HDP) would be ideally suited to modeling groups of songs, since it represents groups of observations (such as songs) as being generated by an initially unspecified number of shared latent components [85].

However, the HDP requires that our observations be directly comparable, which is not the case for audio data. Human listeners need to hear how a sound evolves over time to recognize and interpret that sound, but computers cannot directly observe when events in audio signals begin and end. We therefore modified the HDP to make it invariant to shifts in time by explicitly modeling when in each song latent sources appear.

This allows us to discover a shared vocabulary of latent sources that describe different events in our set of songs, and to produce a rough transcription of each song in terms of that shared vocabulary. This transcription provides a rich representation of our songs with which we can compare and analyze our songs.

We perform posterior inference on the SIHDP using Gibbs sampling [37]. To make it feasible to do inference on large data sets in a reasonable amount of time, we also develop an exact parallel Gibbs sampler for the SIHDP that can also be applied to the original HDP.

## 5.1 A Shift-Invariant Nonparametric Bayesian Model

We define a probabilistic generative model for recorded songs. We assume that a song is generated by repeatedly selecting a sonic component from a set of available components and then selecting both a time at which it occurs and an amplitude with which it is manifested. Such a component might be, for example, a snare drum or the note middle C on a piano. Components may overlap in time. (This resembles the process by which a sample-based sequencer produces audio.)

Below, we will present a probabilistic generative model that corresponds to this process. Rather than operate directly on a time-domain representation of audio, we use a quantized time-frequency representation that is robust to imperceptible changes in phase. Further, this representation allows us to adapt existing models designed to handle counts data.

### 5.1.1 Data Representation

We represent each song using a quantized time-frequency spectrogram representation derived from the Short-Time Fourier Transform (STFT). First, we divide the song into a series of $W$ short non-overlapping frames of $S$ samples each. We multiply each frame by the Hann window and compute the magnitude spectrum of the Discrete Fourier Transform (DFT) for that window of samples. This yields $B = \frac{S}{2} + 1$ coefficients giving the amplitude in that window of evenly spaced frequencies from 0 Hz to one half of the sampling rate.

After doing this for each frame, we have a $B \times W$ matrix $\hat{\boldsymbol{y}}_j$ of non-negative real numbers, with $\hat{y}_{jbw}$ giving the amplitude of DFT frequency bin $b$ at time step $w$. Since the overall amplitude scale of digital audio is arbitrary, we can normalize our spectrograms $\hat{\boldsymbol{y}}_j$ so that $\sum_{b=0}^{B} \sum_{w=1}^{W} \hat{y}_{jbw} = 1$, and $\hat{\boldsymbol{y}}_j$ defines a multinomial probability distribution over times $w$ and frequencies $b$.

Finally, we transform each normalized $\hat{\boldsymbol{y}}_j$ into quantized counts data whose empirical distribution approximates $\hat{\boldsymbol{y}}_j$. We multiply the normalized $\hat{\boldsymbol{y}}_j$ by a constant $\nu \times W \times B$ and round the result to get the number of observed magnitude "quanta" $\bar{y}_{jbw}$ in bin $b$ at time $w$ of song $j$:

$$\bar{y}_{jbw} = \text{round}(\nu W B \hat{y}_{jbw}) \tag{5.1}$$

$$N_j = \sum_{b=1}^{B} \sum_{w=1}^{W} \bar{y}_{jbw} \tag{5.2}$$

$\nu$ is roughly the average number of quanta per time/bin pair, and $N_j$ is the total number of observed quanta in song $j$. We use the notation $y_{ji} = \{w_{ji}, b_{ji}\}$ to refer to the $i$th (exchangeable) quantum of energy in song $j$ as occurring at time $w_{ji}$ and bin $b_{ji}$, for $i \in \{1, \ldots, N_j\}$.

$\nu$ impacts the analysis in three ways. Larger values of $\nu$

1. produce less quantization noise,
2. increase the cost of inference, and
3. make the posterior distribution over our model's parameters more peaked.

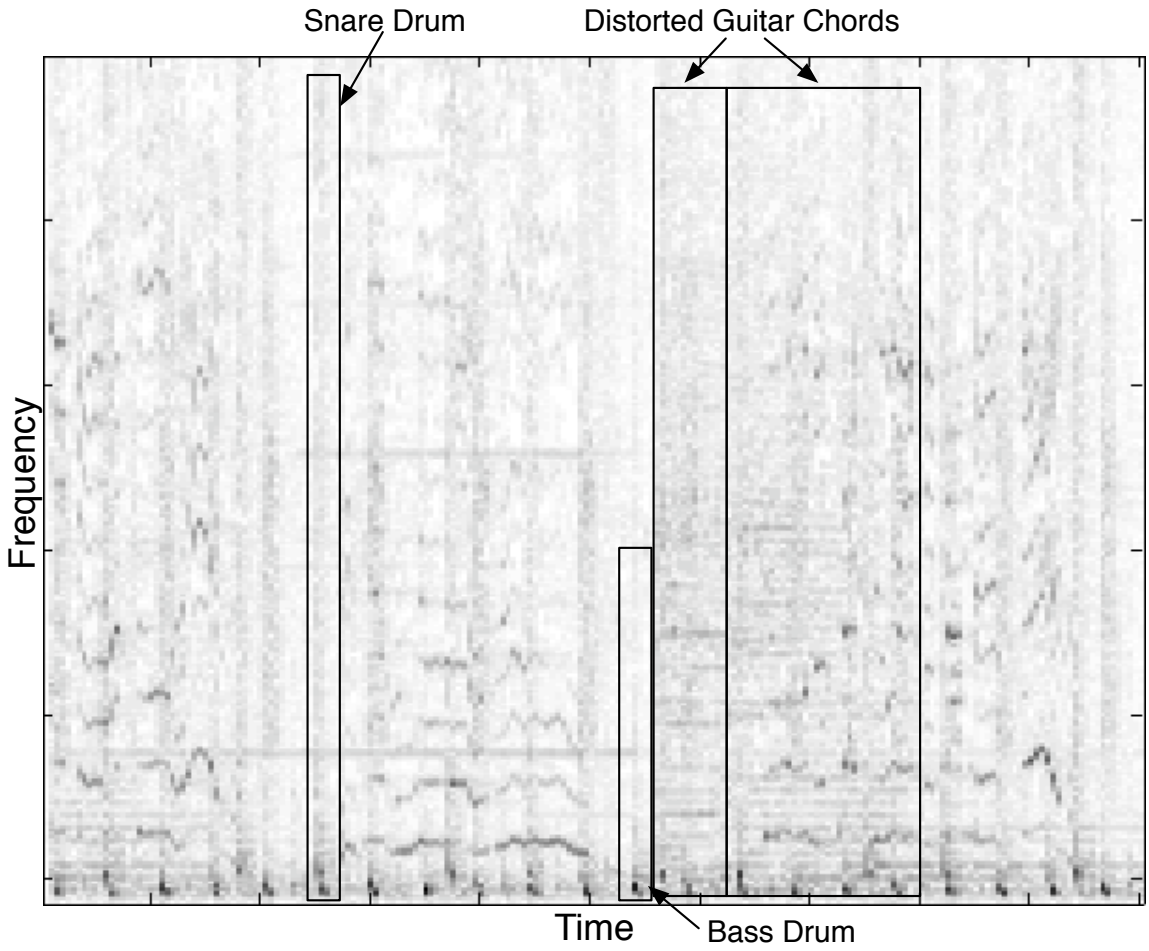


Figure 5.1: Spectrogram of 4.64 seconds (200 512-sample windows) of the AC/DC song "Dirty Deeds Done Dirt Cheap," annotated with the locations in time and frequency of a few instrument sounds.

The first two phenomena present a tradeoff between noise and computational complexity—in practice the sparse distribution of energy in audio spectrograms makes it possible to get away with fairly low levels of $\nu$ (i.e. $\nu < 1$) without significantly affecting the quality of the representation. The third phenomenon is more troubling, since the amount of data given to a Bayesian nonparametric model like the HDP plays a role in how many latent parameters it will use to explain the data. In the following chapter, we will present an alternative latent source decomposition model that addresses this concern, but for the moment we put it aside.

Although we only discuss DFT spectrogram data in this work, our model could also be applied to other time-frequency representations such as the constant-Q spectrogram or the output of a wavelet transform.

### 5.1.2 Generative Process

We present the Shift-Invariant Hierarchical Dirichlet Process (SIHDP), a generative model for our quantized spectrogram data that is an extension of the Hierarchical Dirichlet Process (HDP) [85] with discrete observations. The discrete HDP assumes an infinite number of multinomial mixture components $\boldsymbol{\phi}_k$ drawn independently from a Dirichlet prior with parameter $\epsilon$:

$$\boldsymbol{\phi}_k \sim \text{Dirichlet}(\epsilon, \ldots, \epsilon)$$

An infinite vector $\boldsymbol{\beta}$ defining the global proportions of these components is drawn from a stick-breaking process with concentration parameter $\gamma$ (denoted GEM($\gamma$)). Each group $j$ of observations draws a group-level set of proportions $\boldsymbol{\pi}_j$ from a Dirichlet Process (DP) with concentration parameter $\alpha$ and the multinomial defined by $\boldsymbol{\beta}$ as its base distribution:

$$\boldsymbol{\beta} \sim \text{GEM}(\gamma); \quad \boldsymbol{\pi}_j \sim \text{Dirichlet}(\alpha\boldsymbol{\beta})$$

The $i$th observation in group $j$ is drawn by first choosing a component $k_{ji}$ from the group-level proportion distribution $\boldsymbol{\pi}_j$, and then drawing the observation $y_{ji}$ from $\boldsymbol{\phi}_{k_{ji}}$:

$$k_{ji} \sim \text{Multinomial}(\boldsymbol{\pi}_j); \quad y_{ji} \sim \text{Multinomial}(\boldsymbol{\phi}_{k_{ji}})$$

We will use a variant of the HDP to analyze groups of songs. Our analysis will find:

1. The set of components used to generate those songs.

2. When and how prominently those components appear in each song.

In order to do this, we need to explicitly model how prominent each component is at any given time in each song.

The SIHDP extends the HDP by modeling each observed time $w_{ji}$ as a sum of two terms: a base time $c_{ji} \in \{1, \ldots, C\}$ and a discrete time offset $l_{ji} \in \{-C+1, \ldots, W-1\}$. We define $L = W + C - 1$ to be the size of this set of possible time offsets. We set $C$ to be the length of the latent components that we wish to model (which will be short relative to the song). The time offsets $l$ can take on any range of values such that there is some $c$ for which $l + c \in \{1, \ldots, W\}$.

As in the HDP, we begin by drawing a set of latent components $\boldsymbol{\phi}$ from a symmetric Dirichlet prior with parameter $\epsilon$, but $\boldsymbol{\phi}$ is now a two-dimensional joint distribution over base times $c$ and frequency bins $b$. Each $\boldsymbol{\phi}$ can be interpreted as a normalized spectrogram of a short audio source. The global component proportion vector $\boldsymbol{\beta}$ is again drawn from a stick-breaking process with concentration parameter $\gamma$, and the song-level component proportion vector $\boldsymbol{\pi}_j$ for each song $j$ is drawn from a DP with concentration parameter $\alpha$ and base distribution Multinomial($\boldsymbol{\beta}$).

Each component $k$ in song $j$ in the SIHDP has a set of multinomial distributions $\boldsymbol{\omega}_{jk}$ over time offsets drawn from a symmetric Dirichlet prior with parameter $\eta$.

Each observed quantum of energy $y_{ji}$ consists of a time $w_{ji}$ and a frequency bin $b_{ji}$ at which the quantum appears. To generate $y_{ji}$, we first select a component $k_{ji}$ to generate the quantum. We draw $k_{ji}$ from Multinomial($\boldsymbol{\pi}_j$), the song-level distribution over components.

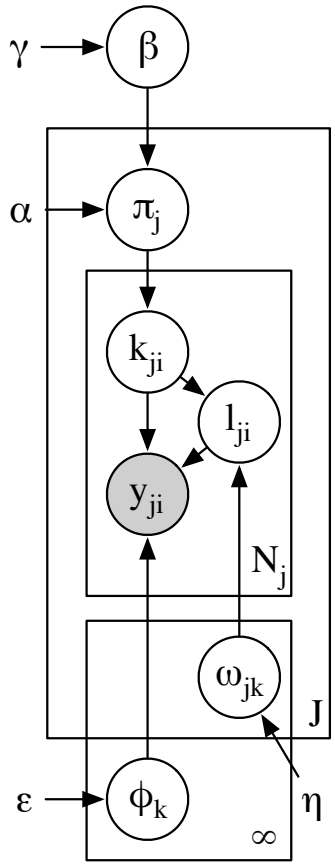


Figure 5.2: The graphical model for the shift-invariant HDP.

We then draw a base time $c_{ji}$ and frequency $b_{ji}$ jointly from $\boldsymbol{\phi}_{k_{ji}}$, and draw a time offset $l_{ji}$ from the distribution over time offsets $\boldsymbol{\omega}_{jk_{ji}}$ for component $k_{ji}$ in song $j$.

The observed quantum appears at time $w_{ji} = c_{ji} + l_{ji}$ and frequency $b_{ji}$.

The full generative process for the SIHDP is:

$$\begin{aligned}
\boldsymbol{\phi}_k &\sim \text{Dirichlet}(\epsilon, \ldots, \epsilon) & \boldsymbol{\omega}_{jk} &\sim \text{Dirichlet}(\eta, \ldots, \eta) \\
\boldsymbol{\beta} &\sim \text{GEM}(\gamma) & \boldsymbol{\pi}_j &\sim \text{Dirichlet}(\alpha\boldsymbol{\beta}) \\
k_{ji} &\sim \text{Multinomial}(\boldsymbol{\pi}_j) & l_{ji} &\sim \text{Multinomial}(\boldsymbol{\omega}_{jk_{ji}}) \\
c_{ji}, b_{ji} &\sim \text{Multinomial}(\boldsymbol{\phi}_{k_{ji}}) & w_{ji} &= c_{ji} + l_{ji} \\
y_{ji} &= \{w_{ji}, b_{ji}\}
\end{aligned} \tag{5.3}$$

The SIHDP is a hierarchical nonparametric Bayesian version of Shift-Invariant Probabilistic Latent Component Analysis (SI-PLCA) [81], which is a probabilistic version of convolutive nonnegative matrix factorization with KL-divergence loss function [79]. It improves on SI-PLCA by allowing components to be shared across multiple songs, and by automatically determining the number of latent components that are needed to explain the data.

This SIHDP is also related to the Transformed Dirichlet Process (TDP) in that draws from a countably infinite global set of mixture components undergo transformations to generate observations [82]. In the TDP, however, transformations are associated with observations indirectly through table assignments in the Chinese Restaurant Franchise (CRF). This means that the concentration paramater $\alpha$ influences both the group-level component proportions $\boldsymbol{\pi}$ and the number of transformations that a group of observations (such as a song or an image) can take on; a high $\alpha$ simultaneously makes each $\boldsymbol{\pi}_j$ less likely to diverge from the global component proportions $\boldsymbol{\beta}$ and makes a large number of transformations in each group $j$ more likely.

Our model takes a simpler approach, directly associating each observation $y_{ji}$ with a transformation $l_{ji}$ that depends only on the cluster assignment $k_{ji}$ and a group-level multinomial distribution over transformations $\boldsymbol{\omega}_{jk_{ji}}$. We do not need to use the CRF machinery of the HDP to generate transformations, since our set of transformations is discrete. We note that this decoupled approach can be generalized to continuous transformations by using a set of DP's for each group to discretize a space of continuous transformations. This may be worth exploring in other models.

Although this work is focused on the applications of the SIHDP to music, it could equally well be applied to any of the other application areas described in [81], such as images or video. Likewise, although we only discuss shift-invariance in time, analogous models can be constructed that are invariant to shifts in frequency at extra computational expense. A log-frequency representation such as the constant-Q transform would be a more appropriate input for such a model than the linear frequency spectrogram.

To learn the posterior distribution over the model parameters conditioned on the observed spectrograms, we adapt the direct assignment Gibbs sampler from [85]. This Gibbs sampler gives us a set of samples from the posterior distribution over a set of the variables in our model, which we then use to compute a Maximum A Posteriori (MAP) estimate of the remaining parameters. Full details of the inference procedure can be found in appendix A.

## 5.2 Evaluation

We conducted several experiments to test the SIHDP on music audio data—one using synthetic drum loops and three using songs taken from the CAL500 dataset, which consists of 500 songs of various genres of Western popular music each recorded by a different artist within the last 50 years [90]. In all experiments, we placed a gamma$(1, 0.0001)$ prior on both $\alpha$ and $\gamma$, and set $\epsilon = 0.02$ and $\eta = 0.01$.

### 5.2.1 Drum Loop Transcription

We synthesized a set of 40 randomly generated 32-beat drum loops lasting 6 seconds each. We studied the SIHDP's ability to discover the drum sounds that were used to create the files, and when in each file they appear. We used a simple algorithm to generate the loops: for each drum $s$ at each beat $i$ in song $j$, we draw a Bernoulli variable $r_{sij} \sim$ Bern$(p_s)$

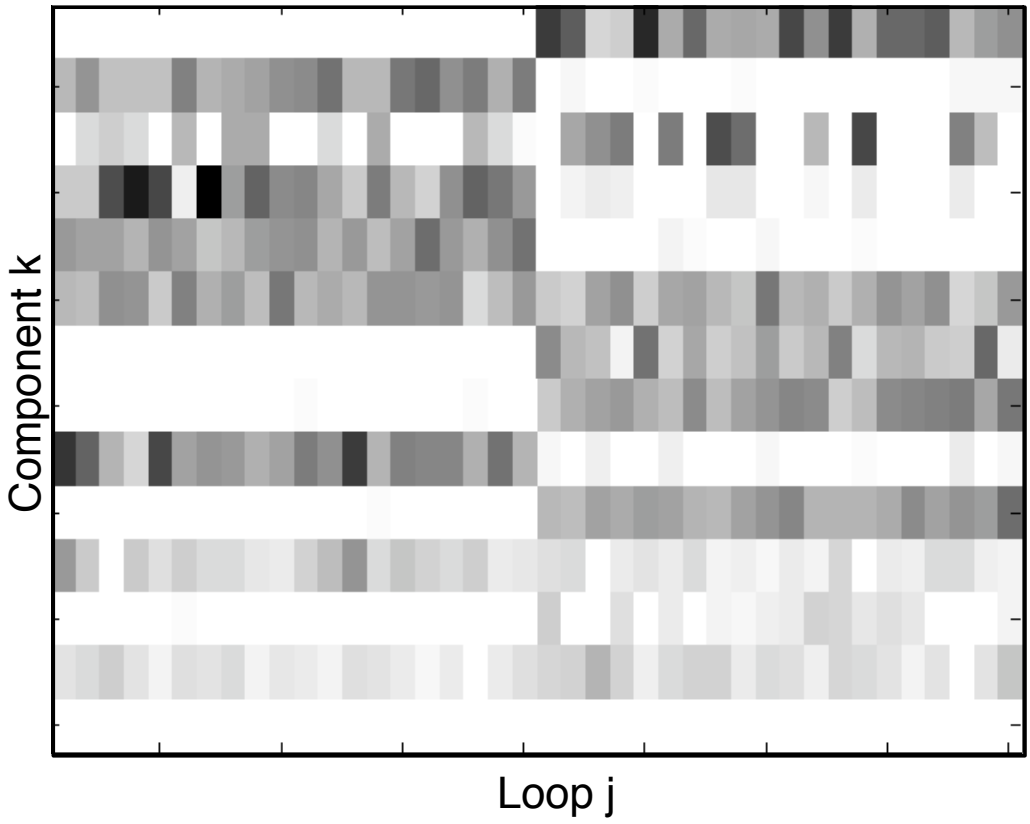


Figure 5.3: A graphical representation of our model's MAP estimate of $\boldsymbol{\pi}$ for the 40 synthetic drum loops. A darker pixel in row $k$ of column $j$ indicates a higher relative proportion $\pi_{jk}$ of latent component $k$ in song $j$.

that indicates whether or not drum $s$ is present at beat $i$. If it is, then we draw its amplitude $a_{sij} \sim \text{Unif}(0.3, 0.9)$, otherwise we set $a_{sij} = 0$. Audio was synthesized using the ChucK music programming language [96] and two sets of drum samples from Apple's GarageBand.

Our objective was to recover from the audio alone an estimate of $\boldsymbol{a}$ for each drum in each song, without any prior knowledge of what the drum samples sound like (aside from their maximum length), how many there are, or how frequently they appear.

We ran our Gibbs sampler until the posterior likelihood failed to increase for 20 iterations on the 40 synthesized files, choosing $C = 10$ and $\nu = 0.25$. We then computed a MAP estimate of the time offset distribution $\boldsymbol{\omega}|\boldsymbol{k}, \boldsymbol{l}$, and calculated a distribution $\boldsymbol{\omega}'$ quantized to 32 beats, so that $\omega'_{jki}$ is the probability that a quantum of energy generated by component $k$ in song $j$ will fall anywhere in beat $i$. $\hat{\omega}_{jki} = \pi_{jk}\omega'_{jki}$ is then the relative prominence of component $k$ at beat $i$ in loop $j$.

Figure 5.3 graphically represents the distribution $\boldsymbol{\pi}$ discovered by our model. Note that most of the latent components tend to appear in either the first 20 loops or the last 20, but not both. This is because the first 20 loops were generated using a different set of drum

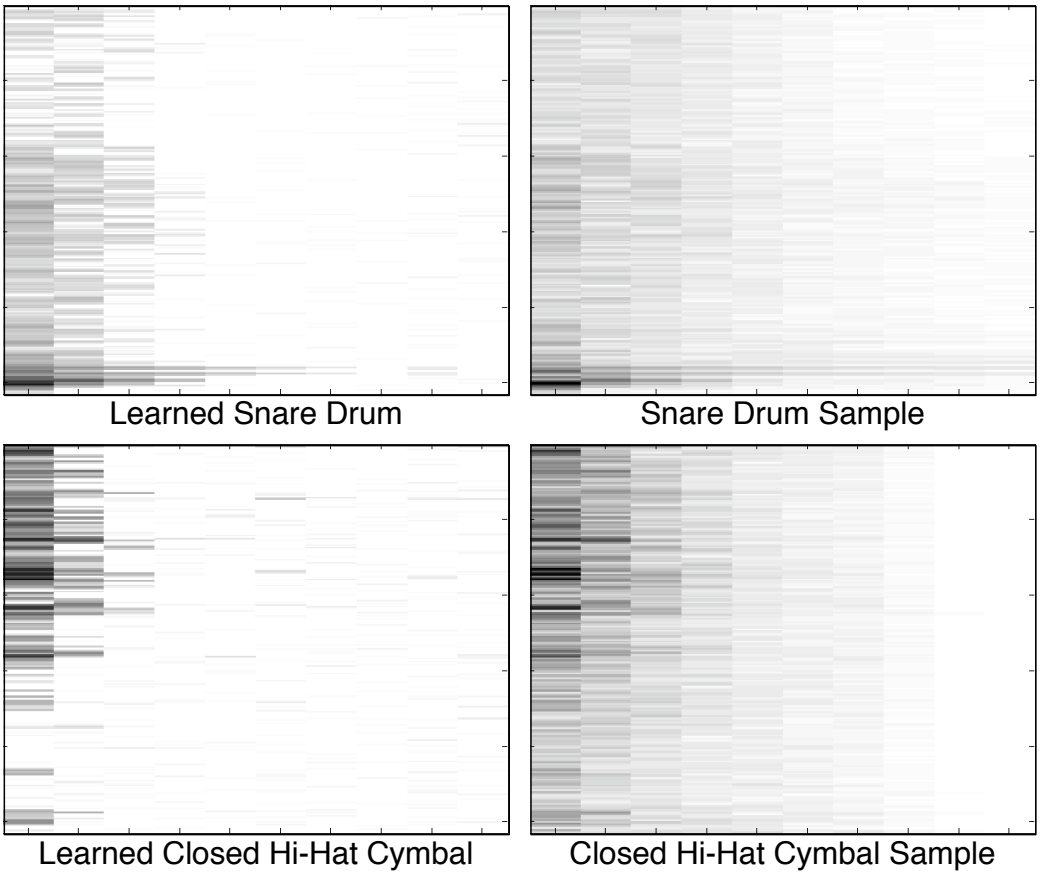


Figure 5.4: Left: Two latent distributions $\phi_k$ discovered by our model. Right: Spectrograms of two drum samples closely matching the latent components at left.

samples than the last 20, and our model was able to distinguish between the two synthetic drum kits.

We evaluated the Bhattacharyya distance for each song between the joint distribution over components and times defined by $\hat{\boldsymbol{\omega}}_j$ and the joint distribution over drums and times defined by our ground truth $\boldsymbol{a}_j$ (normalized so that it can be treated as a multinomial distribution). The Bhattacharyya distance between two probability distributions $p$ and $q$ over the same domain $\boldsymbol{X}$ is a symmetric measure of the dissimilarity of those two distributions, and is defined as

$$D_B(p, q) = -\log\left(\sum_{x \in \boldsymbol{X}} \sqrt{p(x)q(x)}\right) \tag{5.4}$$

We assume (naïvely) that the component $k$ in song $j$ that corresponds to drum $s$ is the one that maximizes $\text{Corr}(\hat{\boldsymbol{\omega}}_{jk}, \boldsymbol{a}_{js})$. The average Bhattacharyya distance between our transcription and the ground truth was 0.4236 with a standard error of 0.0204. The average Bhattacharyya distance obtained by repeating the experiment with a a normalized matrix of numbers drawn uniformly at random substituted for $\hat{\boldsymbol{\omega}}_j$ was 1.1923 with a standard error of 0.0131. The SIHDP did dramatically better than chance at transcribing the drum tracks.

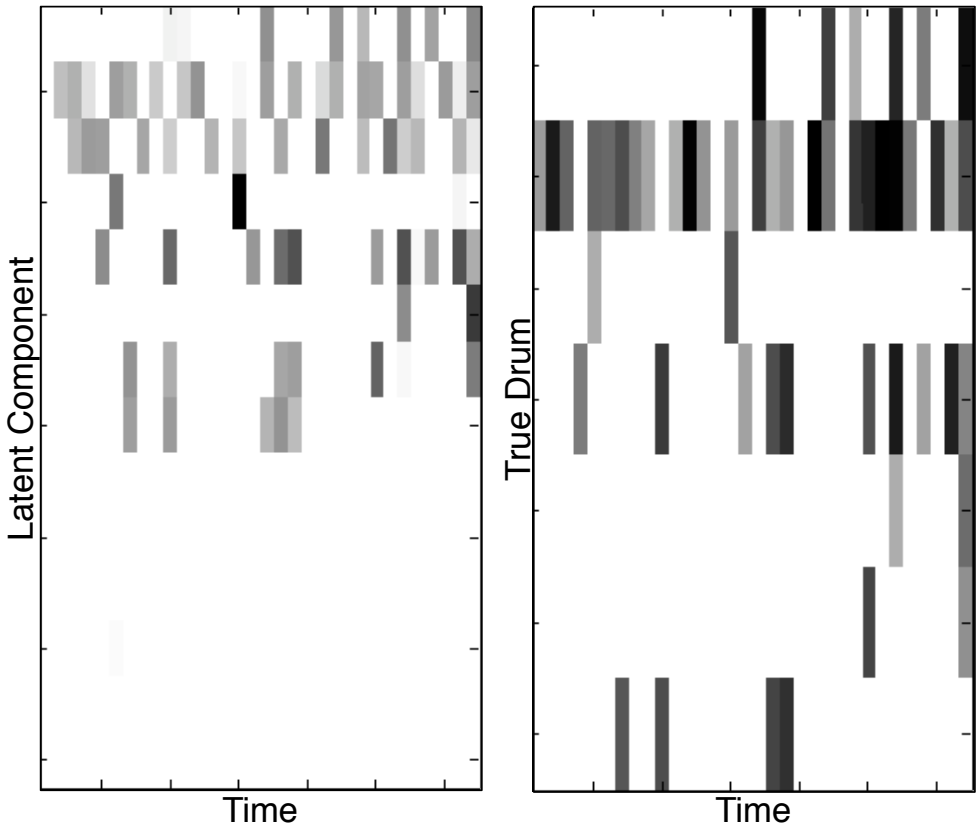


Figure 5.5: Left: An unsupervised transcription $\hat{\boldsymbol{\omega}}$ generated by our model of a drum loop. Right: The actual times and amplitudes of the drum loop. Darker pixels correspond to higher amplitudes.

Figure 5.4 compares the spectrograms of two of the drum samples used to generate our data with the discovered latent components they most closely match. Figure 5.5 compares the ground truth transcription $\boldsymbol{a}$ with the SIHDP's transcription $\hat{\boldsymbol{\omega}}$ for a single drum loop. The rows of $\hat{\boldsymbol{\omega}}$ have been manually ordered to make their correspondence to the rows of $\boldsymbol{a}$ clearer. The second drum seems to have been split across two components, but most of the drums have a clear one-to-one mapping to latent components, which confirms our quantitative results. The empty rows correspond to latent components not used to model this loop.

### 5.2.2 Experiments on Recorded Popular Music

We ran our distributed Gibbs sampler on a training set of 48 songs from the CAL500 dataset to get MAP estimates of the global component proportions $\boldsymbol{\beta}$, the global components $\boldsymbol{\phi}$, the song-level component proportions $\boldsymbol{\pi}$, and the song-level offset distributions $\boldsymbol{\omega}$. We set the length $C$ of the latent components to 20 windows. We used 2000 512-sample windows (46 seconds of audio) from each song with $\nu$ set to 1, for an average of 514,000 observations

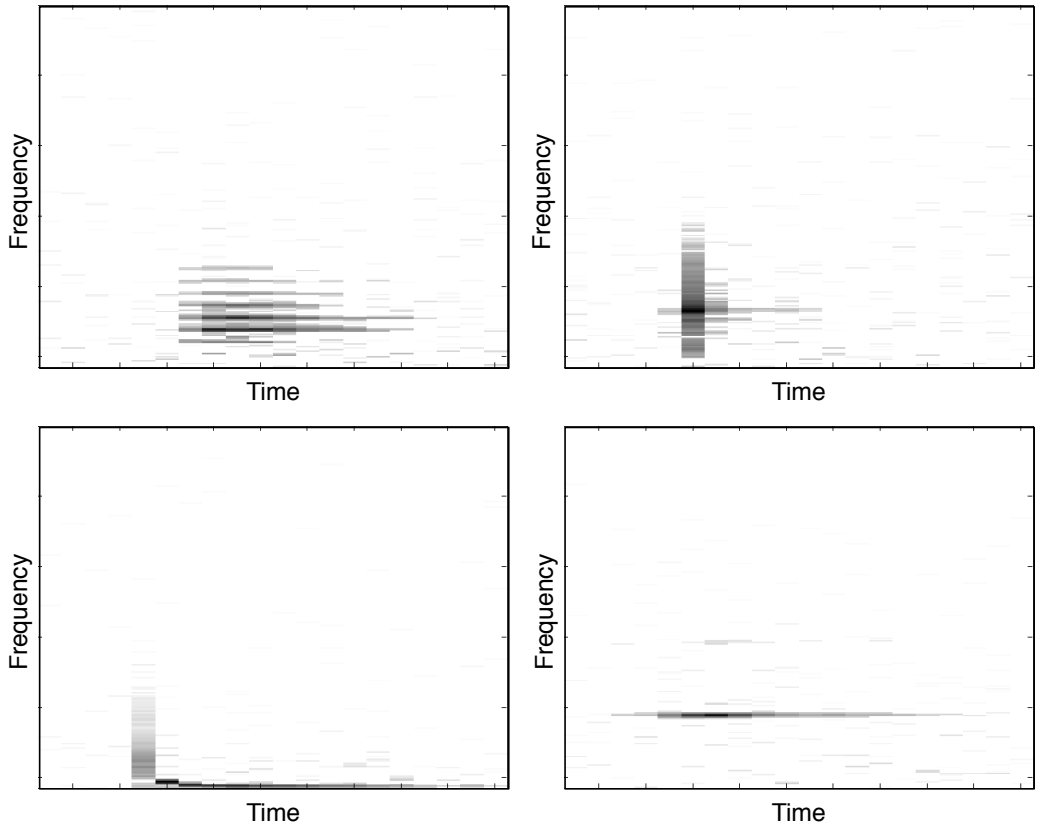


Figure 5.6: Four latent components discovered from 48 songs taken from the CAL500 corpus of popular music.

per song. The Gibbs sampler took about a day to converge running on 48 processors, and discovered 575 components.

Figure 5.6 shows several latent components discovered by the SIHDP from the 48 training songs. Qualitatively, these sound like (clockwise from bottom left) a bass drum, a male voice singing "aah," a snare drum, and a high-pitched whistle. While the first three components clearly correspond to real-world sound sources, it seems more likely that the fourth component is being used to model fine details of the data that are cannot be captured by the more complex components.

Figure 5.7 shows the intensities $\hat{\omega}_{jkl} = \omega_{jkl}\pi_{jk}$ with which the 10 most prominent components $k$ appear at each time offset $l$ in the song "Dirty Deeds Done Dirt Cheap" by AC/DC. Different, but related rhythmic patterns for each component are clearly visible. Exploiting the rhythmic information in this representation may prove valuable for music information retrieval tasks [23].

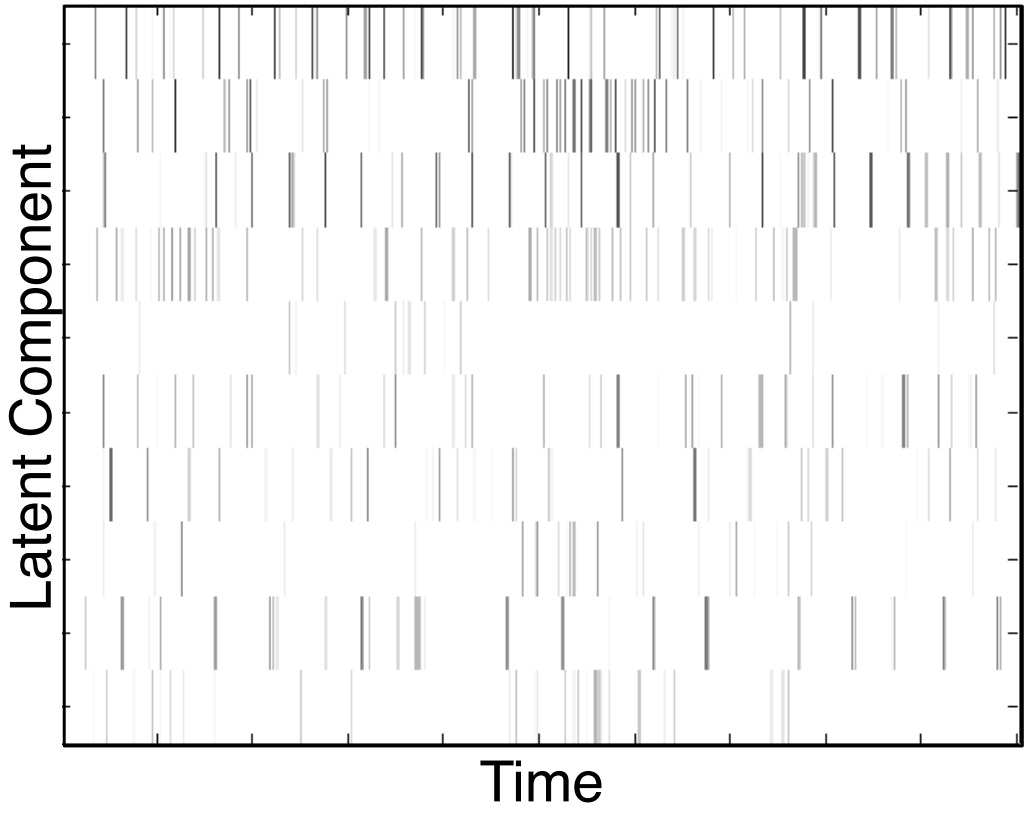


Figure 5.7: The 10 most prominent components of the unsupervised transcription $\hat{\boldsymbol{\omega}}$ inferred from 11 seconds of the AC/DC song "Dirty Deeds Done Dirt Cheap." Some components are relatively weak here, but become more prominent elsewhere in the song.

**Perplexity**

After obtaining MAP estimates of the global component proportions $\boldsymbol{\beta}$ and the global components $\boldsymbol{\phi}$, we ran our Gibbs sampler on 400 held-out songs from the same dataset holding the global component proportions $\boldsymbol{\beta}$ and the component distributions $\boldsymbol{\phi}$ fixed at the MAP estimates from the training set, and estimated the perplexity on the held-out data using the harmonic mean of the likelihoods of the data under samples from the posterior of the hidden parameters[1]. The perplexity of a model defining a probability distribution $p(\boldsymbol{x})$ on a held-out data set $\boldsymbol{x}$ is defined as

$$\text{perplexity}(x) \triangleq \exp\left\{-\frac{1}{N}\sum_{n}\log p(x_n)\right\}. \tag{5.5}$$

For comparison, we also built a simple DP model that also assumes an infinite set of latent components, but has each song choose a single latent component that it uses to generate

[1]While this is a widely used method (cf. e.g. [40]) there is some debate in the statistics community as to its effectiveness compared with alternative estimation methods such as [63].

every observed quantum. We estimated the perplexity of this model on the same held-out data. The DP's perplexity was 1265.2, and our SIHDP model's perplexity was 62.1. This dramatic reduction in perplexity illustrates the value of modeling songs as mixtures of latent components.

## 5.3 Discussion

In this chapter, we presented the Shift-Invariant Hierarchical Dirichlet Process (SIHDP), a model that can discover a rich representation of groups of songs in terms of the instruments and vocal sounds that generated those songs. We developed an exact parallel Gibbs sampler that enabled us to run experiments on a significant number of songs, and showed that the SIHDP can discover latent audio sources that are shared across multiple songs, as well as when those sources occur in each song. The ability of the SIHDP to automatically determine how many latent sources are needed to explain the data is a significant advance over previous methods, which typically assume that the number of latent sources is specified a priori.

# Chapter 6

# Nonparametric Latent Source Discovery Part II: Gamma Process Nonnegative Matrix Factorization

## 6.1 Introduction

Although the SI-HDP model discussed in the previous chapter does a good job of discovering the latent sources present in audio, it suffers from two related theoretical issues.

The first issue is that the quantization level determines the number of "observations" that the model is to explain. This is troubling from the perspective of Bayesian statistics, in that the variance of the posterior over the model parameters depends strongly on the number of observations, which is controlled by an arbitrary factor. Even more troubling is that the number of components that a Bayesian nonparametric model can justify associating with data depends on the number of observations given to the model—thus, the quality of model order selection done by the SI-HDP may depend on the level of quantization.

The second issue relates to the signal model implied by the SI-HDP. The SI-HDP assumes that the observed quantized spectrogram is generated by introducing multinomial noise to the convolution of the latent components by their activation functions, scaled by a set of overall gains. Although this noise model simplifies inference, there is no justification in signal processing for the assumption of multinomial noise in audio spectrogram data.

In this chapter we develop *Gamma Process Nonnegative Matrix Factorization* (GaP-NMF), an alternative Bayesian nonparametric (BNP) approach to decomposing spectrograms. As in the SI-HDP, we posit a generative probabilistic model of spectrogram data where, given an observed audio signal, posterior inference reveals both the latent sources and their number. Unlike the SI-HDP, GaP-NMF does not need to quantize real-valued spectrogram data, and it assumes a noise model that corresponds to a more reasonable model of how audio sources combine in the frequency domain to form power spectra.

The central computational challenge posed by our model is posterior inference. Unlike other BNP factorization methods, our model is not composed of conjugate pairs of distributions—we chose our distributions to be appropriate for spectrogram data, not for computational convenience.

We use variational inference to approximate the posterior, and develop a novel variational approach to inference in nonconjugate models. Variational inference approximates the posterior with a simpler distribution, whose parameters are optimized to be close to the true posterior [53]. In mean-field variational inference, each variable is given an independent distribution, usually of the same family as its prior. Where the model is conjugate, optimization proceeds by an elegant coordinate ascent algorithm. Researchers usually appeal to less efficient scalar optimization where conjugacy is absent. We instead use a *bigger* variational family than the model initially asserts. We show that this gives an analytic coordinate ascent algorithm, of the kind usually limited to conjugate models.

We evaluated GaP-NMF on several problems—extracting the sources from music audio, predicting the signal in missing entries of the spectrogram, and classical measures of Bayesian model fit. Our model performs as well as or better than the current state-of-the-art. It finds simpler representations of the data with equal statistical power, without needing to explore many fits over many numbers of sources, and thus with much less computation.

## 6.2 GaP-NMF Model

We model the Fourier power spectrogram $\boldsymbol{X}$ of an audio signal. The spectrogram $\boldsymbol{X}$ is an $M$ by $N$ matrix of non-negative reals; the cell $X_{mn}$ is the power of our input audio signal at time window $n$ and frequency bin $m$. Each column of the power spectrogram is obtained as follows. First, take the discrete Fourier transform of a window of $2(M-1)$ samples. Next, compute the squared magnitude of the complex value in each frequency bin. Finally, keep only the first $M$ bins, since the remaining bins contain only redundant information.

We assume the audio signal is composed of $K$ static sound sources. As a consequence, we can model the observed spectrogram $\boldsymbol{X}$ with the product of two non-negative matrices: an $M$ by $K$ matrix $\boldsymbol{W}$ describing these sources and a $K$ by $N$ matrix $\boldsymbol{H}$ controlling how the amplitude of each source changes over time [80]. Each column of $\boldsymbol{W}$ is the average power spectrum of an audio source; cell $W_{mk}$ is the average amount of energy source $k$ exhibits at frequency $m$. Each row of $\boldsymbol{H}$ is the time-varying gain of a source; cell $H_{kn}$ is the gain of source $k$ at time $n$. These matrices are unobserved. For simplicity of exposition, we will assume static sources unlike the time-varying sources of the SI-HDP. The modeling and inference techniques we develop for GaP-NMF can easily be extended to accommodate time-varying sources.

GaP-NMF builds on the signal model motivating the Itakura-Saito NMF algorithm developed in [4] and [32], which we briefly rederive here. Consider a set of $K$ audio signals whose complex spectra at each time $n$ are distributed according to

$$\text{Real}(C_{mnk}) \sim \mathcal{N}(0, W_{mk}); \quad \text{Imag}(C_{mnk}) \sim \mathcal{N}(0, W_{mk}), \tag{6.1}$$

where the variance $W_{mk}$ is the average power at frequency $m$ of source $k$ and $C_{mnk}$ is the complex Fourier coefficient of the audio signal generated by source $k$ at frequency $m$ at time $n$. Linearly combining these $K$ signals in the time domain, weighted by their time

varying gains $H_{kn}$, yields a complex mixture spectrogram $\boldsymbol{X}^c$:

$$\text{Real}(X^c_{mn}) = \sum_k \text{Real}(C_{mnk})\sqrt{H_{kn}}; \quad \text{Imag}(X^c_{mn}) = \sum_k \text{Imag}(C_{mnk})\sqrt{H_{kn}}; \quad (6.2)$$

basic properties of the normal distribution then imply that

$$\text{Real}(X^c_{mn}) \sim \mathcal{N}(0, \sum_k W_{mk}H_{kn}); \quad \text{Real}(X^c_{mn}) \sim \mathcal{N}(0, \sum_k W_{mk}H_{kn}). \quad (6.3)$$

It is then straightforward to show that the elements of the power spectrogram $\boldsymbol{X}$ are distributed according to an exponential distribution:

$$X_{mn} \triangleq \text{Real}(X^c_{mn})^2 + \text{Imag}(X^c_{mn})^2 \sim \text{Exponential}\left(\sum_k W_{mk}H_{kn}\right). \quad (6.4)$$

Previous spectrogram decompositions assume the number of components $K$ is known. In practice, this is rarely true. Our goal is to develop a method that infers both the characters and number of latent audio sources from data. We develop a Bayesian nonparametric model with an infinite number of latent components, a finite number of which are active when conditioned on observed data.

We now describe the Gamma Process Nonnegative Matrix Factorization model (GaP-NMF). As in previous matrix decomposition models, the spectrogram $\boldsymbol{X}$ arises from hidden matrices $\boldsymbol{W}$ and $\boldsymbol{H}$. In addition, the model includes a hidden vector of non-negative values $\boldsymbol{\theta}$, where each element $\theta_l$ is the overall gain of the corresponding source $l$. The key idea is that we allow for the *possibility* of a large number of sources $L$, but place a sparse prior on $\boldsymbol{\theta}$. During posterior inference, this prior biases the model to use no more sources than it needs.

Specifically, GaP-NMF assumes that $\boldsymbol{X}$ is drawn according to the following generative process:

$$\begin{aligned} W_{ml} &\sim \text{Gamma}(a, a) \\ H_{ln} &\sim \text{Gamma}(b, b) \\ \theta_l &\sim \text{Gamma}(\alpha/L, \alpha c) \\ X_{mn} &\sim \text{Exponential}(\textstyle\sum_l \theta_l W_{ml} H_{ln}). \end{aligned} \quad (6.5)$$

As the truncation level $L$ increases towards infinity, the vector $\boldsymbol{\theta}$ approximates an infinite sequence drawn from a gamma process with shape parameter $\alpha$ and inverse-scale parameter $\alpha c$ [55]. A property of this sequence is that the number of elements $K$ greater than some number $\epsilon > 0$ is finite almost surely. Specifically:

$$K \sim \text{Poisson}\left(\frac{1}{c}\int_\epsilon^\infty x^{-1}e^{-x\alpha c}dx\right). \quad (6.6)$$

For truncation levels $L$ that are sufficiently large relative to the shape parameter $\alpha$, we likewise expect that only a few of the $L$ elements of $\boldsymbol{\theta}$ will be substantially greater than 0.

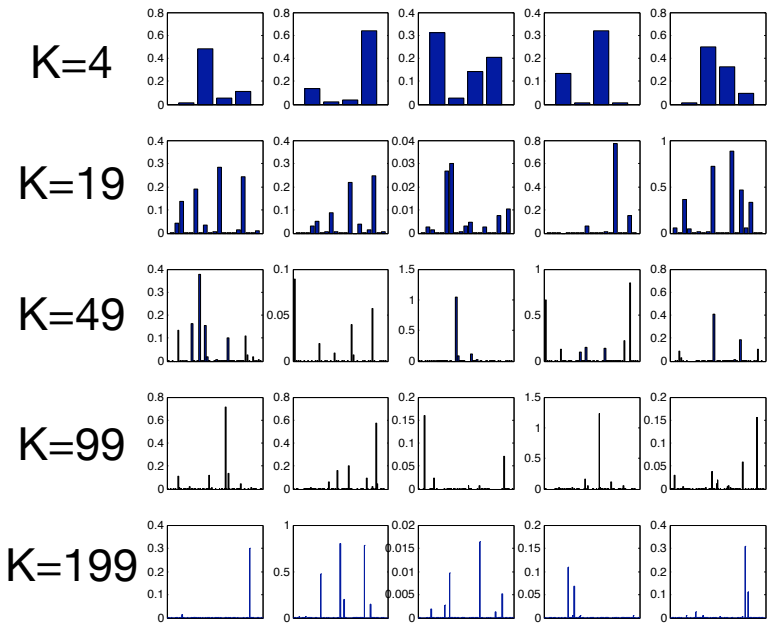


Figure 6.1: Five vectors whose entries are drawn independently from a Gamma$(2/K, 2)$ distribution for various dimensionalities $K$.

Figure 6.1 illustrates this sparseness phenomenon. During posterior inference, this property leads to a preference for explanations that use relatively few components.

The resemblance between figure 6.1 and figure 2.2 is not a coincidence. The Dirichlet and gamma processes are closely related, as are the Dirichlet and gamma distributions. In general, if a set of variables are distributed according to a set of gamma distributions with constant scale parameter, then normalizing these variables to sum to one gives a Dirichlet distribution. Specifically, if $\theta_k \sim \text{Gamma}(\alpha_k, c)$ and $\pi_k \triangleq \theta_k / \sum_i \theta_i$ then $\boldsymbol{\pi} \sim \text{Dirichlet}(\boldsymbol{\alpha})$. Similarly, a gamma process normalized so that the weights of its atoms sum to 1 is a Dirichlet process. The Dirichlet process was in fact originally derived as a normalized version of the gamma process [31].

Note that the expected value of $X_{mn}$ under the GaP-NMF model is constant with respect to $L$, $\alpha$, $a$, and $b$:

$$\mathbb{E}_p[X_{mn}] = \textstyle\sum_l \mathbb{E}_p[\theta_l]\mathbb{E}_p[W_{ml}]\mathbb{E}_p[H_{ln}] = \frac{1}{c}. \tag{6.7}$$

This equation suggests the heuristic of setting the expected mean of the spectrogram $\boldsymbol{X}$ under the prior equal to its empirical mean $\bar{X}$ by setting $c = 1/\bar{X}$.

## 6.3 Variational Inference

Posterior inference is the central computational problem for analyzing data with the GaP-NMF model. Given an observed spectrogram $\boldsymbol{X}$, we want to compute the posterior dis-

tribution $p(\boldsymbol{\theta}, \boldsymbol{W}, \boldsymbol{H}|\boldsymbol{X}, \alpha, a, b, c)$. Exact Bayesian inference is intractable. We appeal to mean-field variational inference [53].

Variational inference is a deterministic alternative to Markov Chain Monte Carlo (MCMC) methods that replaces sampling with optimization. It has permitted efficient large-scale inference for several Bayesian nonparametric models [e.g. 16, 26, 70]. Variational inference algorithms approximate the true posterior distribution with a simpler variational distribution controlled by free parameters. These parameters are optimized to make the variational distribution close (in Kullback-Leibler divergence) to the true posterior of interest. Mean-field variational inference uses a fully factorized variational distribution—i.e., under the variational distribution all variables are independent. In conjugate models this permits easy coordinate ascent updates using variational distributions of the same families as the prior distributions.

Less frequently, variational methods are applied to non-conjugate models, which allow increased model expressivity at the price of greater algorithmic challenges. Our model is such a model. The usual strategy is to use a factorized variational distribution with the same families as the priors, bound or approximate the objective function, and use numerical techniques to optimize difficult parameters [17, 20].

We use a different strategy. We adopt an expanded family for our variational distributions, one that generalizes the priors' family. This allows us to derive analytic coordinate ascent updates for the variational parameters, eliminating the need for numerical optimization.

### 6.3.1 Variational Objective Function

It is standard in mean-field variational inference to give each variable a variational distribution from the same family as its prior distribution [53]. We instead use the more flexible Generalized Inverse-Gaussian (GIG) family [54]:

$$\begin{aligned} q(W_{ml}) &= \text{GIG}(\gamma_{ml}^{(W)}, \rho_{ml}^{(W)}, \tau_{ml}^{(W)}) \\ q(H_{ln}) &= \text{GIG}(\gamma_{ln}^{(H)}, \rho_{ln}^{(H)}, \tau_{ln}^{(H)}) \\ q(\theta_l) &= \text{GIG}(\gamma_{l}^{(\theta)}, \rho_{l}^{(\theta)}, \tau_{l}^{(\theta)}). \end{aligned} \tag{6.8}$$

The GIG distribution is an exponential family distribution with sufficient statistics $x$, $1/x$, and $\log x$, and its PDF (in canonical exponential family form) is

$$\text{GIG}(y; \gamma, \rho, \tau) = \frac{\exp\{(\gamma - 1)\log y - \rho y - \tau/y\}\rho^{\gamma/2}}{2\tau^{\gamma/2}\mathcal{K}_\gamma(2\sqrt{\rho\tau})}, \tag{6.9}$$

for $x \geq 0$, $\rho \geq 0$, and $\tau \geq 0$. ($\mathcal{K}_\nu(x)$ denotes a modified Bessel function of the second kind.)

Note that the GIG family's sufficient statistics ($y$, $1/y$, and $\log y$) are a superset of those of the gamma family ($y$ and $\log y$), and so the gamma family is a special case of the GIG family where $\gamma > 0, \tau \to 0$.

To compute the bound in equation 6.11, we will need the expected values of each $W_{ml}$, $H_{ln}$, and $\theta_l$ and of their reciprocals under our variational GIG distributions. For a variable $y \sim \text{GIG}(\gamma, \rho, \tau)$ these expectations are

$$\mathbb{E}[y] = \frac{\mathcal{K}_{\gamma+1}(2\sqrt{\rho\tau})\sqrt{\tau}}{\mathcal{K}_{\gamma}(2\sqrt{\rho\tau})\sqrt{\rho}}; \quad \mathbb{E}\left[\frac{1}{y}\right] = \frac{\mathcal{K}_{\gamma-1}(2\sqrt{\rho\tau})\sqrt{\rho}}{\mathcal{K}_{\gamma}(2\sqrt{\rho\tau})\sqrt{\tau}}. \tag{6.10}$$

Having chosen a fully factorized variational family, we can lower bound the marginal likelihood of the input spectrogram under the GaP-NMF model [53]:

$$\begin{aligned} \log p(\boldsymbol{X}|\alpha, a, b, c) \geq\ & \mathbb{E}_q[\log p(\boldsymbol{X}|\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{\theta})] \\ & + \mathbb{E}_q[\log p(\boldsymbol{W}|a)] - \mathbb{E}_q[\log q(\boldsymbol{W})] \\ & + \mathbb{E}_q[\log p(\boldsymbol{H}|b)] - \mathbb{E}_q[\log q(\boldsymbol{H})] \\ & + \mathbb{E}_q[\log p(\boldsymbol{\theta}|\alpha, c)] - \mathbb{E}_q[\log q(\boldsymbol{\theta})]. \end{aligned} \tag{6.11}$$

The difference between the left and right sides of equation 6.11 is the Kullback-Leibler (KL) divergence between the true posterior and the variational distribution $q$. Thus, maximizing this bound with respect to $q$ minimizes the KL divergence between $q$ and our posterior distribution of interest.

The second, third, and fourth lines of equation 6.11 can be computed using the expectations in equation 6.10.

The likelihood term in equation 6.11 expands to

$$\mathbb{E}_q[\log p(\boldsymbol{X}|\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{\theta})] \ = \ \sum_{m,n} \mathbb{E}_q\left[\frac{-X_{mn}}{\sum_l \theta_l W_{ml} H_{ln}}\right] - \mathbb{E}_q\left[\log \sum_l \theta_l W_{ml} H_{ln}\right]. \tag{6.12}$$

We cannot compute either of the expectations on the right. However, we can compute lower bounds on both of them.

First, the function $-x^{-1}$ is concave. Jensen's inequality says that for any vector $\boldsymbol{\phi}$ such that $\phi_l \geq 0$ and $\sum_l \phi_l = 1$

$$-\frac{1}{\sum_l x_l} = -\frac{1}{\sum_l \phi_l \frac{x_l}{\phi_l}} \geq -\sum_l \phi_l \frac{1}{\frac{x_l}{\phi_l}} = -\sum_l \phi_l^2 \frac{1}{x_l}. \tag{6.13}$$

We use this inequality to derive a bound on the first expectation in equation 6.12:

$$\mathbb{E}_q\left[\frac{-X_{mn}}{\sum_l \theta_l W_{ml} H_{ln}}\right] \geq \sum_l \phi_{lmn}^2 \mathbb{E}_q\left[\frac{-X_{mn}}{\theta_l W_{ml} H_{ln}}\right] \tag{6.14}$$

Second, the function $-\log x$ is convex. We can therefore bound the second expectation in equation 6.12 using a first-order Taylor approximation about an arbitrary (positive) point

$\omega_{mn}$ as in [17] [1]:

$$-\mathbb{E}_q\left[\log \sum_l \theta_l W_{ml} H_{ln}\right] \geq -\log(\omega_{mn}) + 1 - \frac{1}{\omega_{mn}} \sum_l \mathbb{E}_q\left[\theta_l W_{ml} H_{ln}\right]. \tag{6.15}$$

We use equations 6.14 and 6.15 to bound equation 6.12:

$$\mathbb{E}_q[\log p(\boldsymbol{X}|\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{\theta})] \geq \sum_{m,n} -X_{mn} \sum_l \phi_{lmn}^2 \mathbb{E}_q\left[\frac{1}{\theta_l W_{ml} H_{ln}}\right] - \log(\omega_{mn}) + 1 - \frac{1}{\omega_{mn}} \sum_l \mathbb{E}_q\left[\theta_l W_{ml} H_{ln}\right]. \tag{6.16}$$

Note that this bound involves the expectations both of the model parameters *and* of their reciprocals under the variational distribution $q$. Since both $y$ and $1/y$ are sufficient statistics of GIG$(y; \gamma, \rho, \tau)$, this will not pose a problem during inference, as it would if we were to use variational distributions from the gamma family.

We denote as $\mathcal{L}$ the sum of the likelihood bound in equation 6.16 and the second, third, and fourth lines of equation 6.11. $\mathcal{L}$ lower bounds the likelihood $p(\boldsymbol{X}|\alpha, a, b, c)$. Our variational inference algorithm maximizes this bound over the free parameters, yielding an approximation $q(\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{\theta})$ to the true posterior $p(\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{\theta}|\boldsymbol{X}, \alpha, a, b, c)$.

### 6.3.2 Coordinate Ascent Optimization

We maximize the bound $\mathcal{L}$ using coordinate ascent, iteratively optimizing each parameter while holding all other parameters fixed. There are two sets of parameters to optimize: those used to bound the likelihood term in equation 6.12 and those that control the variational distribution $q$.

#### Tightening the likelihood bound

In equations 6.14 and 6.15, we derived bounds on the intractable expectations in equation 6.12. After updating the variational distributions on each set of parameters $\boldsymbol{W}$, $\boldsymbol{H}$, and $\boldsymbol{\theta}$, we update $\boldsymbol{\phi}$ and $\boldsymbol{\omega}$ to re-tighten these bounds.

Using Lagrange multipliers, we find that the optimal $\boldsymbol{\phi}$ is

$$\phi_{lmn} \propto \mathbb{E}_q\left[\frac{1}{\theta_l W_{ml} H_{ln}}\right]^{-1}. \tag{6.17}$$

The bound in equation 6.15 is tightest when

$$\omega_{mn} = \sum_l \mathbb{E}_q\left[\theta_l W_{ml} H_{ln}\right]. \tag{6.18}$$

[1] [20] observe that this bound is maximized when the Taylor approximation is taken around the expected value of the argument to the logarithm function, which corresponds to the 0th-order delta method. However, retaining the "redundant" parameter $\omega_{mn}$ permits faster and simpler updates for our other parameters.

I.e., this bound is tightest when we take the Taylor approximation about the expected value of the function's argument.

**Optimizing the variational distributions**

The derivative of $\mathcal{L}$ with respect to any of $\gamma_{ml}^{(W)}$, $\rho_{ml}^{(W)}$, or $\tau_{ml}^{(W)}$ equals 0 when

$$\gamma_{ml}^{(W)} = a; \quad \rho_{ml}^{(W)} = a + \mathbb{E}_q[\theta_l] \sum_n \frac{\mathbb{E}_q[H_{ln}]}{\omega_{mn}};$$
$$\tau_{ml}^{(W)} = \mathbb{E}_q\left[\frac{1}{\theta_l}\right] \sum_n X_{mn}\phi_{lmn}^2 \mathbb{E}_q\left[\frac{1}{H_{ln}}\right]. \tag{6.19}$$

Simultaneously updating the parameters $\boldsymbol{\gamma}^{(\boldsymbol{W})}$, $\boldsymbol{\rho}^{(\boldsymbol{W})}$, and $\boldsymbol{\tau}^{(\boldsymbol{W})}$ according to equation 6.19 will maximize $\mathcal{L}$ with respect to those parameters.

Similarly, the derivative of $\mathcal{L}$ with respect to any of $\gamma_{ln}^{(H)}$, $\rho_{ln}^{(H)}$, or $\tau_{ln}^{(H)}$ equals 0 and $\mathcal{L}$ is maximized when

$$\gamma_{ln}^{(H)} = b; \quad \rho_{ln}^{(H)} = b + \mathbb{E}_q[\theta_l] \sum_m \frac{\mathbb{E}_q[W_{ml}]}{\omega_{mn}};$$
$$\tau_{ln}^{(H)} = \mathbb{E}_q\left[\frac{1}{\theta_l}\right] \sum_m X_{mn}\phi_{lmn}^2 \mathbb{E}_q\left[\frac{1}{W_{ml}}\right]. \tag{6.20}$$

Finally, the derivative of $\mathcal{L}$ with respect to any of $\gamma_l^{(\theta)}$, $\rho_l^{(\theta)}$, or $\tau_l^{(\theta)}$ equals 0 and $\mathcal{L}$ is maximized when

$$\gamma_l^{(\theta)} = \frac{\alpha}{L}; \quad \rho_l^{(\theta)} = \alpha c + \sum_m \sum_n \frac{\mathbb{E}_q[W_{ml}H_{ln}]}{\omega_{mn}};$$
$$\tau_l^{(\theta)} = \sum_m \sum_n X_{mn}\phi_{lmn}^2 \mathbb{E}_q\left[\frac{1}{W_{ml}H_{ln}}\right]. \tag{6.21}$$

We iterate between updating bound parameters and variational parameters according to equations 6.17, 6.18, 6.19, 6.20, and 6.21. Each update tightens the variational bound on $\log p(\boldsymbol{X}|\alpha, a, b, c)$, ultimately reaching a local optimum.

### 6.3.3 Accelerating Inference

[70] observed that if $\mathbb{E}_q[\theta_l]$ becomes small for some component $l$, then we can safely skip the updates for the variational parameters associated with that component. (In our experiments we used 60 dB below $\sum_l \mathbb{E}_q[\theta_l]$ as a threshold.) This heuristic allows the use of large truncation levels $L$ (yielding a better approximation to an infinite gamma process) without incurring too severe a performance penalty. The first few iterations will be expensive, but the algorithm will require less time per iteration as it becomes clear that only a small number of components (relative to $L$) are needed to explain the data.

## 6.4 Evaluation

We conducted several experiments to assess the decompositions provided by the GaP-NMF model. We tested GaP-NMF's ability to recover the true parameters used to generate a synthetic spectrogram, compared the marginal likelihoods of real songs under GaP-NMF to the marginal likelihoods of those songs under a simpler version of the model, evaluated GaP-NMF's ability to predict held-out data with a bandwidth expansion task, and evaluated GaP-NMF's ability to separate individual notes from mixed recordings.

We compared GaP-NMF to two variations on the same model:

**Finite Bayesian model.** This is a finite version of the GaP-NMF model fit using the same variational updates but without the top-level gain parameters $\boldsymbol{\theta}$. This simpler model's generative process is

$$\begin{aligned} W_{mk} \sim \text{Gamma}(a, ac); \quad H_{kn} \sim \text{Gamma}(b, b); \\ X_{mn} \sim \text{Exponential}(\textstyle\sum_k W_{mk} H_{kn}), \end{aligned} \tag{6.22}$$

where $k \in \{1, \ldots, K\}$ and the model order $K$ is chosen a priori. The hyperparameters $a$, $b$, and $c$ are set to the same values as in the GaP-NMF model in all experiments. We will refer to this model as GIG-NMF, for Generalized Inverse-Gaussian Nonnegative Matrix Factorization.

**Finite non-Bayesian model.** This model fits $\boldsymbol{W}$ and $\boldsymbol{H}$ to maximize the likelihood in equation 8.9. [32] derive iterative multiplicative updates to maximize this likelihood, calling the resulting algorithm Itakura-Saito Nonnegative Matrix Factorization (IS-NMF).

We also compared GaP-NMF to the two nonnegative matrix factorization (NMF) algorithms described by [58]. Both of these algorithms also attempt to approximately decompose the spectrogram $\boldsymbol{X}$ into an $M$ by $K$ matrix $\boldsymbol{W}$ and a $K$ by $N$ matrix $\boldsymbol{H}$ so that $\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}$. The first algorithm, which we refer to as EU-NMF, minimizes the sum of the squared Euclidean distances between the elements of $\boldsymbol{X}$ and $\boldsymbol{W}\boldsymbol{H}$. The second algorithm, which we refer to as KL-NMF, minimizes the generalized KL-divergence between $\boldsymbol{X}$ and $\boldsymbol{W}\boldsymbol{H}$. KL-NMF (and its extensions) in particular is widely used to analyze audio spectrograms [e.g. 80, 9].

We focus on approaches that explain power spectrograms in terms of components that can be interpreted as audio power spectra. Other approaches may be useful for some tasks, but they do not decompose mixed audio signals into their component sources. This requirement excludes, for example, standard linear Gaussian factor models, whose latent factors cannot be interpreted as audio spectra unless audio signals are allowed to have negative power.

We normalized all spectrograms to have a maximum value of 1.0. (The high probability densities in our experiments result from low-power bins in the spectrograms.) To avoid numerical issues, we forced the values of the spectrograms to be at least $10^{-8}$, 80 dB below the peak value of 1.0.

In all experiments, we initialized the variational parameters $\boldsymbol{\rho}$ for each $W$, $H$, and $\theta$ with random draws from a gamma distribution with shape parameter 100 and inverse-scale parameter 1000, the variational parameters $\boldsymbol{\tau}$ to 0.1, and each $\gamma^{(W)}_{mk} = a$, $\gamma^{(H)}_{kn} = b$, and $\gamma^{(\theta)}_{k} = \alpha/K$. This yields a diffuse and smooth initial variational posterior, which helped

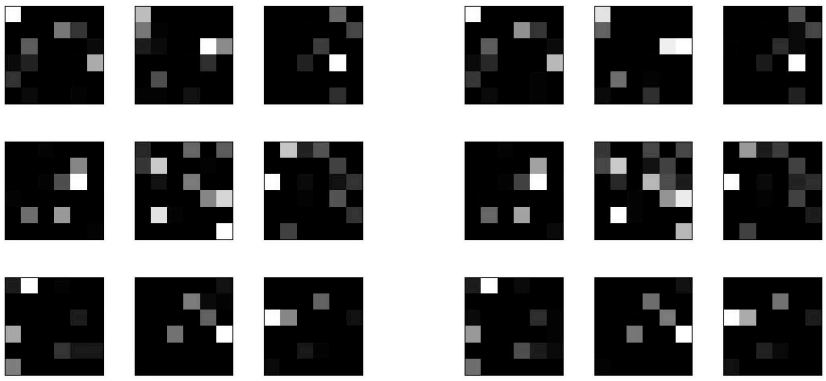

Figure 6.2: True synthetic bases (left) and expected values under the variational posterior of the nine bases found by the model (right). Brighter denotes more active. The 36-dimensional basis vectors are presented in $6 \times 6$ blocks for visual clarity.

avoid local optima. We ran variational inference until the variational bound increased by less than 0.001%. The GIG-NMF and IS-NMF algorithms were optimized to the same criterion. KL-NMF and EU-NMF were iterated until their cost functions decreased by less than 0.01 and 0.001, respectively. (We found no gains in performance from letting EU-NMF or KL-NMF run longer.) All algorithms were implemented in MATLAB[2].

We found GaP-NMF to be insensitive to the choice of $\alpha$, and so we set $\alpha = 1$ in all reported experiments.

### 6.4.1 Synthetic Data

We evaluated the GaP-NMF model's ability to correctly discover the latent bases that generated a matrix $\boldsymbol{X}$, and how many such bases exist. To test this, we fit GaP-NMF to random matrices $\boldsymbol{X}$ drawn according to the process:

$$\begin{aligned} W_{mk} &\sim \text{Gamma}(0.1, 0.1); \\ H_{kn} &\sim \text{Gamma}(0.1, 0.1); \\ X_{mn} &\sim \text{Exponential}(\textstyle\sum_k W_{mk} H_{kn}), \end{aligned} \tag{6.23}$$

where $m \in \{1, \ldots, M = 36\}$, $n \in \{1, \ldots, N = 300\}$, $k \in \{1, \ldots, K\}$ for $K = 9$.

We ran variational inference with the truncation level $L$ set to 50, and hyperparameters $\alpha = 1$, $a = b = 0.1$, $c = 1/\bar{X}$ (where $\bar{X}$ is the mean of $\boldsymbol{X}$). After convergence, only nine of these components were associated with the observed data. (The smallest element of $\theta$ associated with one of these nine components was 0.06, while the next largest element was $2.4 \times 10^{-8}$). Figure 6.2 shows that the latent components discovered by the model correspond closely to those used to generate the data.

[2]MATLAB code for GaP-NMF is available at `http://www.cs.princeton.edu/~mdhoffma`.

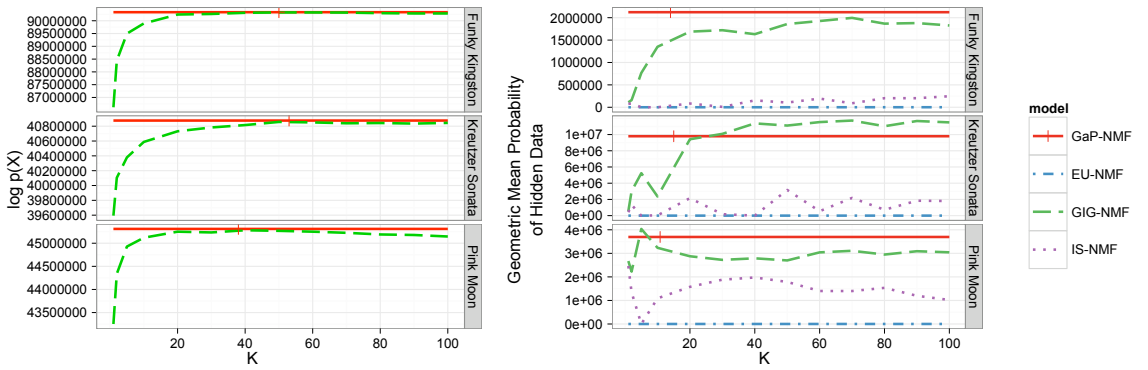


Figure 6.3: Left: Bounds on $\log p(\boldsymbol{X}|\text{prior})$ for the nonparametric GaP-NMF model and its parametric counterpart GIG-NMF with different numbers of latent components $K$. Ticks on the horizontal lines showing the bound for the GaP-NMF model indicate the number of components $K$ used to explain the data. For all three songs the values of $K$ chosen by GaP-NMF are close to the optimal value of $K$ for the parametric model. Right: Geometric mean of the likelihood assigned to each censored observation by the nonparametric, finite, and unregularized models. Ticks again indicate the number of components $K$ used to explain the data. The unregularized models overfit. EU-NMF performs badly, with likelihoods orders of magnitude lower than the other models.

### 6.4.2 Marginal Likelihood

We want to evaluate the ability of GaP-NMF to choose a good number of components to model recorded music. To determine a "good" number of components, we use variational inference to fit GIG-NMF with various orders $K$ and examine the resulting variational bounds on the marginal log-likelihood $\log p(\boldsymbol{X}|a, b, c)$.

As above, we set the prior parameters for the GaP-NMF model to $\alpha = 1$, $a = b = 0.1$, and $c = 1/\bar{X}$. We set the prior parameters for the simplified model to $a = b = 0.1$ and $c = 1/\bar{X}$. The value of $0.1$ for $a$ and $b$ was chosen because it gave slightly better bounds than higher or lower values. The results were not very sensitive to $\alpha$.

We computed power spectrograms from three songs: *Pink Moon* by Nick Drake, *Funky Kingston* by Toots and the Maytals, and a clip from the *Kreutzer Sonata* by Ludwig van Beethoven. These analyses used 2048-sample (46 ms) Hann windows with no overlap, yielding spectrograms of 1025 frequency bins by 2731, 6322, and 2584 time windows, respectively. We fit variational posteriors for GaP-NMF and GIG-NMF, conditioning on these spectrograms. We used a truncation level $L$ of 100 for the nonparametric model, and values of $K$ ranging from 1 to 100 for the finite GIG-NMF model.

The computational cost of fitting the GaP-NMF model was lower than the cost of fitting GIG-NMF with $K = 100$ (thanks to the accelerated inference trick in section 6.3.3), and much lower than the cost of repeatedly fitting GIG-NMF with different values of $K$. For example, on a single core of a 2.3 GHz AMD Opteron 2356 Quad-Core Processor, fitting the 100-component GIG-NMF model to *Pink Moon* took 857 seconds, while fitting the GaP-NMF model to the same song took 645 seconds.

The results are summarized in figure 6.3 (left). The GaP-NMF model used 50, 53, and 38 components to explain the spectrograms of *Funky Kingston*, the *Kreutzer Sonata*, and *Pink Moon* respectively. In each case the value of $K$ chosen by GaP-NMF was close to the best value of $K$ tested for the GIG-NMF model. This suggests that GaP-NMF performs automatic order selection as well as the more expensive approach of fitting multiple finite-order models.

### 6.4.3 Bandwidth Expansion

One application of statistical spectral analysis is bandwidth expansion, the problem of inferring what the high-frequency content of a signal is likely to be given only the low-frequency content of the signal [9]. This task has applications to restoration of low-bandwidth audio and lossy audio compression. This is a missing data problem. We compared the ability of different models and inference algorithms to predict the held-out data.

We computed a power spectrogram from 4000 1024-sample (23 ms) Hann windows taken from the middles of the same three songs used to evaluate marginal likelihoods: *Funky Kingston*, the *Kreutzer Sonata*, and *Pink Moon*. For each song, this yielded a $513 \times 4000$ spectrogram $\boldsymbol{X}$ describing 93 seconds of the song. We ran five-fold cross-validation to compare GaP-NMF's predictions of the missing high-frequency content to those of GIG-NMF, EU-NMF, and IS-NMF. (It is more difficult to evaluate KL-NMF's ability to predict missing data, since it does not correspond to a probabilistic model of continuous data.) We divided each spectrogram into five contiguous 800-frame sections. For each fold, we censored the top two octaves (i.e., the top 384 out of 513 frequency bins) of one of those sections. We then predicted the values in the censored bins based on the data in the uncensored bins.

The prior hyperparameters for the Bayesian models were set to $a = b = 1$, $c = 1/\bar{X}$, and $\alpha = 1$ (for GaP-NMF). We chose a higher value for $a$ and $b$ for this experiment since stronger smoothing can improve the models' ability to generalize to held-out data.

For each fit model, we computed an estimate $X_{mn}^{\text{pred}}$ of each missing value $X_{mn}^{\text{miss}}$. For the models fit using variational inference, we used the expected value of the missing data under the variational posterior $q$, $\mathbb{E}_q[\mathbb{E}_p[X_{mn}^{\text{miss}}]]$. For the GaP-NMF model, this expectation is

$$X_{mn}^{\text{pred}} = \mathbb{E}_q[\mathbb{E}_p[X_{mn}^{\text{miss}}]] = \textstyle\sum_k \mathbb{E}_q[\theta_k W_{mk} H_{kn}],$$

and for the GIG-NMF model it is

$$X_{mn}^{\text{pred}} = \mathbb{E}_q[\mathbb{E}_p[X_{mn}^{\text{miss}}]] = \textstyle\sum_k \mathbb{E}_q[W_{mk} H_{kn}].$$

For IS-NMF and EU-NMF we predicted $X_{mn}^{\text{pred}} = \sum_k W_{mk} H_{kn}$.

To evaluate the quality of fit for IS-NMF, GaP-NMF, and GIG-NMF, we compute the likelihood of each unobserved element $X_{mn}^{\text{miss}}$ under an exponential distribution with mean $X_{mn}^{\text{pred}}$. To evaluate EU-NMF, we first compute the mean squared error of the estimate of the observed data $\sigma^2 = \text{Mean}[(X^{\text{obs}} - [\boldsymbol{W}\boldsymbol{H}]^{\text{obs}})^2]$. We then compute the likelihood of each unobserved element $X_{mn}^{\text{miss}}$ under a normal distribution with mean $X_{mn}^{\text{pred}}$ and variance $\sigma^2$.

Figure 6.3 (right) plots the geometric mean of the likelihood of each unobserved element of $\boldsymbol{X}$ for the nonparametric model and for models fit with different numbers of components $K$. The Bayesian models do very well compared with the unregularized models, which overfit badly for any number of components $K$ greater than 1. GaP-NMF used fewer components to explain the songs than in the previous experiment, which we attribute to the stronger smoothing, smaller number of observations, and smaller window size.

### 6.4.4 Blind Monaural Source Separation

GaP-NMF can also be applied to blind source separation, where the goal is to recover a set of audio signals that combined to produce a mixed audio signal. For example, we may want to separate a polyphonic piece of music into notes to facilitate transcription [80], denoising, or upmixing [32].

The GaP-NMF model assumes that the audio signal is a linear combination of $L$ sources (some of which have extremely low gain). Given the complex magnitude spectrogram $\boldsymbol{X}^c$ of the original audio and an estimate of the model parameters $\boldsymbol{W}$, $\boldsymbol{H}$, and $\boldsymbol{\theta}$, we can compute maximum-likelihood estimates of the spectrograms of the $L$ unmixed sources using Wiener filtering [32]:

$$\hat{X}_{lmn} = X^c_{mn} \frac{\theta_l W_{ml} H_{ln}}{\sum_{i \in \{1,\ldots,L\}} \theta_i W_{mi} H_{in}}, \tag{6.24}$$

where $\hat{X}_{lmn}$ is the estimate of the complex magnitude spectrum of the $l$th source at time $n$ and frequency $n$. We can invert these spectrograms to obtain estimates of the audio signals that are combined in the mixed audio signal.

We evaluated GaP-NMF's ability to separate signals from two synthesized pieces of music. We used synthetic music rather than live performances so that we could easily isolate each note. The pieces we used were a randomly generated four-voice clarinet piece using 21 unique notes, and a two-part Bach invention synthesized using a physical model of a vibraphone using 36 unique notes.

We compared the Signal-to-Noise Ratios (SNRs) of the separated tracks for the GaP-NMF model with those obtained using GIG-NMF, IS-NMF, EU-NMF, and KL-NMF. For the finite models we also used Wiener filtering to separate the tracks, dropping $\theta$ from equation 6.24.

The models do not provide any explicit information about the correspondence between sources and notes. To decide which separated signal to associate with which note, we adopt the heuristic of assigning each note to the component $k$ whose gain signal $\boldsymbol{H}_k$ has the highest correlation with the power envelope of the true note signal. We only consider the $V$ components that make the largest contribution to the mixed signal, where $V$ is the true number of notes.

Figure 6.4 shows the average SNRs of the tracks corresponding to individual notes for each piece. The approaches based on the exponential likelihood model do comparably well. The KL-NMF and EU-NMF models perform considerably worse, and are sensitive to the model order $K$. GaP-NMF decomposed the clarinet piece into 34 components, and the

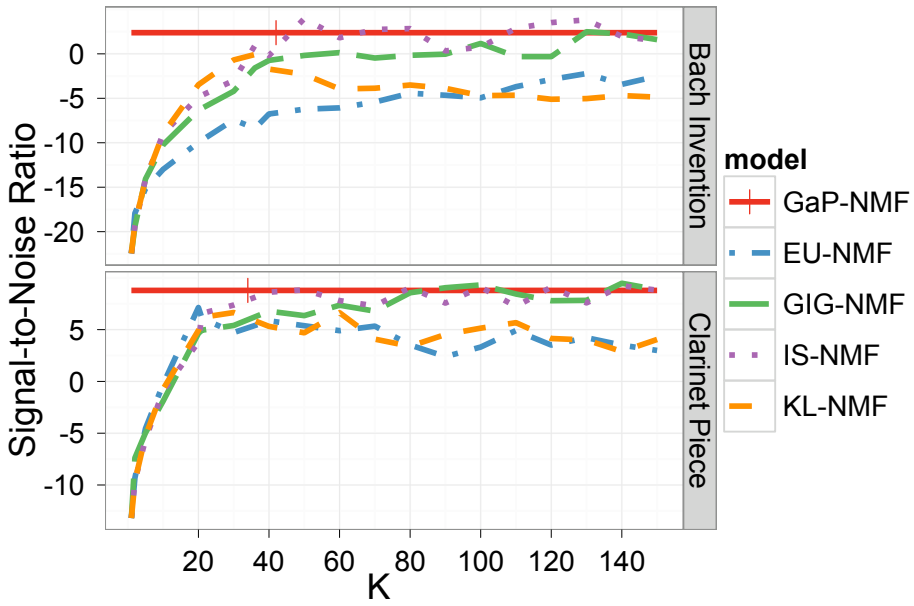


Figure 6.4: Average Signal-to-Noise Ratios (SNRs) across notes in the source separation task. Approaches based on the exponential likelihood model do well, EU-NMF and KL-NMF do less well. Ticks on the horizontal lines showing GaP-NMF's performance denote the final number of components $K$ used to explain the data.

Bach invention into 42 components. In both cases, some of these components were used to model the temporal evolution of the instrument sounds.

## 6.5 Related Work

GaP-NMF is closely related to recent work in Bayesian nonparametrics and probabilistic interpretations of NMF.

**Bayesian Nonparametrics** Most of the literature on Bayesian nonparametric latent factor models focuses on conjugate linear Gaussian models in conjunction with the Indian Buffet Process (IBP) [41] or the Beta Process (BP) [86], using either MCMC or variational methods for posterior inference [e.g. 26, 70]). (An exception that uses MCMC in non-conjugate infinite latent factor models is [84].)

A standard linear Gaussian likelihood model is not appropriate for audio spectrogram data, whereas the exponential likelihood model has theoretical justification and gives interpretable components. The nonlinearity and lack of conjugacy of the exponential likelihood model make inference using an IBP or BP difficult. Our use of a gamma process prior allows us to derive an infinite latent factor model that is appropriate for audio spectrograms and permits a simple and efficient variational inference algorithm.

**Probabilistic NMF** [33] suggest the outline of a variational inference algorithm for Itakura-Saito NMF based on the space-alternating generalized expectation-maximization algorithm in [32]. This approach introduces $K \times M \times N$ complex hidden variables whose posteriors must be estimated. In our informal experiments, this gave a much looser variational bound, much longer convergence times, and a less flexible approximate posterior than the variational inference algorithm presented in this chapter. The model order selection approach they suggest requires that many models be fit independently, whereas our GaP-NMF model can determine how many latent components are necessary without the need to fit multiple models.

Our approach of weighting the contribution of each component $k$ by a parameter $\theta_k$ resembles the strategy of Automatic Relevance Determination (ARD), which has been used in a Maximum A Posteriori (MAP) estimation algorithm for a different NMF cost function [83]. Though similar in spirit, this ARD approach is less amenable to fully Bayesian inference.

## 6.6 Discussion

We developed the GaP-NMF model, a Bayesian nonparametric model capable of determining the number of latent sources needed to explain an audio spectrogram. We demonstrated the effectiveness of the GaP-NMF model on several problems in analyzing and processing recorded music. Although this work has focused on analyzing music, GaP-NMF is equally applicable to other types of audio, such as speech or environmental sounds.

# Chapter 7

# HMM-Based Feature-Based Synthesis

Up to this point we have discussed applications of probabilistic modeling to computational audio analysis problems. In this and the following chapter we will explore applications of probabilistic modeling to audio synthesis.

## 7.1 Introduction

In contrast with discriminative models that typically try to infer response variables while making few or no assumptions about observable data, generative probabilistic models define a stochastic process that is assumed to be responsible for a set of observations. An interesting property of generative models is that, once fit, they can be used to synthesize new data. In most domains, there is little demand for this kind of artificial data, since the objective is to make predictions about data from the real world. But musical data may be an exception—many listeners care very little whether the musical "data" they listen to are "artificial" or "natural." Indeed, it could be argued that most music qualifies as artificial data manipulated for aesthetic purposes, in contrast to naturally occurring environmental sound.

In this chapter we explore a novel application of the Hidden Markov Model (HMM) and its Bayesian nonparametric cousin the HDP-HMM to data-driven music generation. By fitting an HDP-HMM to a sequence of feature vectors extracted from sliding windows of a recorded song, generating a new sequence of feature vectors from the trained model, and then using feature-based or concatenative synthesis to transform the feature vectors into audio, we can synthesize unlimited amounts of new audio based on a finite amount of training audio. This audio is of value both for its aesthetic interest[1] and for the subjective insights it gives into models of audio that may be used for analysis problems.

The rest of the chapter is organized as follows. We begin by reviewing some background on Markov chains, HMMs, feature-based synthesis, and concatenative synthesis. We then discuss the more recently developed hierarchical Dirichlet process HMM, which is better suited than the traditional HMM to the long and complex input sequences produced by analyzing some musical audio. Finally, we present some experimental results that demonstrate the viability of our approach and discuss directions for future work.

[1] To an admittedly small audience.

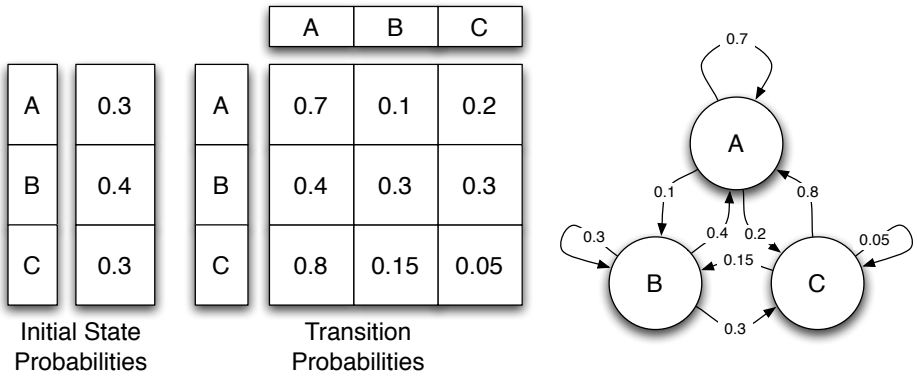


Figure 7.1: Initial state probabilities, transition matrix, and finite-state machine representation of a simple first-order three-state Markov chain.

## 7.2 Markov Chains

A first-order Markov chain is defined by a set of states, the transition probabilities between those states, and the probability of starting in each state. The transition matrix contains a vector of probabilities for each state describing how likely it is that the process will transition from that state to each other state. For example, in the model in figure 7.1, if at time $t$ the model is in state $A$, at time $t + 1$ there is a $70\%$ chance of being in state $A$, a $10\%$ chance of being in state $B$, and a $20\%$ chance of being in state $C$. The probability of generating the sequence $ABAC$ would be $0.3 \cdot 0.1 \cdot 0.4 \cdot 0.2 = 0.024$, that is, the probability of starting in state $A$ multiplied by the probabilities of the three transitions $AB$, $BA$, and $AC$.

The assumption that the future state of a process depends only on its single most recent state is often unreasonable. In such cases, it makes sense to take more than one previous state into account using a higher-order Markov chain. Higher-order Markov chains can be defined by letting transition probabilities depend on the last $N$ states in addition to the most recent state. Figure 7.2 shows a simple second-order Markov chain's starting probabilities vector and transition matrix, as well as a few example state sequences and their likelihoods.

Fitting Markov chains (of any order) to data is fairly straightforward if state values are directly observable, as they are for symbolic data such as text and musical scores. To obtain the Markov chain under which the observed data are most likely to have occurred, one simply sets the transition probability vector from each state (or sequence of $N$ states for an order-$N$ model) to match the relative frequencies of each observed transition. For example, in the sequence $AABBABBA$ we find one $AA$ transition and two $AB$ transitions, so the likelihood of going from $A$ to $A$ would be $1/3$ and the likelihood of going from $A$ to $B$ would be $2/3$; there are two $BA$ transitions and two $BB$ transitions, so the likelihood of going from $B$ to $A$ would be $2/4$ and the likelihood of going from $B$ to $B$ would also be $2/4$.

Once their parameters have been inferred, Markov chains can be used to generate new state sequences of arbitrary length that bear some resemblance to the data on which they were trained, a property that has been used to creative ends in interesting ways. The poet Jeff Harrison has generated poems using Markov chains fit to text documents (treating each

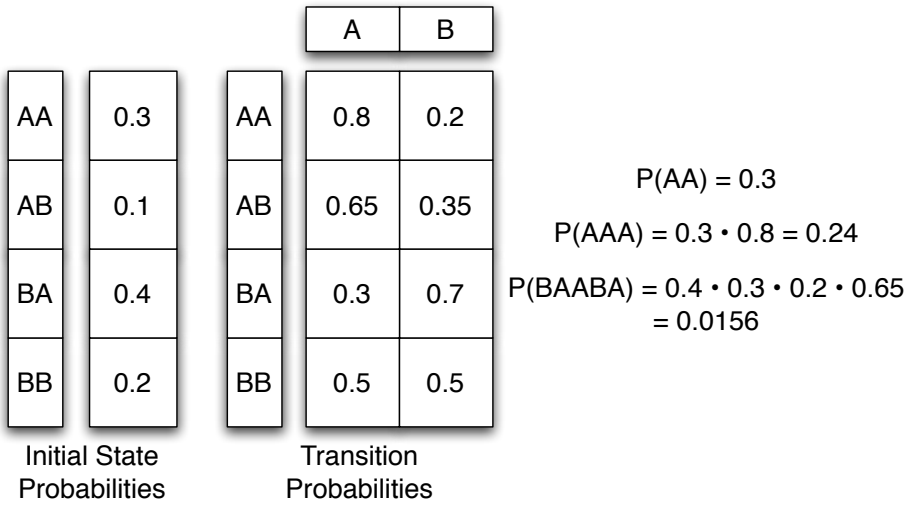


Figure 7.2: Initial state probabilities and transition matrix of a second-order two-state Markov chain, as well as some example sequences and their likelihoods.

unique word as a state); a few Bell Labs researchers fit a Markov chain on posts from the net.singles Usenet group and posted the random posts it generated back to the group under the punning pseudonym "Mark V. Shaney."[2] Markov chains have also been used since at least the 1950's to produce musical scores [75, 5].

## 7.3 Hidden Markov Models

Although simple Markov chains lend themselves well to applications involving symbolic data, to model continuous data such as feature vector sequences describing audio we need to add another layer of complexity. The hidden Markov model assumes that there is still a set of states generating our data, and that the identity of each successive state still depends only on the state(s) before it, but now we cannot observe these states directly. Instead, each state is associated with an emission probability density function (PDF) that generates our observed data. Figure 7.3 represents a three-state HMM in finite-state machine form, and figure 7.4 shows the graphical model representation of a (Bayesian) HMM with multivariate Gaussian observations. Each hidden variable $z_t$ depends both on the transition matrix $\boldsymbol{\pi}$ and on the previous hidden state $z_{t-1}$.

Parameter inference for HMMs is more complicated than inferring simple Markov chains, and is usually done either by attempting to maximize the likelihood of the training data (using the Baum-Welch expectation maximization algorithm [74]) or by placing priors on the HMM's parameters and trying to find parameters that maximize the posterior likelihood of the data [35]. Alternately, one can take a fully Bayesian approach and use MCMC sampling or variational inference to reason about the posterior (as in e.g. [34] and [11]).

[2]Those interested in playing with letter-based Markov chain text generation can open a text document in GNU Emacs and type "<META>-x dissociated-press."

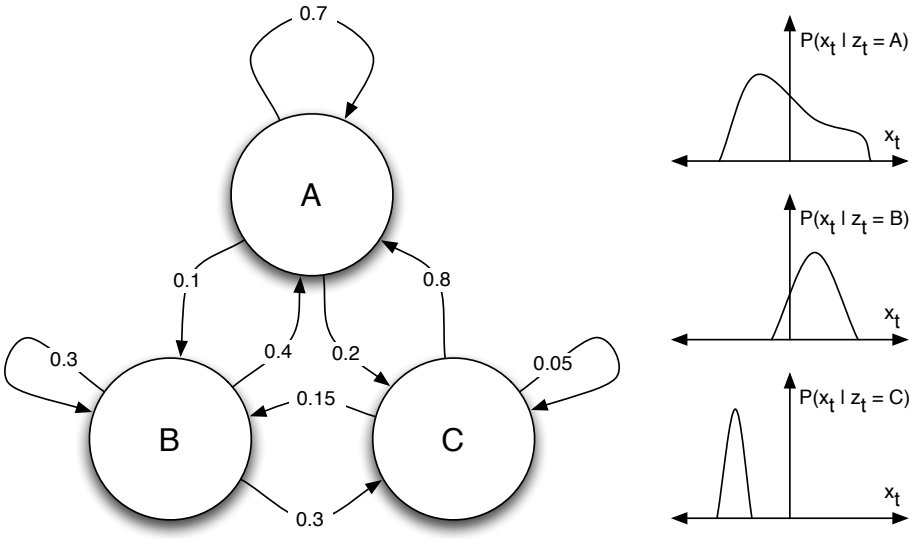


Figure 7.3: One perspective on the hidden Markov model. Left: Finite-state machine representation of transition probabilities between the hidden states $A$, $B$, and $C$. Right: Probability density functions (PDFs) for the observed data given the underlying state of the model $\boldsymbol{x}_t$ is the observation at time $t$, and $z_t$ is the underlying state label at time $t$.

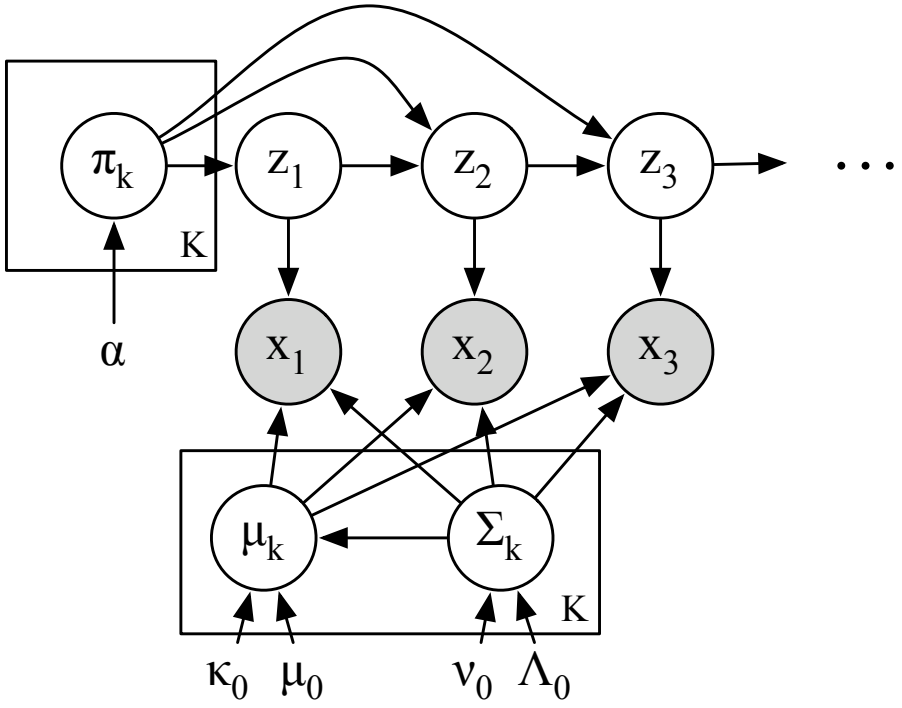


Figure 7.4: Graphical model representation of a Bayesian first-order HMM with Gaussian emission distributions.

### 7.3.1 Higher-order HMMs

Although it is possible to define higher-order HMMs as well as first-order HMMs, as the order grows even moderately high the transition matrix grows exponentially and becomes increasingly difficult to learn. A suitable workaround for our purposes is to build higher-order Markov models based on the state sequences obtained by fitting low-order HMMs. If we take these state sequences as observed "true" data, then we can build Markov models of arbitrarily high order as described for symbolic data above. This is much the same approach as that taken by Casey [21] to build 2nd- to 8th-order lexemes for audio retrieval.

### 7.3.2 Generating new feature vector sequences

Once an HMM has been fit, it can be used to generate new sequences of observations in much the same way as a Markov chain with no hidden layer. First we generate a sequence of state assignments, then for each time step we draw an "observation" from the PDF associated with the state for that time step.

For audio, we might fit an HMM to a sequence of vectors of 20 Mel-Frequency Cepstral Coefficients (MFCCs) extracted from a recorded song, assuming that each state generates a 20-dimensional MFCC vector from its own multivariate normal distribution. The inference process estimates the initial state probability vector, transition probability matrix, and the means and covariance matrices associated with each state. These estimated parameters describe a model that could have generated the training song. Once the model has been fit, we can then generate a new sequence of MFCC vectors as described above, and that sequence of MFCC vectors will describe another "song" that could have been generated by our model. This sequence of MFCC vectors is of little interest, however, unless we can turn it into something audible.

## 7.4 Feature-Based and Concatenative Synthesis

The question of how to use feature vectors describing short windows of audio to drive sound synthesis has received some attention. We will consider several approaches to bridging the gap between HMM-generated state sequences and audio.

### 7.4.1 Feature-based synthesis

Some of our previous research [46] has focused on the problem of feature-based synthesis, which involves finding ways of synthesizing audio characterized by a desired vector of features. For some feature sets, efficient closed-form solutions exist to this problem. The MFCC extraction process, for example, can be reversed step by step to produce noisy audio with the appropriate coarse spectral character.

It is not necessarily as easy to reverse arbitrary feature vectors consisting of multiple concatenated feature sets, however. In such circumstances we can resort to global optimization algorithms such as simulated annealing or genetic algorithms to find synthesizer

parameters that will produce audio described by a feature vector as close as possible to the desired feature vector.

### 7.4.2 Concatenative synthesis

Another approach uses concatenative synthesis to produce audio matching feature descriptors. This approach searches for previously recorded grains of audio that closely match the desired feature values [76]. These short grains are retrieved from a large, efficiently indexed database. This approach is potentially more flexible than feature-based synthesis (since the database is not limited to synthetic sounds) and can have much less computational overhead than repeatedly synthesizing and testing new short audio segments. However, it requires that a sufficiently large database be compiled, analyzed, and indexed ahead of time, and leaves no recourse if no clips described by an adequately similar feature vector are in the database.

### 7.4.3 Cluster mosaicing

We also consider another concatenative approach, this one leveraging information that comes directly out of fitting the HMM. Assuming that we still have the audio from which the feature vector sequence(s) used to train the HMM were extracted, we can use the maximum-likelihood state sequence obtained using the Viterbi algorithm during training to associate each window from the training audio with a state of the HMM.

To generate audio for a new state sequence, for each time step $t$ we can simply choose a window uniformly at random from those windows whose state labeling is the same as the current state $z_t$, add it to the end of the current audio stream (with appropriate crossfading), and move on to the next time step $t+1$. This is in lieu of drawing from the emission density associated with $z_t$. This density should nonetheless be reasonably well approximated, since the empirical distribution of observations is what the emission density is supposed to be modeling in the first place.

Note that the database of windows available to this technique can be expanded beyond those provided by the audio used for training. The Viterbi algorithm can also be used to provide maximum-likelihood labelings for new audio recordings, and each window of a new recording can be associated with its appropriate state. Doing so does, however, make it more likely that the empirical distribution of available windows for each state will become skewed. In this case, it may be best to actually draw a feature vector from the emission density and choose the cluster member with the smallest Euclidean distance, or to use some other heuristic.

Elements of this approach resemble the audio oracle, which deterministically creates a Markov chain in which each window of audio is a state [27]. Another approach suggested by Casey also used HMMs to cluster windows of audio, with a focus on indexing large databases for audio mosaicing [21].

## 7.5 The HDP-HMM

When using traditional HMMs one must decide ahead of time how many states are necessary to capture the complexity of one's data. Choosing too few states results in an inadequately rich model, while choosing too many may result in overfitting or difficulties in training. By exploiting the close relationship between mixture models and HMMs, the Hierarchical Dirichlet Process (HDP) can be extended to produce the HDP-HMM, an HMM with a countably infinite number of states that can make a principled choice of how many states are needed to explain the data [85].

### 7.5.1 The basic model

In the standard generative HDP-HMM model, an HDP-HMM is created by first drawing a vector $\boldsymbol{\beta}$ of infinite length (whose elements sum to one) from Sethuraman's stick-breaking construction [77] with hyperparameter $\gamma$, denoted GEM($\gamma$). This vector $\boldsymbol{\beta}$ describes the relative frequencies with which states are expected to be visited for the whole model. $\boldsymbol{\beta}$ defines a multinomial distribution over states. Next, for each state $k$ in $\{1, \ldots, \infty\}$ a vector $\boldsymbol{\pi}_k$ (also summing to one) is drawn from a DP with hyperparameter $\alpha$ and base distribution $\boldsymbol{\beta}$. This vector $\boldsymbol{\pi}_k$ defines a multinomial distribution over transitions from state $k$, that is, it is the transition vector associated with state $k$. The higher the value of $\alpha$, the less likely it becomes that $\pi_k$ will deviate significantly from $\boldsymbol{\beta}$. Next, parameters $\boldsymbol{\phi}_k$ for the emission distributions $f(\boldsymbol{\phi})$ associated with each state $k$ are drawn from their previously specified prior distributions $H$. In summary:

$$\begin{aligned} \boldsymbol{\beta} &\sim \text{GEM}(\gamma) \\ \boldsymbol{\pi}_k &\sim \text{Dirichlet}(\alpha\boldsymbol{\beta}) \\ \boldsymbol{\phi}_k &\sim H. \end{aligned} \tag{7.1}$$

Note that this process exactly corresponds to that used in the HDP to generate the parameters $\boldsymbol{\beta}$, $\boldsymbol{\pi}$, and $\boldsymbol{\phi}$.

Next, it is assumed that such a model generated our training data $\boldsymbol{y}$, as well as a state sequence $\boldsymbol{z}$. Assume for simplicity that the starting state $z_1$ is drawn from $\boldsymbol{\beta}$. Then $z_{t+1}$ is drawn from $\boldsymbol{\pi}_{z_t}$ for $t = 1, \ldots, N-1$. Once each $z_t$ is drawn, each observation $\boldsymbol{y}_t$ (the observed data) is drawn from $f(\boldsymbol{\phi}_{z_t})$:

$$\begin{aligned} z_t &\sim \text{Multinomial}(\boldsymbol{\pi}_{z_{t-1}}) \\ \boldsymbol{y}_t &\sim f(\boldsymbol{\phi}_{z_t}) \end{aligned} \tag{7.2}$$

Given $\boldsymbol{y}$, and having defined $H$ and chosen values for $\alpha$ and $\gamma$, we can use the Gibbs sampling method described by Teh et al. to infer the state assignments $\boldsymbol{z}$. Given $\boldsymbol{z}$, we can then infer those parts of $\boldsymbol{\beta}$, $\boldsymbol{\pi}$, and $\boldsymbol{\phi}$ that we care about—that is, those associated with states that are associated with observations. There are still theoretically an infinite number of states in the model, but those states that are not associated with observations can be dealt with in the aggregate. This algorithm requires that $H$ be a conjugate prior distribution on $f(\theta)$, however, which can be a problematic restriction.

Another Gibbs sampling method can be used to train the HDP-HMM, which Fox et al. call the blocked-z Gibbs sampler [34]. This method operates on a finite approximation to the full HDP-HMM that uses $L$ states instead of an infinite number of states, and converges to the HDP-HMM as $L \to \infty$:

$$\begin{aligned} \boldsymbol{\beta} &\sim \text{Dirichlet}(\gamma/L, \ldots, \gamma/L) \\ \boldsymbol{\pi}_k &\sim \text{Dirichlet}(\alpha\boldsymbol{\beta}); \quad \boldsymbol{\phi}_k \sim H \\ z_t &\sim \pi_{z_{t-1}}; \quad \boldsymbol{y}_t \sim f(\boldsymbol{\phi}_{z_t}) \end{aligned} \tag{7.3}$$

The blocked-z Gibbs sampler fully instantiates and samples $\boldsymbol{\beta}$, $\boldsymbol{\pi}$, $\boldsymbol{z}$, and $\boldsymbol{\phi}$ for this finite model, and can therefore exploit the relatively simple structure of the HMM to sample $Z$ jointly using a variant of the forward-backward procedure [74], which speeds convergence. Another advantage of this algorithm is that it does not require that $H$ be a conjugate prior. Note that although the number of states $L$ to use when training the model is specified, as long as $L$ is sufficiently large not all $L$ states will be used, and as $L$ becomes very large the model converges to the infinite HDP-HMM described previously, in which only a finite number of states are actually used. In practice, setting $L$ to be more than 2–4 times the number of states that the model winds up using does not seem to significantly alter results.

Although the hyperparameters $\alpha$ and $\gamma$ are assumed to be given, we can place vague prior distributions (we used Gamma$(1, 0.005)$) on each of them and let the model choose them as well [34]. If we allow the model to control $\alpha$ and $\gamma$ then the only parameters we need to specify are those associated with the priors over emission densities.

### 7.5.2 Priors on emission density parameters

We choose our prior densities to require only two parameters to be manually set – one controlling the expected size of each cluster and one controlling how much to allow cluster sizes to vary. If we specify a preference for smaller clusters, then the training algorithm will respond by allocating more clusters and therefore a richer model to capture the complexity of the data. The stronger a preference we specify for clusters of a particular size, the less variation there will be in cluster size.

Our emission distributions are multivariate normal. The conjugate prior distribution for the mean and covariance parameters of a multivariate normal distribution is the normal-inverse-Wishart [36]. Determining the posterior of this distribution based on observed samples is computationally straightforward (although one must be careful of numerical issues). But the normal-inverse-Wishart distribution makes it difficult to specify different levels of prior certainty about the shape of the distribution (as captured by the correlations between dimensions) and its spread (the standard deviations of the dimensions). Ideally we would like to be able to express the sort of preferences with respect to spread described in the previous paragraph while letting the clusters take any shape the data suggest.

Mathematically, we accomplish this by factorizing the multivariate normal distribution's covariance matrix parameter $\boldsymbol{\Sigma}$ into $\boldsymbol{\Sigma} = \boldsymbol{SRS}$, where $\boldsymbol{S}$ is a square matrix with the standard deviation for each dimension on its diagonal and zeros elsewhere and $\boldsymbol{R}$ is a square symmetric positive definite correlation matrix where $R_{ij}$ is the correlation between dimensions $i$ and $j$. For each dimension we place an independent scaled inverse-$\chi^2(\nu, \sigma^2)$

distribution on the square of its standard deviation (i.e. its variance), where $\sigma^2$ is our desired or expected average cluster variance in that dimension and $\nu$ is a degrees of freedom parameter that controls how much weight $\sigma^2$ receives in posterior inference. Depending on the data, it may make sense to specify different values of $\sigma^2$ for each dimension. One way to avoid having to do this manually is to look at the empirical standard deviations of the entire sequence (or the average standard deviations of smaller clusters) of feature vectors, scale them all by the same constant, and square them. This only requires that one parameter be specified a priori, while keeping the proportions of the emission distributions reasonable with respect to the data.

Unfortunately, few off-the-shelf distributions over symmetric positive definite matrices exist, so we use a variant on a parameter expansion technique described by Boscardin and Zhang [19]. We define an auxiliary diagonal standard deviation matrix $\boldsymbol{Q}$ and place an inverse-Wishart prior $\mathcal{IW}(\boldsymbol{I}, d+1)$ on the positive definite matrix $\boldsymbol{QRQ}$, where $d$ is the dimensionality of the feature vectors and $\boldsymbol{I}$ is the $d$-dimensional identity matrix. The marginal prior distribution of each correlation $R_{ij}$ under this distribution is uniform from -1 to 1 [36]. Although we no longer have a conjugate prior for the covariance matrix, we can use Gibbs sampling and the Metropolis-Hastings [43] algorithm to sample from the posterior distributions of $\boldsymbol{S}$, $\boldsymbol{R}$, and $\boldsymbol{Q}$, and therefore from the posterior distribution of $\boldsymbol{\Sigma} = \boldsymbol{SRS}$.

To simplify computation, we choose a conjugate multivariate normal prior on the mean of our emission distribution [36].

### 7.5.3 Adding another layer of hierarchy

We add another layer of hierarchy to the standard HDP-HMM in order to allow ourselves to fit models for multiple songs simultaneously in a way that allows these models to share a common vocabulary of states. Sharing state vocabularies across models allows us to use the technique described in section 7.3.1 to build Markov chains that combine the transition characteristics of multiple songs.

The new generative model is:

$$\begin{aligned} \boldsymbol{\beta}_0 &\sim \text{GEM}(\delta) \\ \boldsymbol{\beta}_i &\sim \text{Dirichlet}(\gamma\boldsymbol{\beta}_0) \\ \boldsymbol{\pi}_{ij} &\sim \text{Dirichlet}(\alpha\boldsymbol{\beta}_i); \quad z_{it} \sim \boldsymbol{\pi}_{z_{i,t-1}} \\ \boldsymbol{\phi}_k &\sim H; \quad \boldsymbol{y}_{it} \sim f(\boldsymbol{\phi}_{z_{it}}), \end{aligned} \tag{7.4}$$

where each song $i$ has its own song-level state likelihood vector $\boldsymbol{\beta}_i$, transition matrix $\boldsymbol{\pi}_i$, state sequence $\boldsymbol{z}_i$, and observation sequence $\boldsymbol{y}_i$, and the emission density parameters $\boldsymbol{\phi}$ are shared across all models. $\boldsymbol{\beta}_0$ is an infinitely long vector that defines the global likelihood of being in a particular state across all songs, while each $\boldsymbol{\beta}_i$ defines the likelihood of being in a particular state for song $i$. The hyperparameter $\gamma$ determines how much each $\boldsymbol{\beta}_i$ is likely to deviate from $\boldsymbol{\beta}_0$, much like $\alpha$ determines how much each $\boldsymbol{\pi}_{ij}$ is likely to deviate from each $\boldsymbol{\beta}_i$. The same type of prior can be placed on $\delta$ as on $\alpha$ and $\gamma$, so it can be inferred from data in the same way.

## 7.6 Experiments

We ran experiments on the dance-pop song "Chewing Gum" by the Norwegian recording artist Annie. The song was selected because of the prominence of its strong, repeating beat. Sound examples and a link to a stream of the original song are available at `http://www.cs.princeton.edu/~mdhoffma/icmc2008`. All algorithms were implemented in MATLAB and C++.

We began by breaking the song into non-overlapping 1024-sample windows and extracting each window's RMS power and first 20 Log-Frequency Cepstral Coefficients (LFCCs) [21], resulting in a 21-dimensional feature vector for each window, 10,086 feature vectors in all. We chose these features because they are simple to compute and, more importantly, simple and efficient to reverse. In the future, we plan on experimenting more with other features, particularly chroma vectors.

As a preprocessing stage, we ran Principal Component Analysis (PCA) on our 10,086 feature vectors to ensure that no global correlations exist between the 21 dimensions of the data we are trying to model. We transform our synthetic feature vectors back to their original basis before transforming them into audio. We then ran the blocked-z Gibbs sampler from [34] with $L = 175$ until the log-probability of the data under the model failed to find a new maximum for several iterations. We chose the parameters for our priors as follows.

For the $\sigma^2$ parameters to the scaled-inverse-$\chi^2$ priors on each variance, we chose the average variance in each respective dimension of 100 clusters of 70 points each. The clusters were selected at random by first choosing a random point and then finding the 70 closest points to that first point under the $\ell^2$ (Euclidean) norm. The goal was to push the model to expect to find clusters with a median size of about 70 points. We chose 50 for the degrees of freedom parameter to each scaled-inverse-$\chi^2$ prior.

The mean and covariance parameters to the prior on the emission distribution's mean were chosen as the mean and the covariance matrix of the entire data set.

### 7.6.1 Results

Figure 7.5 shows a spectrogram of a roughly 10-second clip from "Chewing Gum." Below it is a spectrogram of a resynthesized version of the same clip generated by extracting the feature set described above and then converting the result back into audio. Notice that all fine detail in the spectrum has been washed out. This is because low-order cepstral coefficients are designed to smooth away such fine detail.

Figure 7.6 compares spectrograms of 10-second audio clips generated by 1st, 4th, and 8th-order Markov chains created from our trained model. We manually reduced the variance of each emission density by $50\%$ when generating new feature sequences to achieve a less noisy result. As information about more previous states is included, the model can produce audio with more structure, but also finds itself more constrained. If the order of the model $N$ becomes too high, there will be almost no state sequences of length $N$ that are not unique, and the model will be forced to reproduce the original state sequence.

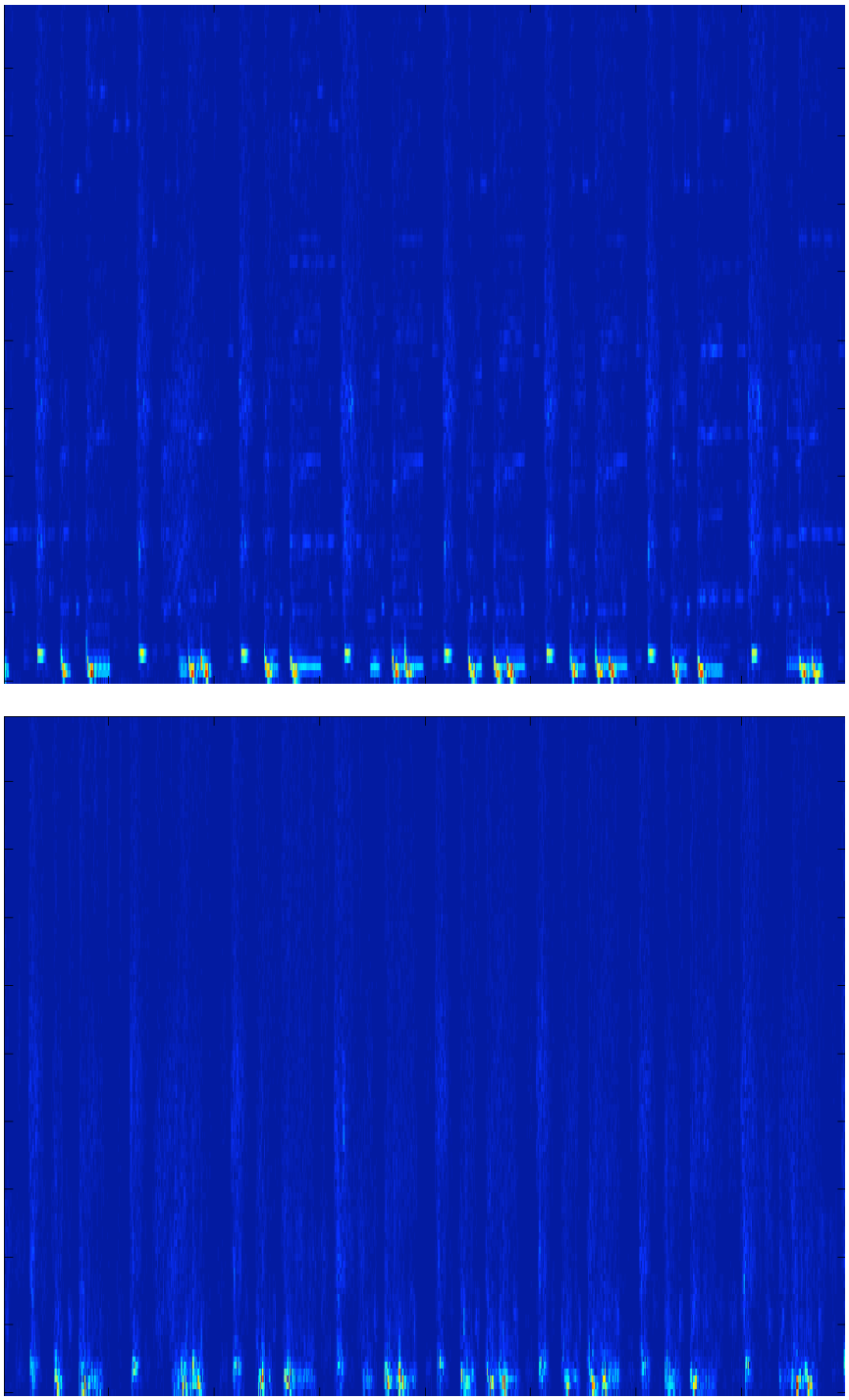

Figure 7.5: Top: spectrogram of a clip from “Chewing Gum.” Bottom: spectrogram of resynthesized clip.

## 7.7 Discussion

Although our early results are interesting, they reveal some limitations of HMMs fit to features extracted from mixed recordings. In particular, a separate state must be learned for each distinguishable combination of instrument sounds. This leads to a large number of states being necessary, which in turn leads to a very large and difficult-to-learn transition matrix. In the next chapter we will look at using a model for synthesis that uses a more factorial, source-based approach.

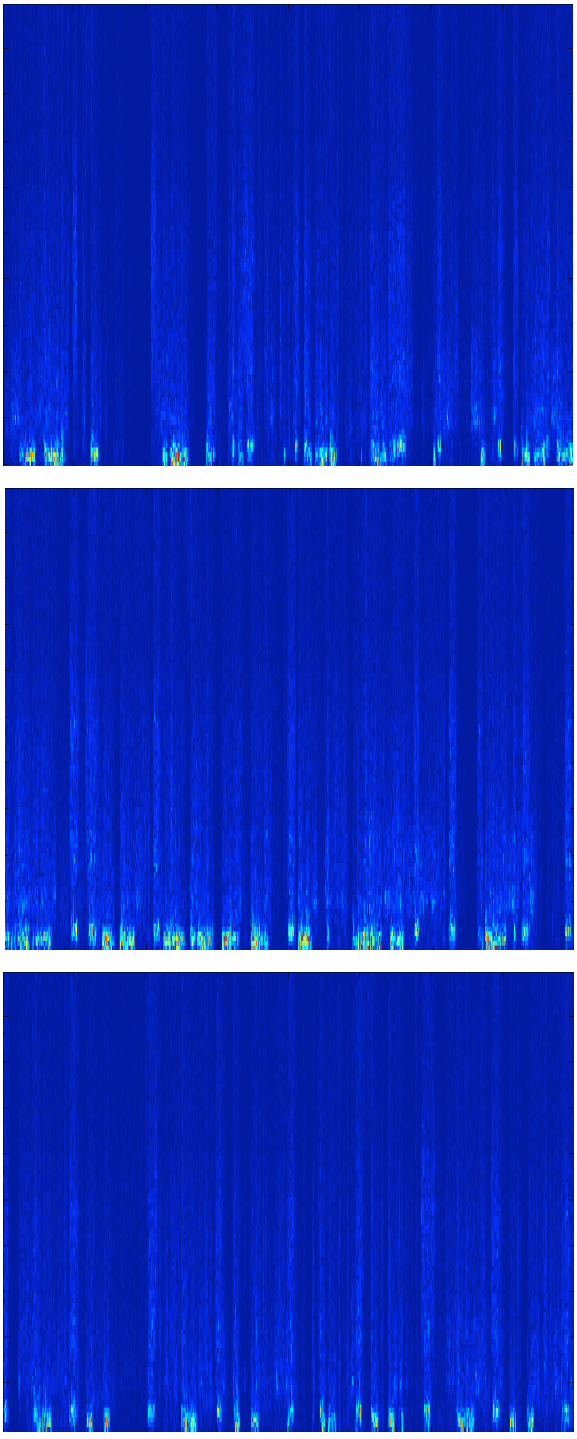

Figure 7.6: Spectrograms of audio produced by (from top to bottom) a 1st-order Markov chain, 4th-order Markov chain, and an 8th-order Markov chain.

# Chapter 8

# Bayesian Spectral Matching: Turning Young MC Into MC Hammer Via MCMC Sampling

## 8.1 Introduction

In the previous chapter, we used Hidden Markov Models (HMMs) fit to features of mixed audio signals to generate novel audio. One limitation of working with this sort of feature-based representation is that the number of distinguishable mixed audio signals has the potential to grow exponentially in the number of sound sources that may be mixed. When working with mixture models or HMMs, this may mean that a very large number of states is needed to adequately capture variation in the audio signal.

In chapters 5 and 6, we developed model formulations that explicitly take into account the fact that musical audio signals are usually mixtures of more than one source. In this chapter we adapt the SI-HDP developed in chapter 5 to the problem of audio mosaicing [99, 57]. The problem we want to solve is this: given a set of (short) recorded source sounds, how can we match a (longer) target sound as closely as possible by repeating and combining our source sounds at different times and amplitudes? More formally, we have a set of $K$ source sounds $\boldsymbol{x}_k$, and we want to find a set of $K$ functions $g(t, k)$ with which to convolve each sound $\boldsymbol{x}_k$ such that the sum of these convolutions $\boldsymbol{z}$ is perceptually similar to our target sound:

$$z(t) = \sum_{k=1}^{K} \sum_{u=0}^{\infty} g(t-u, k) x_k(u) \tag{8.1}$$

We define a probabilistic generative model, the Shift-Invariant Mixture of Multinomials (SIMM), corresponding to a parametric process that we will use to generate our output sound from our source sounds. If we assume that this model actually generated our target sound, the problem of combining our sources to match the target sound becomes one of parameter inference. SIMM has a matrix of hidden variables $\boldsymbol{\omega}$ that correspond to the functions $g(t, k)$ that we want to find. We can find a good set of functions $g(t, k)$ by finding a value for $\boldsymbol{\omega}$ with high posterior likelihood given the target sound—that is, a value for $\boldsymbol{\omega}$

that could plausibly have led to our model generating our target sound. Our probabilistic construction allows us to use a Gibbs sampling algorithm to perform approximate posterior inference [67].

In the sequel, we describe our generative model, define a Gibbs sampler to infer the model's hidden variables, show how those hidden variables tell us how to produce our output sound, and present the results of applying our approach to various combinations of input sources and target sounds.

## 8.2 The SIMM Model

Our SIMM model is adapted from the Shift-Invariant Hierarchical Dirichlet Process (SI-HDP) defined in chapter 5.

### 8.2.1 Data Representation

We begin by computing the magnitude spectrogram of our target audio using $W$ non-overlapping windows of $S$ samples each (multiplied by a Hann window), yielding $B = \frac{S}{2} + 1$ frequency bins per window [1]. We will refer to the magnitude in bin $b$ of window $w$ as $\hat{y}_{wb}$. We normalize the magnitude spectrogram $\hat{\boldsymbol{y}}$ so that $\sum_{b=1}^{B} \sum_{w=1}^{W} \hat{y}_{wb} = 1$.

We compute a scaled and quantized version of $\hat{\boldsymbol{y}}$, $\bar{\boldsymbol{y}}$, which we will treat as a histogram giving the counts of amplitude quanta at each time $w$ and frequency bin $b$:

$$\bar{y}_{wb} = \text{round}(WB\nu\hat{y}_{wb}) \tag{8.2}$$

$$N = \sum_{b=1}^{B} \sum_{w=1}^{W} \bar{y}_{wb} \tag{8.3}$$

$\nu$ is a constant controlling how finely we quantize the spectrogram. Choosing $\nu = 1$ gives us an average of about one quantum per window/bin; higher values of $\nu$ yield a closer approximation to the continuous spectrogram and more expense. The order of these quanta is arbitrary, so we can model them as being drawn independently from our model.

### 8.2.2 Generative Process

We assume we are given a set of $K$ normalized magnitude spectrogram matrices $\boldsymbol{\phi}_k$ of size $C \times B$, such that $\phi_{kcb}$ is the magnitude in frequency bin $b$ at window $c$ in sound source $k$, and $\sum_{c=1}^{C} \sum_{b=1}^{B} \phi_{kcb} = 1$ for each $k \in \{1, \ldots, K\}$. These spectrograms come from the sound sources we will use to reconstruct the target sound. The normalized spectrograms can also be interpreted as joint multinomial distributions over base times $c$ and bins $b$. That is, $\phi_{kcb}$ gives the probability of drawing a quantum $i$ with base time $c$ and frequency $b$ given that the quantum is coming from the $k$th source sound.

The generative process for SIMM is:

[1] A shorter hop size can be used, but using non-overlapping windows is simpler and reduces computational overhead. A lack of time resolution has not been a problem in our experiments.

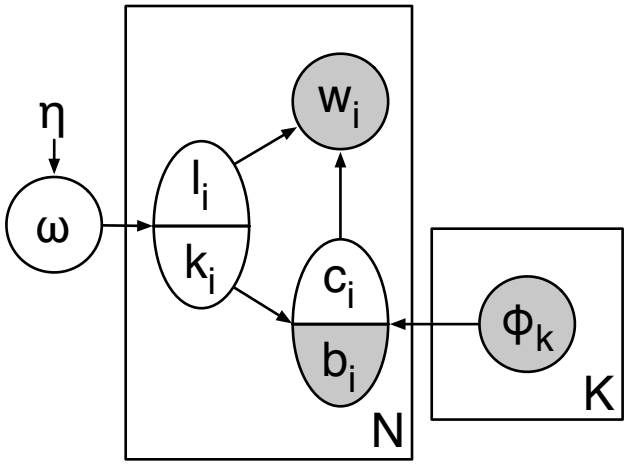


Figure 8.1: The graphical model for SIMM. Nodes with two variable names denote tuples drawn jointly—for example, $c_i$ and $b_i$ are drawn jointly from a multinomial distribution with parameter $\boldsymbol{\phi}_{k_i}$, and depend on both $k_i$ and $\boldsymbol{\phi}_{k_i}$. Only $b_i$ is directly observed, so only that half of the node is shaded.

1. Draw a $K \times L$ matrix $\boldsymbol{\omega}$ defining a joint multinomial distribution over sources $k$ and time offsets $l$ from a symmetric Dirichlet distribution with parameter $\eta$:

$$\boldsymbol{\omega} \sim \text{Dirichlet}(\eta, \ldots, \eta) \tag{8.4}$$

   $\omega_{kl}$ is the joint probability of drawing a quantum from source $k$ with time offset $l$.

2. For each quantum $i \in \{1, \ldots, N\}$:

   (a) Draw a source ID $k_i$ and a time offset $l_i$ jointly from Multinomial($\boldsymbol{\omega}$):

$$\{k_i, l_i\} \sim \text{Multinomial}(\boldsymbol{\omega}) \tag{8.5}$$

   (b) Draw a base time $c_i$ and a frequency bin $b_i$ jointly from the spectrogram/joint distribution $\boldsymbol{\phi}_{k_i}$:

$$\{c_i, b_i\} \sim \text{Multinomial}(\boldsymbol{\phi}_{k_i}) \tag{8.6}$$

   (c) Set the observed time $w_i$ for quantum $i$ based on the base time $c_i$ and the time offset $l_i$:

$$w_i = c_i + l_i \tag{8.7}$$

3. For each time $w$ and frequency $B$, count the quanta appearing at $w$ and $b$ to yield $\bar{y}_{wb}$, the magnitude in the quantized spectrogram at $w$ and $b$.

Each observed quantum $i$ appears at time $w_i$ and frequency bin $b_i$, which are selected according to the process above. We assume that quanta always add constructively. As discussed earlier, this signal model has little theoretical justification, but makes inference

simple and works reasonably well in practice. One could equally well develop a model based on the more principled assumptions made by GaP-NMF, and adapt the variational inference algorithm from chapter 6 to this problem.

Figure 8.1 shows SIMM as a graphical model, which summarizes the dependencies between the variables. Given this generative process and an observed spectrogram $\hat{\boldsymbol{y}}$, we will infer values for the process's hidden parameters $\boldsymbol{k}, \boldsymbol{l}$, and $\boldsymbol{\omega}$.

## 8.3 Inference and Synthesis

Our primary objective is to find a good value for the matrix $\boldsymbol{\omega}$, which defines the joint distribution over time offsets $l$ and sources $k$. Once we have inferred $\boldsymbol{\omega}$ from the data, it will tell us by how much to time-shift and scale each short component to recreate the target sound.

### 8.3.1 Gibbs Sampler

We use Gibbs sampling, a Markov Chain Monte Carlo (MCMC) technique that allows us to approximate a sample from the posterior distribution $p(\boldsymbol{k}, \boldsymbol{l}|\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{\phi}, \eta)$, since this distribution is difficult to compute analytically. In Gibbs sampling, we repeatedly sample new values for each variable conditioned on the values of all other variables. After an initial "burn-in" period, the distribution of the sampled $\boldsymbol{k}$ and $\boldsymbol{l}$ converges to their true posterior distribution [67].

We can avoid sampling $\boldsymbol{\omega}$, since we have placed a conjugate Dirichlet prior on $\boldsymbol{\omega}$ and can therefore compute the posterior predictive likelihood of $\{k_i, l_i\}$ given the other $k$'s and $l$'s (denoted $\boldsymbol{k}_{-i}$ and $\boldsymbol{l}_{-i}$) and the hyperparameter $\eta$. We therefore resample only the values for the source indicators $\boldsymbol{k}$ and the time offsets $\boldsymbol{l}$. This leads to faster convergence, since it lets us work in a lower-dimensional space. Once we have estimates for $\boldsymbol{k}$ and $\boldsymbol{l}$, we can compute the Maximum A Posteriori (MAP) value for $\boldsymbol{\omega}|\boldsymbol{k}, \boldsymbol{l}, \eta$.

To resample each pair $k_i, l_i$, we need to compute the joint posterior likelihood that the quantum $i$ appearing at time $w_i$ and bin $b_i$ was drawn from a source $k$ at a time offset $l$, holding all other variables fixed:

$$\begin{aligned} p(k_i = k, l_i = l|w_i, b_i, \boldsymbol{k}_{-i}, \boldsymbol{l}_{-i}, \boldsymbol{\phi}, \eta) \propto \\ p(c_i = w_i - l, b_i|k_i = k, l_i = l, \boldsymbol{\phi}) \times \\ p(k_i = k, l_i = l|k_{-i}, l_{-i}, \eta) \end{aligned} \tag{8.8}$$

The joint likelihood of the base time $c_i = w_i - l$ and the frequency bin $b_i$ is given by the component distribution $\boldsymbol{\phi}_k$:

$$p(c_i = w_i - l, b_i|k_i = k, l_i = l, \boldsymbol{\phi}_k) = \phi_{kc_ib_i} \tag{8.9}$$

The likelihood of the pair $k, l$ conditioned on $\eta$ and the other source indicators $\boldsymbol{k}_{-i}$ and time offsets $\boldsymbol{l}_{-i}$ is

$$\begin{aligned} & p(k_i = k, l_i = l | k_{-i}, l_{-i}, \eta) \\ = & \int_{\boldsymbol{\omega}} p(\boldsymbol{\omega} | k_{-i}, l_{-i}, \eta) \omega_{kl} d\boldsymbol{\omega} \\ = & \frac{n_{kl} + \eta}{N - 1 + KL\eta} \end{aligned} \tag{8.10}$$

Where $n_{kl}$ is the number of other quanta coming from source $k$ with time offset $l$. We can compute the integral in equation 8.10 analytically because the Dirichlet distribution is conjugate to the multinomial distribution.

Using equations 8.9 and 8.10, equation 8.8 becomes:

$$p(k_i = k, l_i = l | w_i, b_i, \boldsymbol{k}_{-i}, \boldsymbol{l}_{-i}, \boldsymbol{\phi}, \eta) \propto \phi_{kc_i b_i} \frac{n_{kl} + \eta}{N - 1 + KL\eta} \tag{8.11}$$

We repeatedly resample the source indicator $k_i$ and time offset $l_i$ for each observed quantum $i$ conditioned on the other indicators $\boldsymbol{k}_{-i}$ and $\boldsymbol{l}_{-i}$ until 20 iterations have gone by without the posterior likelihood $p(\boldsymbol{k}, \boldsymbol{l} | \boldsymbol{w}, \boldsymbol{b}, \eta, \boldsymbol{\phi})$ yielding a new maximum. At this point we assume that the Gibbs sampler has converged and that we have found a set of values for $\boldsymbol{k}$ and $\boldsymbol{l}$ that is likely conditioned on the data.

Once we have drawn values from the posterior for $\boldsymbol{k}$ and $\boldsymbol{l}$, we compute the MAP estimate $\hat{\boldsymbol{\omega}}$ of the joint distribution over sources and times $\boldsymbol{\omega}$ conditioned on $\boldsymbol{k}$, $\boldsymbol{l}$, and the hyperparameter $\eta$. Since the prior on $\boldsymbol{\omega}$ is a Dirichlet distribution, the MAP estimate $\hat{\boldsymbol{\omega}}$ of $\boldsymbol{\omega} | \boldsymbol{k}, \boldsymbol{l}, \eta$ is given by:

$$\hat{\omega}_{kl} \propto \max(0, n_{kl} + \eta - 1) \tag{8.12}$$

Here $n_{kl}$ is the total number of observed quanta that came from source $k$ at time $l$.

### 8.3.2 Sonifying the MAP Estimate

By sonifying $\hat{\boldsymbol{\omega}}$, the MAP estimate of $\boldsymbol{\omega}$, we can produce an approximate version of our input audio using only the short sources corresponding to the component distributions $\boldsymbol{\phi}$. $\hat{\omega}_{kl}$ gives the amplitude of source $k$ at time offset $l$, which corresponds to sample $S(l-1)$, where $S$ is the number of samples per window, and samples begin at sample 0. If we convolve each short input source $k$ by a signal $g$ such that

$$g(t, k) = \begin{cases} 0 & \text{if } \operatorname{mod}(t, S) \neq 0 \\ \hat{\omega}_{k, \frac{t}{S}+1} & \text{if } \operatorname{mod}(t, S) = 0 \end{cases} \tag{8.13}$$

and add the result for each source, we obtain a signal whose spectrogram approximates the spectrogram of the target. Figure 8.2 shows an example of the final result of this process.

| Sound Source | K | Sample Length | AC/DC | | Young MC | |
|---|---|---|---|---|---|---|
| | | | Error | $\eta$ | Error | $\eta$ |
| Noise | N/A | N/A | 0.6395 | N/A | 0.6265 | N/A |
| Ramones | 100 | 116 ms | 0.3844 | 0.004569 | 0.4039 | 0.001979 |
| Ramones | 200 | 116 ms | 0.3787 | 0.002386 | 0.4100 | 0.003376 |
| AC/DC | 100 | 116 ms | 0.3455 | 0.005579 | 0.3821 | 0.003689 |
| AC/DC | 200 | 116 ms | 0.3349 | 0.002382 | 0.3841 | 0.001939 |
| MC Hammer | 100 | 116 ms | 0.3838 | 0.005017 | 0.3753 | 0.003875 |
| MC Hammer | 200 | 116 ms | 0.3740 | 0.002993 | 0.3732 | 0.002499 |
| TIMIT | 100 | 464 ms | 0.5898 | 0.004742 | 0.6102 | 0.003262 |
| TIMIT | 200 | 464 ms | 0.5275 | 0.002097 | 0.6110 | 0.001796 |

Table 8.1: Errors obtained by our approach when trying to match songs by AC/DC and Young MC using various sets of sound sources, and the learned values of the hyperparameter $\eta$. In all cases our method outperforms a baseline of white noise. Note that lower errors do not necessarily translate to a more aesthetically interesting result.

### 8.3.3 Resampling $\eta$

$\eta$ controls the sparseness of our joint distribution $\boldsymbol{\omega}$ over times and sources. Rather than specify $\eta$ a priori, we place a gamma prior on $\eta$ and adapt the hyperparameter sampling technique in [30] to resample $\eta$ each iteration.

## 8.4 Evaluation

Ultimately the effectiveness of our approach should be evaluated qualitatively. Sound examples generated by the method described in this chapter are available at `http://www.cs.princeton.edu/~mdhoffma/icmc2009`.

We also performed a quantitative evaluation of our approach. We tested SIMM's ability to find an arrangement of the given components $\boldsymbol{\phi}$ to match the target spectrogram $\hat{\boldsymbol{y}}$ by computing and sonifying a MAP estimate $\hat{\boldsymbol{\omega}}$ of the joint distribution over times and components as described in section 8.3, then comparing the sum of the magnitudes of the differences between the normalized spectrograms of the target sound and resynthesized sound. Let $\hat{\boldsymbol{z}}$ be the normalized spectrogram of the resynthesized sound. Our error metric is

$$err = 0.5 \sum_{b=1}^{B} \sum_{w=1}^{W} |\hat{z}_{wb} - \hat{y}_{wb}| \tag{8.14}$$

which ranges between 0.0 (perfect agreement between the spectrograms) and 1.0 (no overlap between the spectrograms).

Table 8.4 presents the errors obtained by our approach when trying to match 23.2 second clips (1000 512-sample windows at 22.05 KHz) from the songs "Dirty Deeds Done Dirt Cheap" by AC/DC and "Bust a Move" by Young MC, using samples selected at random from the songs "Dirty Deeds Done Dirt Cheap," "Blitzkrieg Bop" by the Ramones,

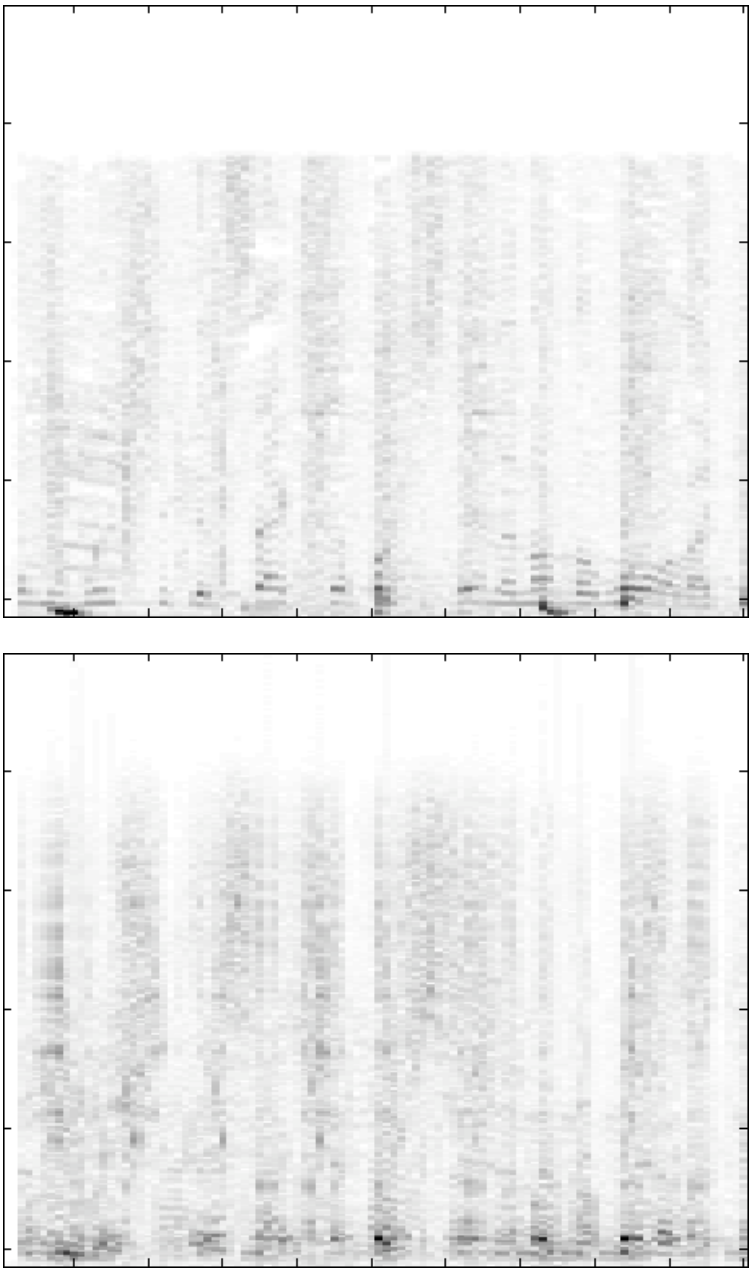

Figure 8.2: Top: Spectrogram of 2.3 seconds of Young MC's "Bust a Move." Bottom: Spectrogram of 2.3 seconds of the same song reconstructed from spoken words from the TIMIT corpus using our SIMM model.

and "U Can't Touch This" by MC Hammer. We also used words spoken by various speakers from the TIMIT corpus of recorded speech as source samples. Samples from similar songs tend to produce lower errors, whereas the model had trouble reproducing music using spoken words. The speech samples produce a quantitatively weaker match to the target audio, but the "automatic a cappella" effect of trying to reproduce songs using speech proved aesthetically interesting.

All output sounds are available at the URL given above.

## 8.5 Transcription

Besides creating novel sounds, the method described above can be adapted to produce rough transcriptions of polyphonic music. For example, if a set of piano sample spectrograms is used for $\boldsymbol{\phi}$ and the target audio spectrogram $\hat{\boldsymbol{y}}$ is a recording of piano music, then the MAP estimate of $\omega_{kl}$ will give an estimate of how forcefully the $k$th sample was activated at time $l$. By thresholding the MAP estimate of $\boldsymbol{\omega}$, we can predict whether or not a note was struck at any particular time.

To evaluate this transcription technique, we analyzed two pieces of music from the Midi-Aligned Piano Sounds (MAPS) database of piano music [29]—Bach's Prelude in C Minor, and Beethoven's "Pathetique" piano sonata. Both were synthesized from midi scores provided as part of the MAPS database. The prelude was synthesized using a Yamaha Disklavier, and the piano sonata was synthesized using Bosendorfer 290 Imperial model from Native Instruments' "Akoustik Piano" sound library. All recordings had a sampling rate of 44.1KHz. Each piece was analyzed once using a set of isolated samples from the same instrument or synthesizer used to produce the mixed recording, and once using a set of isolated samples from the other instrument/synthesizer. We used 2048-sample (46.4ms) Hann windows to compute the spectrograms. As in the experiments above, we ran the collapsed Gibbs sampler until 20 iterations went by without finding a new maximum of the joint likelihood, and then computed a MAP estimate of $\boldsymbol{\omega}$ from the last sample of $\boldsymbol{k}$ and $\boldsymbol{l}$. A threshold $\tau$ was then applied to the elements of $\boldsymbol{\omega}$ to yield a binary transcription $\boldsymbol{\psi}$ of the recording—if $omega_{kl} > \tau$, then $\psi_{kl} = 1$, otherwise $\psi_{kl} = 0$.

The binary transcriptions $\boldsymbol{\psi}$ were compared with the ground truth midi files used to synthesize the recordings. A positive label $\psi_{kl} = 1$ was considered a true positive if the note $k$ was played within 69.7ms of the center of the $l$th window—that is, if it appeared in a window from $l - 1$ to $l + 1$. Following [73], we used the following metric to score each transcription:

$$\text{acc} \triangleq \frac{\text{\# true positives}}{\text{\# true positives} + \text{\# false positives} + \text{\# false negatives}}. \tag{8.15}$$

This metric defines accuracy as the number of true notes found divided by the sum of the total number of true notes plus the number of false alarms, i.e. notes in the transcription that did not appear in the recording.

Table 8.5 shows the performance of our approach for each sample set/recording pair with the best threshold $\tau$. Clearly, using a set of samples that matches the timbre of the instrument used to generate the recording being analyzed results in much better transcription performance.

Figure 8.3 summarizes the performance of our approach for various thresholds $\tau$. Performance depends strongly on using a threshold within an appropriate range for the piece, but the best threshold does not seem to be very sensitive to the choice of sample set used to analyze the recording. The Bach piece (recorded using the "Bosendorfer" sample set) de-

| Sample Set | Transcription Accuracy | |
|---|---|---|
| | Beethoven (Disklavier) | Bach (Bosendorfer) |
| Disklavier | 0.733 | 0.345 |
| Bosendorfer | 0.358 | 0.808 |

Table 8.2: Accuracies obtained by the SIMM model for two sample sets and two recordings, for the best setting of the threshold $\tau$.

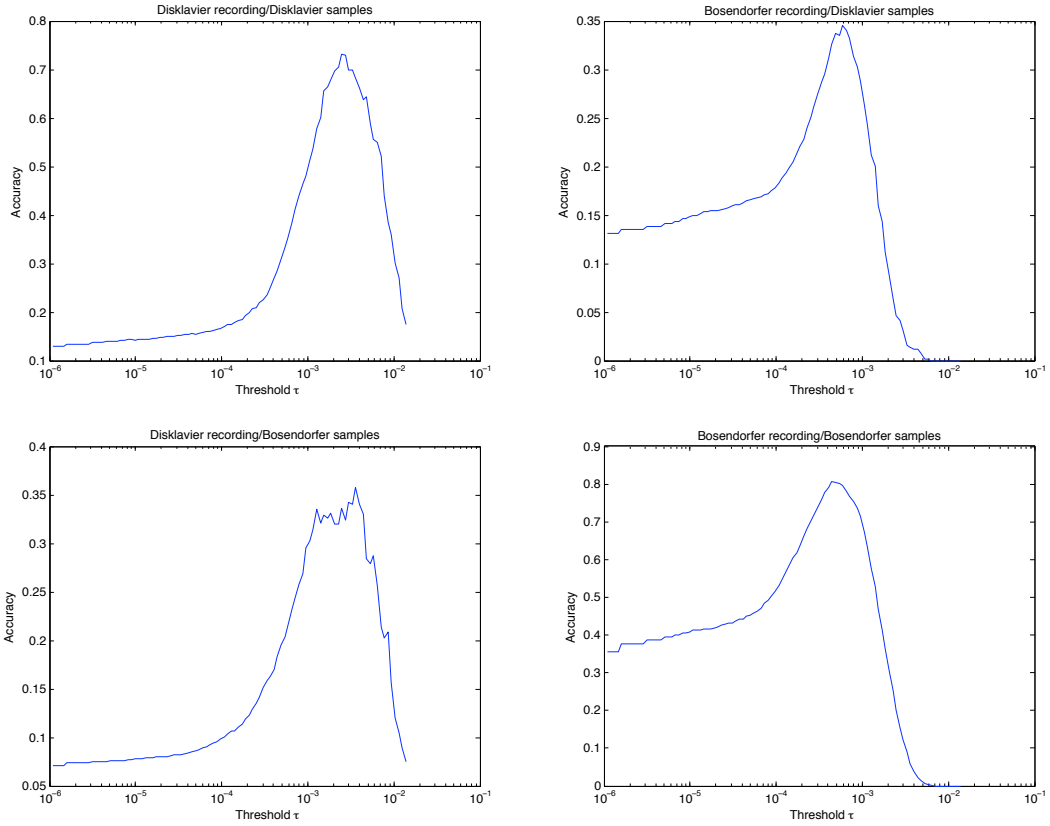


Figure 8.3: Plots of transcription accuracy versus binarization threshold $\tau$ for two recordings analyzed using two sets of piano samples.

mands a lower threshold than the Beethoven piece (recorded using the "Disklavier" sample set), which can be attributed to the greater number of notes per second in the Bach piece.

These results are not bad, considering how little tuning and postprocessing went into the transcription procedure. However, they are certainly less impressive than those obtained by state-of-the-art methods designed to address the problem of polyphonic transcription such as those proposed in [42, 98]. Any number of model improvements are possible that would improve transcription performance. For example, we could explicitly model the on-off sparsity that should be present in the activation matrix via a spike-and-slab prior [66], or we might build a model that allows samples to adapt to the sounds actually present

in a recording. Such elaborations would allows us to sidestep the problems of choosing a threshold $\tau$, and alleviate the errors caused by using a sample set not actually used to generate the recording being analyzed.

## 8.6 Discussion

We presented a new audio mosaicing approach that attempts to match the spectrogram of a target sound by combining a vocabulary of shorter sounds at different time offsets and amplitudes. We introduced the SIMM model and showed how to use it to find a set of time offsets and amplitudes that will result in an output sound that matches the target sound. We also explored the applicability of the SIMM model to the problem of polyphonic transcription.

# Chapter 9

# Conclusions and Future Work

## 9.1 Contributions

In this dissertation, we presented several new techniques making use of the framework of probabilistic graphical modeling to address problems in computational audio analysis:

**Timbral similarity using the Hierarchical Dirichlet Process:** Using the Hierarchical Dirichlet Process (HDP) to discover a shared representation of the distributions of feature vectors within multiple songs, we were able to compare those songs with greater accuracy and lower computational cost than previous methods, while eliminating the problem of "hubs," i.e. songs that are wrongly evaluated to be similar to all other songs.

**Automatic tagging with the Codeword Bernoulli Average model:** We derived the Codeword Bernoulli Average (CBA) model, a simple probabilistic model that predicts the probability that a tag will apply to a song based on a vector-quantized representation of that song's feature data. CBA was able to achieve state-of-the-art performance on both annotation and retrieval tasks compared with previous methods, despite its comparative simplicity and the low computational complexity of inferring its parameters.

**Latent source discovery with the Shift-Invariant Hierarchical Dirichlet Process:** We derived the Shift-Invariant HDP (SI-HDP), an extension of the HDP that can discover the time-varying latent sources present in a set of audio recordings, when they occur, and how many sources are needed to explain the data. The ability of the SI-HDP to automatically choose how many latent sources to use addresses is particularly important to this application, where it is difficult to know a priori how many sources will appear in a piece of music.

**Latent source discovery with Gamma Process Nonnegative Matrix Factorization:** We derived the Gamma Process Nonnegative Matrix Factorization (GaP-NMF) model, which addresses certain theoretical limitations with the SI-HDP's signal model and allows for more principled Bayesian inference while retaining the ability to choose how many latent sources are needed to explain an audio spectrogram. In order to derive a variational

inference algorithm for GaP-NMF, we had to devise new techniques that can be used to derive variational inference algorithms for other non-conjugate models.

**Hidden Markov Models for audio synthesis:** We explored the potential of Hidden Markov Models (HMMs) to synthesize musical audio. The results are suggestive both of the power of generative models to produce audio and of the limitations of HMMs trained on features extracted from mixed audio.

**Bayesian spectral matching:** The SI-HDP was adapted to the problem of layering short samples to create an audio signal whose spectrogram resembles that of a target audio signal. The resulting model can be applied to create novel transformations of existing recordings, to find ways to mimic recordings by arranging a limited set of instruments, and to produce rough polyphonic transcriptions.

## 9.2 Future Work

One of the most powerful features of the graphical modeling framework is it extensibility, in particular to hierarchical applications. Any of the models presented in this thesis could be extended and made more useful by relaxing oversimplistic assumptions or incorporating new sources of information. Below we present several ideas for what such extensions might look like. Some of these ideas we are currently pursuing actively, others are more speculative.

### 9.2.1 Beyond Similarity: New Applications of Mixed-Membership Mixture Modeling

In chapter 3, we presented an application of the Hierarchical Dirichlet Process (HDP) with a multivariate Gaussian observation model to the problem of estimating the timbral similarity of two songs. The HDP learned to represent songs as a set of mixture weight vectors over a set of shared mixture components, and this representation turned out to be useful for estimating the similarity of two songs. This sort of higher-level representation could allow us to leverage techniques from the text modeling literature for music analysis problems.

One classic problem in the area of Music Information Retrieval (MIR) is genre classification—given a set of recordings labeled with genre information, the goal is to predict the genres of unlabeled recordings from the audio signal [91]. To address this problem, we could consider an approach like that taken in supervised topic modeling [18], where the topic labels given to each word in a document are replaced with cluster labels given to each feature vector in a song. Adapting the model in [18], we might assume a model like

$$\begin{aligned} \boldsymbol{\phi}_k \sim H; \quad \boldsymbol{\pi}_j \sim \text{Dirichlet}(\boldsymbol{\alpha}); \quad z_{ji} \sim \text{Multinomial}(\boldsymbol{\pi}_j); \quad \boldsymbol{y}_{ji} \sim f(\boldsymbol{\phi}_{z_{ji}}); \\ \boldsymbol{\eta}_{ck} \sim \mathcal{N}(0, \sigma^2); \quad p(g_j = c) \propto \exp\{\textstyle\sum_k \eta_{ck} \bar{z}_{jk}\}, \end{aligned} \tag{9.1}$$

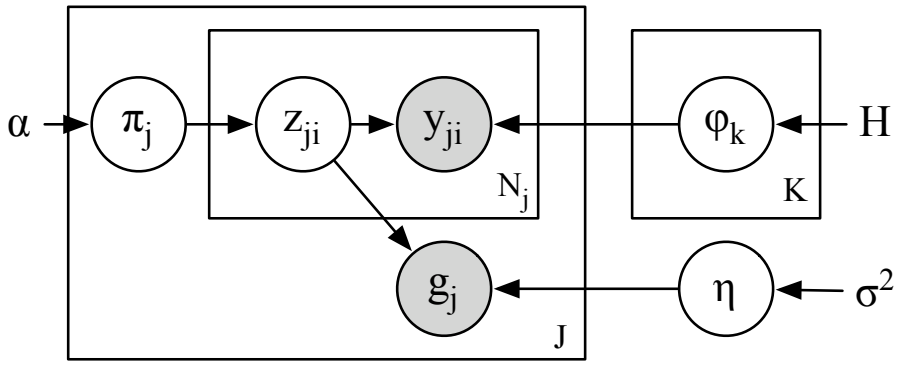


Figure 9.1: Graphical model for the genre classification model proposed in equation 9.1.

where $g_j = c$ denotes song $j$ as belonging to genre $c$, and $\bar{z}_{jk} \triangleq \frac{1}{N_j} \sum_{i=1}^{N_j} \mathbb{I}[z_{ji} = k]$ is the normalized counts vector of how many times cluster $k$ is associated with a feature vector in song $j$. $\boldsymbol{\eta}$ is a matrix of weights for a logistic regression model whose covariates are the hidden matrix $\bar{\boldsymbol{z}}$ and whose responses are the genre labels $\boldsymbol{g}$. The graphical model for this generative process is given in figure 9.1.

Posterior inference reveals

1. a set of clusters parameterized by $\boldsymbol{\phi}$ capturing the latent structure of the training songs' feature data and

2. a set of weights $\boldsymbol{\eta}$ that describe the relationship between the clusters that a song's feature vectors fall into and the genre label assigned to that song by a listener.

To label a new song $a$ whose genre label $g_a$ we are not given, we examine the posterior $p(g_a|\boldsymbol{y}_a, \boldsymbol{\alpha}, \boldsymbol{\eta}, \boldsymbol{\phi}) = \sum_{\boldsymbol{z}_a} p(g_a|\boldsymbol{z}_a, \boldsymbol{\eta}) p(\boldsymbol{z}_a|\boldsymbol{y}_a, \boldsymbol{\alpha}, \boldsymbol{\phi})$. By learning the clusters and weights simultaneously, the model may be able to discover representations of feature data that are tuned to be useful for solving the genre classification problem.

Other topic models designed for text could be adapted to address problems in audio analysis via a similar approach. For example, the influence model presented in [38] could be adapted using a similar strategy to determine what songs in a corpus were influential on what other songs[1]. Another interesting research direction would be to use this mixed-membership representation to combine content-based and collaborative filtering methods for music recommendation—a heuristic model presented in [97] suggests that such an approach may be promising.

### 9.2.2 Fully Generative Codeword Bernoulli Average

The Codeword Bernoulli Average (CBA) model presented in chapter 4 analyzes songs by first finding a vector quantized representation of a set of continuous-valued feature vectors, and then analyzing the relationship between the fixed "codeword" counts for each song and

[1]Strictly speaking, this model cannot guarantee that the relationships it finds are causal—it may be more accurate to say that it would find what songs caught onto emerging trends before any other songs in the corpus.

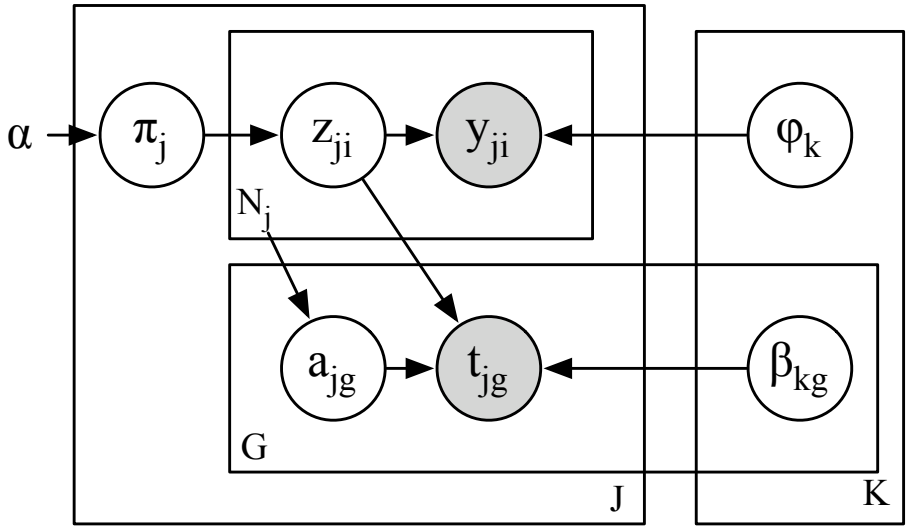


Figure 9.2: Graphical model for the autotagging model proposed in equation 9.2.

that song's tag labels. An alternative approach would be to build a model that combines the clustering and tag modeling steps, again making use of the mixed-membership clustering paradigm. One such model is given by the following generative process:

$$\begin{aligned} \boldsymbol{\pi}_j \sim \text{Dirichlet}(\boldsymbol{\alpha}); \quad z_{ji} \sim \text{Multinomial}(\boldsymbol{\pi}_j); \quad \boldsymbol{y}_{ji} \sim f(\boldsymbol{\phi}_{z_{ji}}); \\ a_{jg} \sim \text{Uniform}(1, \ldots, N_j); \quad t_{jg} \sim \text{Multinomial}(\boldsymbol{\beta}_{a_{jg}g}), \end{aligned} \tag{9.2}$$

where $\boldsymbol{y}_{ji}$ is the $i$th feature vector in song $j$ and $t_{jg}$ is the binary label denoting whether or not tag $g$ applies to song $j$. The graphical model for this generative process is given in figure 9.2. This model is very closely related to the Correspondance Latent Dirichlet Allocation (Corr-LDA) annotation model in [15]. The main difference is that this process models both positive and negative annotations via a Bernoulli distribution, whereas Corr-LDA models only positive annotations via a multinomial distribution.

As in the model defined in equation 9.1, jointly fitting $\boldsymbol{\phi}$ and $\boldsymbol{\beta}$ allows the clusters found by the model to be tuned to be useful for solving the autotagging problem. Note that if $f(\boldsymbol{y}; \boldsymbol{\phi}) = \mathcal{N}(\boldsymbol{\phi}, \epsilon)$, then as $\epsilon \to 0$ we recover the original CBA model, since the mixture model assumed to have generated $\boldsymbol{y}$ reduces to k-means clustering.

### 9.2.3 Pitch-Invariant and Hierarchical GaP-NMF

A limitation of the latent source decompositions presented in chapters 5 and 6 is that they assume total independence between all latent components, even components that correspond to different notes from the same instrument. This may lead to worse performance compared to a model that shared statistical strength between multiple latent components from the same instrument. Furthermore, if we want to isolate or suppress all of the energy associated with a particular instrument across all notes, the only option is to do a post-hoc analysis (by hand or automatically) to partition the components into one instrument class or another.

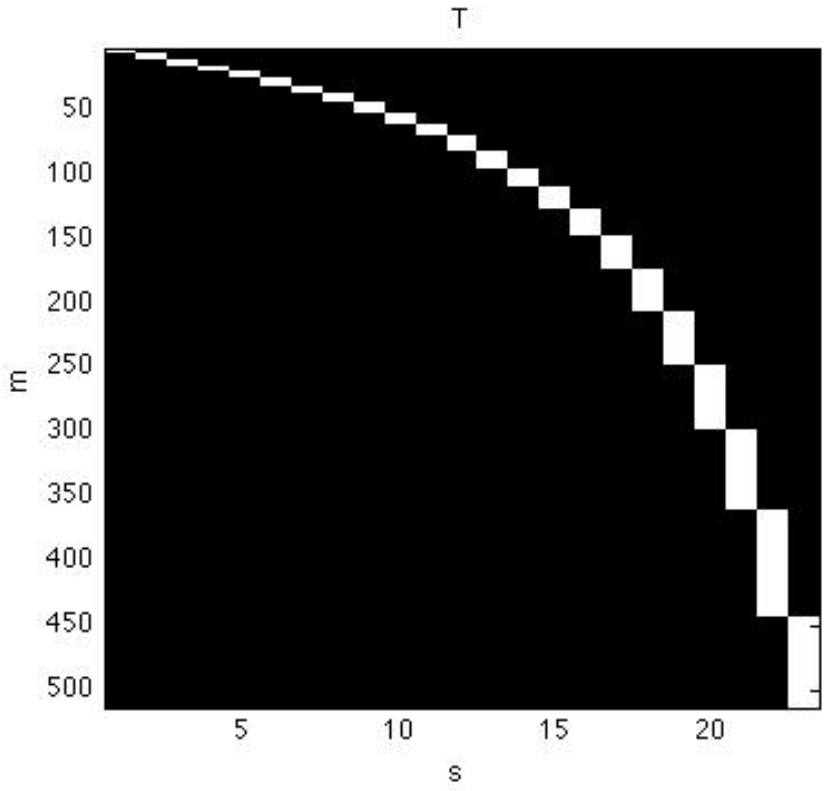


Figure 9.3: A binary subband matrix $\boldsymbol{T}$ with bandwidths determined by the Bark perceptual scale.

One way of addressing this problem might be to incorporate pitch-invariant clustering into the GaP-NMF model. Recall that each element $W_{mk}$ of the matrix $\boldsymbol{W}$ gives the average power at frequency bin $m$ of latent component $k$. The goal would be to cluster the latent components given by the columns of the matrix $\boldsymbol{W}$ in a way that considers their coarse spectral shape (which will be roughly the same for sounds with similar timbre across multiple pitches) but not their fine spectral details (which are strongly affected by pitch). A first step towards doing this is to decompose each column $\boldsymbol{W}_k$ into a set of variables $\boldsymbol{V}_k$ and $\boldsymbol{\phi}_k$ so that $W_{mk} = \phi_{mk} \sum_s T_{ms} V_{sk}$, where every element of $\boldsymbol{V}$ and $\boldsymbol{\phi}$ are nonnegative, and $\sum_m T_{ms}\phi_{mk} = 1$ for any $s \in \{1, \ldots, S\}$. We construct $\boldsymbol{T}$ to be a binary matrix whose columns define subbands of the spectrum, such as the one shown in figure 9.3. The matrix product $\boldsymbol{T}\boldsymbol{V}_k$ gives the coarse spectral shape of $\boldsymbol{W}_k$. Each subvector $\boldsymbol{\phi}_{\{m|T_{ms}=1\},k}$ sums to one, and determines how the energy in the coarse spectrum $\boldsymbol{T}\boldsymbol{V}_k$ is distributed.

We can replace the symmetric gamma prior

$$W_{mk} \sim \text{Gamma}(a, a) \tag{9.3}$$

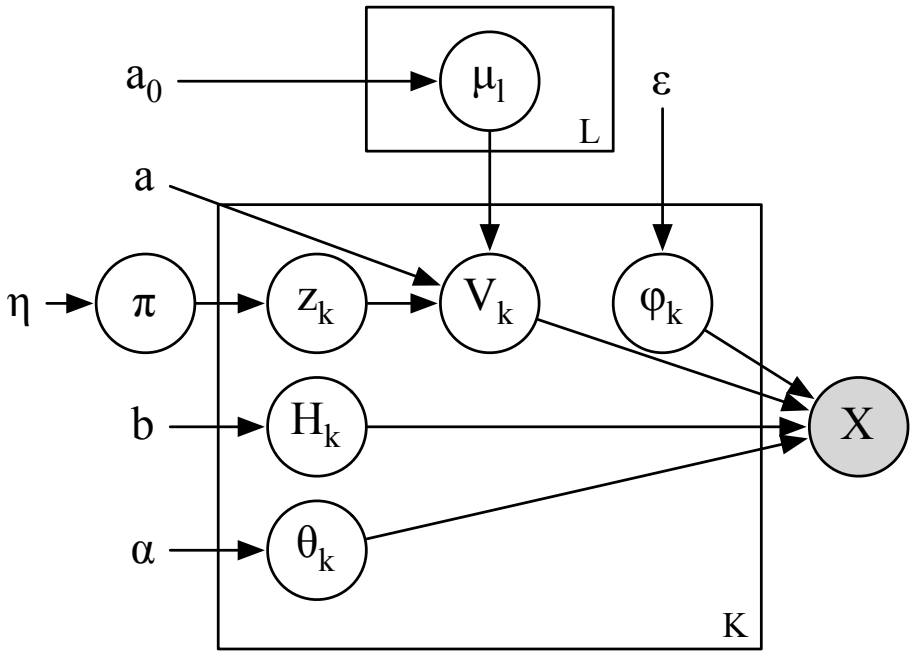


Figure 9.4: Graphical model for a pitch-invariant GaP-NMF model.

used in GaP-NMF, which assumes that all elements $W_{mk}$ are independent and identically distributed, with a generative process of the form

$$\boldsymbol{\pi} \sim \text{Dirichlet}(\eta/L, \ldots, \eta/L); \quad z_k \sim \text{Multinomial}(\boldsymbol{\pi}); \quad \mu_{sl} \sim \text{Gamma}(a_0, a_0); \quad (9.4)$$
$$V_{sk} \sim \text{Gamma}(a, a\mu_{sz_k}); \quad \boldsymbol{\phi}_{\{m|T_{ms}=1\},k} \sim \text{Dirichlet}(\epsilon, \ldots, \epsilon); \quad W_{mk} = \phi_{mk} \sum_s T_{ms} V_{sk}.$$

The hidden variable $z_k$ indicates which cluster the $k$th latent component belongs to, and the hidden parameter $\boldsymbol{\mu}_l$ gives the reciprocal of the expected value of $\boldsymbol{V}_k$ if $z_k = l$. Since only $\boldsymbol{V}_k$, the coarse shape of $\boldsymbol{W}_k$, is affected by $z_k$, the partitioning given by the hidden $\boldsymbol{z}$ variables is largely invariant to pitch. Furthermore, the columns of $\boldsymbol{W}_k$ are tied under this model through the parameter $\boldsymbol{\mu}$ (as can be seen in the graphical model in figure 9.4), allowing the latent components to share timbral information.

Deriving a variational inference algorithm for this model is nontrivial, but not impossible. Early informal results suggest that the unsupervised partitionings of latent components produced by this model do tend to correspond to groups of notes from the same instruments.

The model in equation 9.4 could be further extended to the hierarchical setting where we want to analyze many songs at once. In the simplest case, where we would just assume that $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$ are shared across all songs, variational inference can proceed by repeatedly updating an independent variational posterior $q(\boldsymbol{W}_j, \boldsymbol{H}_j, \boldsymbol{\theta}_j)$ for each song $j$, then updating the top-level variational posterior $q(\boldsymbol{\mu}, \boldsymbol{\pi})$ over the globally shared parameters. That the global representation is pitch-invariant would sidestep an issue with the SI-HDP, namely that two songs that include the same instrument but are in different keys will not use the same set of latent components, making it more difficult to compare them. The graphi-

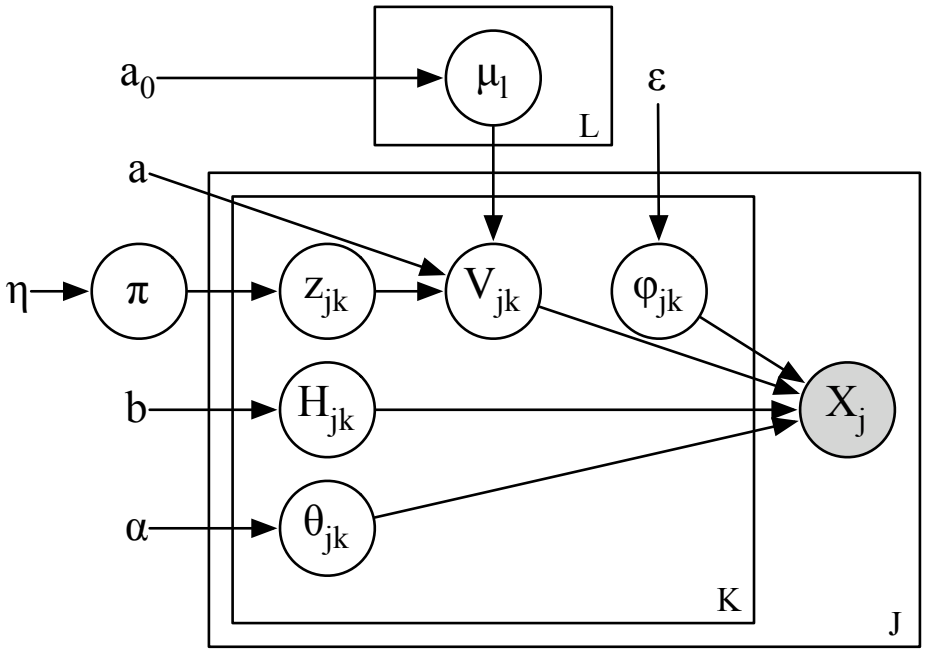


Figure 9.5: Graphical model for a hierarchical pitch-invariant GaP-NMF model.

cal model for this simple hierarchical extension is shown in figure 9.5. A more elaborate version might incorporate a mixed-membership mixture model like the HDP.

### 9.2.4 Temporal Modeling of Latent Source Activations

Just as we can think about relaxing the independence assumptions on the matrix of latent components $\boldsymbol{W}$ in GaP-NMF, we can also think about more elaborate modeling of the matrix of time-varying activations $\boldsymbol{H}$. In the simplest case, this might involve assuming a Markov chain of the form

$$H_{kn} \sim \text{Gamma}(b, bH_{k,n-1}^{-1}), \tag{9.5}$$

which is essentially the approach taken in [32, 93]. This modeling choice says that $\mathbb{E}[H_{kn}] = H_{k,n-1}$, which embodies the intuition that a component being loud at time $n-1$ makes it more likely to be loud at time $n$.

More interesting approaches are also possible. In general, we can consider any prior of the form

$$H_{kn} \sim \text{Gamma}(b, bf(\boldsymbol{H}_{1:n-1})^{-1}), \tag{9.6}$$

where $\boldsymbol{H}_{1:n-1}$ denotes the first $n-1$ columns of $\boldsymbol{H}$, so long as $f$ is a linear additive function of its arguments. A variational inference scheme can be derived that accomodates any such prior using the tricks in chapter 6.

For example, if we knew that the tempo of the song was such that there is a beat every $T$ frames, we might choose a prior like

$$H_{kn} \sim \text{Gamma}(b, \tfrac{1}{3}b(\textstyle\sum_{i=1}^{3} H_{k,n-iT})^{-1}). \tag{9.7}$$

This prior says that the expected activation of a component $k$ at a particular time $n$ is the average activation of that component at the corresponding time over the last three beats.

Any number of further elaborations are possible. We could put a prior on $T$ and learn it as well. Instead of using the constant weight $\frac{1}{3}$, a vector of weights could be learned. Components could be allowed to influence each other's activation levels. The parameters to the prior could be shared across multiple songs. The decision of how to structure the prior on $\boldsymbol{H}$ depends on what information the modeler wants to discover, and to what end.

One possible motivation for incorporating more sophisticated temporal modeling into a latent source decomposition is that it may allow us to do better data-driven audio synthesis. A generative hierarchical latent source model sophisticated enough to learn how audio sources fit together to make a song might be able to learn to produce audio of real aesthetic interest.

## 9.3 Conclusions

In this dissertation we have presented a number of graphical models designed to address specific problems in the analysis of musical audio. We believe that the successes of these models underscore the effectiveness of the basic approach outlined in the introduction:

1. posit a parametric stochastic process that explains the most important aspects of the data,

2. derive and apply an algorithm to infer the parameters to that process from observed data,

3. use the model and inferred parameters to address the problem.

The previous section illustrates two core benefits of the graphical modeling framework beyond its effectiveness at solving specific problems: extensibility and modularity. Most of the proposed extensions above came about as a result of phrasing the inadequacy of a previous model for a particular task in terms of overly simplistic modeling assumptions. For example, GaP-NMF cannot explicitly tell us anything about what components are derived from the same instrument, since it assumes that all components are independent and identically distributed. By replacing this assumption with a more elaborate (though still arguably simplistic) model that takes into account the relationship between Fourier spectra and the percept of timbre, we can extract additional useful information from our data. The modularity inherent in the graphical modeling formalism allows us to relax our models' assumptions piecemeal without having reevaluate our entire algorithmic strategy.

Another attractive feature of the probabilistic modeling framework is that it can be used not just to analyze but also to synthesize audio by sampling from a model's generative process. Perhaps someday someone will derive a generative model that will infer so much about the structure and content of a large corpus of songs that it can learn to stochastically "compose" new recordings that human beings will want to listen to over and over. Until then, there remain many problems in the computational analysis and synthesis of music that can be profitably addressed by devising and extending probabilistic graphical models.

# Appendix A

# Inference Procedures for the SIHDP

### A.0.1 Direct Assignment Gibbs Sampler

To draw from the posterior of the SIHDP, we adapt the direct assignment Gibbs sampler described in [85]. We integrate out all variables besides the component indicators $\boldsymbol{k}$, the time offsets $\boldsymbol{l}$, and the global component proportions $\boldsymbol{\beta}$, whose values comprise the state of the Markov chain.

*Resampling the component indicators $\boldsymbol{k}$ and time offsets $\boldsymbol{l}$:* First, we jointly resample each pair of variables $k_{ji}$, $l_{ji}$ indicating which component $k_{ji}$ at what time offset $l_{ji}$ generated observation $i$ in song $j$, conditioned on the values of all other indicator variables $\boldsymbol{k}_{-ji}$, $\boldsymbol{l}_{-ji}$, the global component proportion weights $\boldsymbol{\beta}$, the observed data $\boldsymbol{y}$, the concentration parameter $\alpha$, and the prior on the mixture components $\boldsymbol{\phi}$ defined by $\epsilon$.

$$\begin{aligned} p(k_{ji}, l_{ji}|\boldsymbol{k}_{-ji}, \boldsymbol{l}_{-ji}, \boldsymbol{\beta}, \alpha, \epsilon, \boldsymbol{y}) \propto \\ p(y_{ji}|\boldsymbol{k}, \boldsymbol{l}, \boldsymbol{y}_{-ji}, \epsilon) p(k_{ji}, l_{ji}|\boldsymbol{k}_{-ji}, \boldsymbol{l}_{-ji}, \boldsymbol{\beta}, \alpha, \eta) = \\ p(y_{ji}|\boldsymbol{k}, \boldsymbol{l}, \boldsymbol{y}_{-ji}, \epsilon) p(l_{ji}|k_{ji}, \boldsymbol{l}_{-ji}, \eta) p(k_{ji}|\boldsymbol{k}_{-ji}, \boldsymbol{\beta}, \alpha) \end{aligned} \tag{A.1}$$

Define $n_{lkj}$ to be the number of observations in song $j$ coming from component $k$ with time offset $l$, excluding the observation we're currently resampling. Define $o_{cbjk}$ to be the number of observations in song $j$ coming from component $k$ with base time $c$ and frequency bin $b$, again excluding the current observation.

Given $l_{ji}$ and $y_{ji} = \{w_{ji}, b_{ji}\}$, we can calculate the base offset $c_{ji} = w_{ji} - l_{ji}$, and so the first term becomes:

$$\begin{aligned} p(y_{ji}|\boldsymbol{k}, \boldsymbol{l}, \boldsymbol{y}_{-ji}, \epsilon) = p(c_{ji}, b_{ji}|\boldsymbol{k}, \boldsymbol{l}, \boldsymbol{y}_{-ji}, \epsilon) \\ = \int_{\phi} p(c_{ji}, b_{ji}|\phi) p(\phi|\boldsymbol{c}_{-ji}, \boldsymbol{b}_{-ji}, \boldsymbol{k}, \boldsymbol{l}, \epsilon) d\phi \\ = (o_{c_{ji}b_{ji}jk_{ji}} + \epsilon)/(o_{\cdot\cdot jk_{ji}} + CB\epsilon) \end{aligned} \tag{A.2}$$

For a new component $k$, the predictive likelihood is a constant $\frac{1}{CB}$, since the prior on $\boldsymbol{\phi}$ is symmetric.

The marginal likelihood of the component indicator $k_{ji}$ conditioned on the other $\boldsymbol{k}_{j,-i}$ in the same song $j$ and on the global component proportions $\boldsymbol{\beta}$ is given by the Chinese

restaurant franchise:

$$p(k_{ji}|\boldsymbol{k}_{j,-i},\boldsymbol{\beta},\alpha) = \begin{cases} \frac{n_{\cdot kj}+\alpha\beta_k}{N_j+\alpha} & \text{if } k \in \{1,\ldots,K\} \\ \frac{\alpha\beta^{\text{new}}}{N_j+\alpha} & \text{if } k = k^{\text{new}} \end{cases} \tag{A.3}$$

Where $\beta^{\text{new}}$ is the global likelihood of choosing a component not currently associated with any observations:

$$\beta^{\text{new}} = 1 - \sum_{k=1}^{K} \beta_k \tag{A.4}$$

The likelihood of time offset $l_{ji}$ conditioned on the other $\boldsymbol{l}_{j,-i}$ and on $k_{ji}$ if $k_{ji} \in \{1,\ldots,K\}$ is given by:

$$\begin{aligned} p(l_{ji}|k_{ji},l_{-ji},\eta) &= \int_{\omega} p(l_{ji}|\omega)p(\omega|\boldsymbol{l}_{j,-i},\eta)d\omega \\ &= \frac{n_{l_{ji}k_{ji}j}+\eta}{n_{\cdot k_{ji}j}+\eta L} \end{aligned} \tag{A.5}$$

The predictive likelihood for a new component $k^{\text{new}}$ is a constant $\frac{1}{L}$, since the prior on $\boldsymbol{\omega}$ is symmetric.

Therefore, the joint posterior likelihood of $k_{ji}$ and $l_{ji}$ for a given observation $y_{ji} = \{c_{ji}+l_{ji}, b_{ji}\}$ conditioned on $\boldsymbol{k}_{-ji}$, $\boldsymbol{l}_{-ji}$, $\boldsymbol{y}_{-ji}$, $\boldsymbol{\beta}$, $\alpha$, and $\epsilon$ is:

$$\begin{aligned} &p(k_{ji}=k, l_{ji}=l|\boldsymbol{k}_{-ji},\boldsymbol{l}_{-ji},\boldsymbol{\beta},\epsilon,\alpha,\boldsymbol{y}) \\ &\propto \frac{(o_{c_{ji}b_{ji}jk}+\epsilon)(n_{\cdot kj}+\alpha\beta_k)(n_{lkj}+\eta)}{(o_{\cdot\cdot jk}+CB\epsilon)(n_{\cdot\cdot j}+\alpha)(n_{\cdot kj}+\eta L)} \end{aligned} \tag{A.6}$$

for $k \in \{1,\ldots,K\}$. For $k = k^{\text{new}}$,

$$p(k_{ji}=k^{\text{new}}, l_{ji}=l|\boldsymbol{\beta},\alpha,N_j) \propto \frac{\alpha\beta^{\text{new}}}{CBL(N_j-1+\alpha)} \tag{A.7}$$

If $n_{\cdot k\cdot} = 0$ for some component $k$ at some point during resampling, then that component may be eliminated from future considerations.

*Creating a new mixture component:* If $k_{ji} = k^{\text{new}}$, then a new mixture component needs to be created. When this happens, we draw a stick-breaking weight $s \sim \text{beta}(1,\gamma)$, set $\beta_{k^{\text{new}}} = s\beta^{\text{new}}$ and then update $\beta^{\text{new}} := (1-s)\beta^{\text{new}}$, as in the direct assignment sampler for the HDP [85]. We choose the time offset $l_{ji}$ uniformly at random from the set of offsets $\{w_{ji}-C+1,\ldots,w_{ji}\}$ that are consistent with an observation at time $w_{ji}$.

*Resampling the global mixture proportions $\boldsymbol{\beta}$:* After the component indicators $\boldsymbol{k}$ and time offsets $\boldsymbol{l}$ have been resampled, we resample the global component proportions $\boldsymbol{\beta}|\boldsymbol{k},\alpha,\gamma$ by simulating the Chinese Restaurant Franchise. Let $m_{jk}$ be the number of tables in restaurant $j$ eating dish $k$. Then

$$\boldsymbol{\beta}|m,\gamma \sim \text{Dirichlet}(m_{\cdot 1},\ldots,m_{\cdot K},\gamma) \tag{A.8}$$

For each restaurant $j$ and dish $k$, draw $m_{jk}|\alpha, \beta, n_{\cdot kj}$ as follows[1]:

1. Set $m_{jk} = 0$
2. For i $\in \{0, \ldots, n_{\cdot kj} - 1\}$:
   (a) Increment $m_{jk}$ by $t_i \sim$ Bernoulli$(\frac{\alpha\beta_k}{\alpha\beta_k + i})$

Once $\boldsymbol{m}$ has been drawn for all $j, k$, redraw $\boldsymbol{\beta}$ according to equation A.8.

*Sampling the components $\boldsymbol{\phi}$:* We can also sample the latent components $\boldsymbol{\phi}$ instead of integrating them out—this slows convergence, but makes the distributed inference algorithm presented in the following section possible, which allows us to apply our model to larger datasets.

If we instantiate $\boldsymbol{\phi}$ rather than integrating it out, equation A.6 simplifies to:

$$\begin{aligned} p(k_{ji} = k, l_{ji} = l | \boldsymbol{k}_{-ji}, \boldsymbol{l}_{-ji}, \boldsymbol{\beta}, \epsilon, \alpha, \boldsymbol{y}) \\ \propto \phi_{c_{ji} b_{ji} k} \frac{(n_{\cdot kj} + \alpha\beta_k)(n_{lkj} + \eta)}{(n_{\cdot\cdot j} + \alpha)(n_{\cdot kj} + \eta L)} \end{aligned} \tag{A.9}$$

All other updates are the same as before.

To update $\boldsymbol{\phi}_k$, we can simply draw from its posterior conditioned on the indicator variables $\boldsymbol{k}, \boldsymbol{l}$, the observations $\boldsymbol{y}$, and the prior parameter $\epsilon$:

$$\boldsymbol{\phi}_k | \boldsymbol{k}, \boldsymbol{l}, \boldsymbol{y}, \epsilon \sim \text{Dirichlet}(o_{1,1,\cdot,k} + \epsilon, \ldots, o_{C,B,\cdot,k} + \epsilon) \tag{A.10}$$

*Resampling the hyperparameters $\alpha$ and $\gamma$:* We can resample the hyperparameters $\alpha$ and $\gamma$ in the same way as in the HDP.

### A.0.2 Distributed Inference

Resampling the component indicators $\boldsymbol{k}$ and time offsets $\boldsymbol{l}$ for each observation requires $O(CKN_{\cdot})$ operations per iteration. Say that our songs are all 2000 512-sample frames long (corresponding to 46 seconds at a sampling rate of 22050 Hz) and we choose $\nu = 1.0$ and $C = 20$ (corresponding to components lasting 460 ms). Then if our model discovers 200 latent components, resampling $\boldsymbol{k}$ and $\boldsymbol{l}$ will require billions of floating-point operations and memory accesses per song. This may lead to unacceptably long run times even for small datasets, particularly if the songs are heterogeneous and a larger number of components is needed to model them.

A solution to this problem is to split the work of resampling $\boldsymbol{k}$ and $\boldsymbol{l}$ across multiple processors, assigning one processor to deal with each song $j$. These indicator variables for each song are conditionally independent of those in all other songs given the global component proportions $\boldsymbol{\beta}$ and the components $\boldsymbol{\phi}$, so the only situation in which we have to do anything differently from the single-processor Gibbs sampler described in the previous section is when creating or eliminating components, since these actions affect the state of the global variables $\boldsymbol{\beta}$ and $\boldsymbol{\phi}$.

[1]Note that $n$ and $o$ here include all observations in all songs, unlike when we were redrawing $k$ and $l$.

We can put off eliminating components until after all $\boldsymbol{k}$ and $\boldsymbol{l}$ have been resampled. At that point, if a component $k$ has no observations associated with it then $p(\beta_k > 0 | \boldsymbol{k}, \alpha, \gamma)$ will be 0 and the component can be eliminated.

Creating a new component $k^{\text{new}}$ is more complicated, since creating a new component involves sampling $\beta_{k^{\text{new}}}$ and $\boldsymbol{\phi}_{k^{\text{new}}}$, which alters the global state of the Markov chain in ways that affect other groups. In [7], Asuncion et al. propose an approximate Gibbs sampler for the HDP that allows each process to create new components as usual, and then merges the component ID's across processors arbitrarily. This approach is not guaranteed to converge, and they report experimental results in which it converges to a final number of topics much more slowly than a single-threaded exact Gibbs sampler does.

We instead propose a method for allowing multiple processes to create new components without sacrificing consistency. Before resampling the indicator variables $\boldsymbol{k}$ and $\boldsymbol{l}$, we draw a set of $A$ global auxiliary components $\boldsymbol{\phi}_{K+1,\ldots,K+A}$ from their prior:

$$\boldsymbol{\phi}_{K+a} \sim \text{Dirichlet}(\epsilon, \ldots, \epsilon) \tag{A.11}$$

We also augment the global component proportions $\boldsymbol{\beta}$ with a series of $A$ additional weights partitioning the probability mass in $\beta^{\text{new}}$ using the stick-breaking process:

$$\begin{gathered} s_a \sim \text{Beta}(1, \gamma); \quad \beta_{K+1} = s_1 \beta^{\text{new}} \\ \beta_{K+a} = s_a (1 - s_{a-1}) \beta_{K+a-1}; \\ \hat{\beta}^{\text{new}} = (1 - s_A) \beta_A \end{gathered} \tag{A.12}$$

Effectively we have sampled from the prior an extra $A$ latent components not associated with any observations, and assigned them weights in $\boldsymbol{\beta}$ according to the stick-breaking process. If we include these auxiliary components when resampling $\boldsymbol{k}$ and $\boldsymbol{l}$, then the model has a set of $A$ new components to which it can assign observations without having to change any global variables. There is still a chance that a song will choose a component $\hat{k}^{\text{new}}$ for which we have not sampled a component $\boldsymbol{\phi}$, however:

$$p(k_{ji} = \hat{k}^{\text{new}}) \propto \frac{\alpha \hat{\beta}^{\text{new}}}{LB(N_j - 1 + \alpha)} \tag{A.13}$$

If the number of auxiliary components $A$ is chosen to be sufficiently large, $\hat{\beta}^{\text{new}}$ will be dramatically smaller than $\beta^{\text{new}}$, and so this will be a much less likely event than choosing a component $k \in \{K+1, \ldots, K+A\}$. It is important to choose a value for $A$ large enough that $\hat{k}^{\text{new}}$ is never chosen, since it is difficult to deal with this event in a principled way. We could simply abort this round of resampling $\boldsymbol{k}$ and $\boldsymbol{l}$, increase $A$, draw a new set of auxiliary variables and try again, but this could potentially introduce a bias that is hard to account for. In our experiments we chose a sufficiently large value for $A$ that $\hat{k}^{\text{new}}$ was never chosen.

If $A$ is large, a naïve approach introduces significant extra computation. Since there are no observations associated with the auxiliary components, however, we can sidestep this extra computation by efficiently precalculating the marginal probability of associating an observation with *any* component not yet associated with any observations. Denote this set

as $\boldsymbol{k}^{\text{new}} = \{K+1, \ldots, K+A, \hat{k}^{\text{new}}\}$.

$$\begin{aligned} & p(k_{ji} \in \boldsymbol{k}^{\text{new}} | y_{ji}, \boldsymbol{\beta}, \hat{\beta}^{\text{new}}, \boldsymbol{\phi}, \alpha, N_j) \\ \propto \quad & p(y_{ji} | k_{ji} \in \boldsymbol{k}^{\text{new}}, \boldsymbol{\phi}, \boldsymbol{\beta}, \hat{\beta}^{\text{new}}) \times \\ & p(k_{ji} \in \boldsymbol{k}^{\text{new}} | \beta^{\text{new}}, \alpha, N_j) \end{aligned} \tag{A.14}$$

The first term can be summarized as a weighted average of the auxiliary components $\boldsymbol{\phi}$ and the likelihood of an observation drawn from a $\boldsymbol{\phi}_{\hat{k}^{\text{new}}}$

$$\begin{aligned} & p(y_{ji} | k_{ji} \in \boldsymbol{k}^{\text{new}}, \boldsymbol{\phi}, \boldsymbol{\beta}, \hat{\beta}^{\text{new}}) \\ = \quad & \sum_{c=1}^{C} \left[ p\left(c, b_{ji} | k_{ji} \in \boldsymbol{k}^{\text{new}}, \boldsymbol{\phi}, \boldsymbol{\beta}, \hat{\beta}^{\text{new}}\right) \times \right. \\ & \left. p\left(l_{ji} = w_{ji} - c | k_{ji} \in \boldsymbol{k}^{\text{new}}\right) \right] \\ = \quad & \frac{1}{L\beta^{\text{new}}} \sum_{c=1}^{C} \left( \frac{\hat{\beta}^{\text{new}}}{CD} + \sum_{k=K+1}^{K+A} \phi_{cb_{ji}k} \beta_k \right) \end{aligned} \tag{A.15}$$

The second term is simply the prior likelihood of sitting at any empty table in the CRF:

$$p\left(k_{ji} \in \boldsymbol{k}^{\text{new}} | \beta^{\text{new}}, \alpha, N_j\right) = \frac{\alpha \beta^{\text{new}}}{N_j - 1 + \alpha} \tag{A.16}$$

Neither of these terms depend on the component indicators $\boldsymbol{k}$ or the time offsets $\boldsymbol{l}$, so they only need to be computed once for each possible frequency bin $b$ before resampling $\boldsymbol{k}$ and $\boldsymbol{l}$. Then, when resampling $k_{ji}$ we can efficiently sample whether or not $k_{ji} \in \boldsymbol{k}^{\text{new}}$. If $k_{ji} \notin \boldsymbol{k}^{\text{new}}$ (as will usually be the case) we can safely ignore all auxiliary variables.

A simpler version of this auxiliary variable method can also be applied to the original HDP formulation, as long as the global component proportions $\boldsymbol{\beta}$ and latent components $\boldsymbol{\phi}$ are sampled rather than integrated out, and the space of possible observations is discrete. Although this case is known to converge slowly, for some very large datasets this might be outweighed by the ability to deploy more computational resources.

# Appendix B

# Glossary of Abbreviations

This dissertation uses many abbreviations to refer to models, algorithms, and mathematical objects. This results in a document that is arguably more readable (and undeniably shorter), but also results in an "alphabet soup" that may be confusing if the reader forgets what a particular recurring jumble of characters refers to.

The glossary below is provided to allow the reader to quickly look up the meaning of these abbreviations without having to search for the place in the text where they were first defined.

**AP:** Average Precision. A measure summarizing retrieval quality.

**ARD:** Automatic Relevancy Determination. A technique for automatic order selection in probabilistic models.

**AROC/AUC:** Area under the Receiver-Operating Characteristic curve. A measure summarizing retrieval quality.

**BMSS:** Blind Monaural Source Separation. The problem of isolating invidiual sound sources from a mixed single-channel audio signal.

**BNP:** Bayesian NonParametric. Describes a class of Bayesian models for which the number of parameters to be inferred from data is not fixed a priori.

**BP:** Beta Process. A Bayesian nonparametric prior on binary matrices of infinite dimension.

**CAL500:** A corpus of human-labeled songs released by the Computer Audition Lab at the University of California, San Diego.

**CBA:** Codeword Bernoulli Average. A probabilistic model for automatically tagging songs based on their audio content.

**CRF:** Chinese Restaurant Franchise. A metaphor for the Hierarchical Dirichlet Process.

**CRP:** Chinese Restaurant Process. A metaphor for the Dirichlet Process.

**DAFX:** Conference on Digital Audio Effects.

**DFT:** Discrete Fourier Transform. Transforms a discrete signal from the time domain to the frequency domain.

**DP:** Dirichlet Process. A Bayesian nonparametric prior on partitions over an infinite set of possible labels.

**DPMM:** Dirichlet Process Mixture Model. A Bayesian nonparametric model that avoids the usual problem in mixture modeling of specifying model order a priori.

**EM:** Expectation-Maximization algorithm. An algorithm for maximum-likelihood parameter estimation in the presence of hidden data.

**EU-NMF:** Nonnegative Matrix Factorization with a EUclidean cost function.

**FBS:** Feature-Based Synthesis. A set of techniques for synthesizing audio characterized by a given vector of automatically extracted features.

**GEM:** Abbreviation for the names Griffiths, Engen, and McCloskey. Often used to denote the stick-breaking distribution over infinite vectors whose elements are nonnegative and sum to one.

**GIG:** Generalized Inverse-Gaussian distribution. A distribution over the nonnegative real line that generalizes not only the inverse-Gaussian but also the gamma and inverse-gamma distributions.

**GK:** Shorthand for the use of a K-component Gaussian mixture model in density estimation.

**GMM:** Gaussian Mixture Model. A model often used to estimate the density from which a set of (possibly multivariate) real-valued observations were sampled independently and identically.

**HDP:** Hierarchical Dirichlet Process. An extension of the Dirichlet process mixture model to the mixed-membership mixture modeling setting.

**HMM:** Hidden Markov Model. A model of time-series data that assumes that each observation depends on a hidden state that in turn depends on the hidden state associated with the previous observation.

**IBP:** Indian Buffet Process. A Bayesian nonparametric prior on binary matrices of infinite dimension. Arises as a special case of the beta process.

**ICMC:** International Computer Music Conference.

**ID3:** A tagging standard for MP3 files.

**IS-NMF:** Nonnegative Matrix Factorization with an Itakura-Saito cost function.

**ISMIR:** The International Conference on Music Information Retrieval (formerly the International Symposium on Music Information Retrieval, currently the conference organized by the International Society for Music Information Retrieval).

$\mathcal{IW}$**:** Used to denote the Inverse-Wishart distribution.

**KHz:** KiloHertz. 1000 cycles per second.

**KL/KLD:** Kullback-Leibler Divergence. A measure of the dissimilarity of two probability mass functions or probability density functions.

**KL-NMF:** NMF with a generalized Kullback-Leibler Divergence cost function.

**LDA:** Latent Dirichlet Allocation. A Bayesian model of text used to find "topics" (distributions over words) that make up text corpora.

**LFCC:** Log-Frequency Cepstral Coefficients. A set of coefficients summarizing the rough spectral shape of a (typically short) audio signal. Uses a log-frequency filter bank.

**MAP:** Maximum A Posteriori. Refers to the practice of estimating the parameters to a probabilistic model from data by choosing the parameter setting that has the highest posterior likelihood.

**MAPS:** Midi-Aligned Piano Sounds database. A database of piano recordings with the midi scores used to generate those recordings. Used to evaluate transcription systems.

**MATLAB:** MATrix LABoratory. A popular software package/programming environment for numerical computation.

**MCMC:** Markov Chain Monte Carlo. A set of techniques for drawing samples from a probability distribution that is known only up to a normalizing constant.

**MFCC:** Mel-Frequency Cepstral Coefficients. A set of coefficients summarizing the rough spectral shape of a (typically short) audio signal. Uses a perceptually motivated Mel-frequency filter bank.

**MIDI:** Musical Instrument Digital Interface. A standard protocol used to allow digital devices to communicate musical commands and information.

**MIR:** Music Information Retrieval. An umbrella term for methods related to extracting and retrieving information from music.

**MIREX:** Music Information Retrieval EXchange. An annual comparative evaluation of the performance of various systems on various music information retrieval tasks.

**ML:** Maximum Likelihood. Refers to the practice of estimating the parameters to a probabilistic model from data by choosing the parameter setting that assigns the highest likelihood to the observed data.

$\mathcal{N}$**:** Used to denote the Normal distribution.

$\mathcal{NIW}$**:** Used to denote the Normal-Inverse-Wishart distribution.

**NMF:** Nonnegative Matrix Factorization. Refers to any of a set of techniques for finding two nonnegative matrices that, when multiplied, approximate a given nonnegative matrix.

**PCA:** Principal Components Analysis. A technique for finding a low-dimensional linear transformation of a set of multidimensional variables that preserves as much of the variance of the original data set as possible.

**PDF:** Probability Density Function. A function whose integral over a set of outcomes gives the likelihood of one of those outcomes occurring.

**PLCA:** Probabilistic Latent Component Analysis. Used to refer to the application of the probabilistic latent semantic indexing model to the problem of latent source discovery.

**RMS:** Root Mean-Squared. A measure of the (square root of the) average power of a signal.

**ROC curve:** Receiver-Operating Characteristic curve. A summary of the true positive rate versus the false positive rate of a classifier. Commonly used as a performance measure in information retrieval.

**RP:** R-Precision. A measure summarizing retrieval quality.

**SI-HDP:** Shift-Invariant Hierarchical Dirichlet Process. An extension of the hierarchical Dirichlet process to the problem of discovering time-varying sources in audio spectrograms.

**SI-PLCA:** Shift-Invariant Probabilistic Latent Component Analysis. An extension to probabilistic latent component analysis to the problem of finding latent components that vary over time, in pitch, or in some other dimension.

**SIMM:** Shift-Invariant Mixture of Multinomials. A probabilistic model used to reconstruct target audio signals from a given set of short audio samples.

**SNR:** Signal-to-Noise Ratio. Among other uses, a measure of the quality of source separation algorithms.

**STFT:** Short-Time Fourier Transform. Transforms a discrete signal of finite duration from the time domain to the frequency domain.

**SVM:** Support Vector Machine. A classification algorithm that predicts binary labels on the basis of corresponding vectors of features.

**SXSW:** South-by-SouthWest festival. In this dissertation, refers to a collection of 121 mp3s distributed to promote bands at the 2007 and 2008 festivals.

**TDP:** Transformed Dirichlet Process. An extension of the hierarchical Dirichlet process that allows for the possibility that groups of observations have undergone transformations before being observed.

**TIMIT:** Refers to the Texas Instruments-Massachusetts Institute of Technology corpus of recorded speech. A well annotated corpus of recordings of American speakers.

**VQ:** Vector Quantization. A technique whereby each of a set of real-valued vectors is represented by one of a finite set of "codewords," typically denoting which of a set of cluster centroids each vector is closest to.

# Bibliography


[1] International Society for Music Information Retrieval website. `http://www.ismir.net`.

[2] MIREX wiki. `http://www.music-ir.org/mirex`.

[3] South by Southwest artist showcase. `http://2008.sxsw.com/music/showcases/alpha/0.html`.

[4] S.A. Abdallah and M.D. Plumbley. Polyphonic music transcription by non-negative sparse coding of power spectra. In *Proc. 5th Int'l Conf. on Music Information Retrieval (ISMIR)*, pages 10–14, 2004.

[5] C. Ames. The Markov process as a compositional model: a survey and tutorial. *Leonardo*, 22(2):175–188, 1989.

[6] C. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974.

[7] A. Asuncion, P. Smyth, and M. Welling. Asynchronous Distributed Learning of Topic Models. In *Advances in Neural Information Processing Systems 20 (NIPS) 20*. MIT Press, 2008.

[8] J.J. Aucouturier and F. Pachet. Improving timbre similarity: How high is the sky. *Journal of Negative Results in Speech and Audio Sciences*, 1(1):1–13, 2004.

[9] D. Bansal, B. Raj, and P. Smaragdis. Bandwidth expansion of narrowband speech using non-negative matrix factorization. In *Proc. 9th European Conf. on Speech Communication and Technology*, 2005.

[10] L. Barrington, D. Turnbull, D. Torres, and G. Lanckriet. Semantic similarity for music retrieval. In *Proceedings of the International Symposium on Music Information Retrieval, Vienna, Austria*. Citeseer, 2007.

[11] M Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

[12] J. Bennett and S. Lanning. The Netflix prize. In *Proceedings of KDD Cup and Workshop*, volume 2007. Citeseer, 2007.

[13] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere. Autotagger: a model for predicting social tags from acoustic features on large music databases. *Journal of New Music Research*, 37(2):115–135, 2008.

[14] C.M. Bishop et al. *Pattern recognition and machine learning*. Springer New York:, 2006.

[15] D. Blei and M. Jordan. Modeling annotated data. In *Proc. 26th annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 127–134. ACM Press, 2003.

[16] D. Blei and M. Jordan. Variational methods for the Dirichlet process. In *Proc. 21st Int'l Conf. on Machine Learning*, 2004.

[17] D. Blei and J. Lafferty. Correlated topic models. In *Advances in Neural Information Processing Systems 18 (NIPS) 18*, pages 147–154. MIT Press, 2006.

[18] D. Blei and J. Lafferty. A correlated topic model of Science. *Annals of Applied Statistics*, 1(1):17–35, 2007.

[19] W.J. Boscardin and X. Zhang. Modeling the covariance and correlation matrix of repeated measures. *Applied Bayesian modeling and causal inference from incomplete-data perspectives*, pages 215–226, 2004.

[20] M. Braun and J. McAuliffe. Variational inference for large-scale models of discrete choice. *arXiv*, (0712.2526), 2008.

[21] M.A. Casey. Acoustic lexemes for organizing internet audio. *Contemporary Music Review*, 24(6):489–508, 2005.

[22] Òscar Celma and Paul Lamere. If you like the Beatles you might like...: a tutorial on music recommendation. In *MM '08: Proceeding of the 16th ACM international conference on Multimedia*, pages 1157–1158, New York, NY, USA, 2008. ACM.

[23] P. Chordia and A. Rae. Using source separation to improve tempo detection. In *Proc. Tenth Int'l Conf. on Music Information Retrieval (ISMIR)*, 2009.

[24] E. Coviello, L. Barrington, A.B. Chan, and G.R.G. Lanckriet. Automatic music tagging with time series models. In *Proceedings of the International Conference on Music Information Retrieval*, 2010.

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

[26] F. Doshi-Velez, K.T. Miller, J. Van Gael, and Y.W. Teh. Variational inference for the indian buffet process. In *Proc. 13th Int'l Conf. on Artificial Intelligence and Statistics*, pages 137–144, 2009.

[27] S. Dubnov, G. Assayag, and A. Cont. Audio oracle: A new algorithm for fast learning of audio structures. In *Proceedings of International Computer Music Conference (ICMC)*, pages 224–228, 2007.

[28] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In *Advances in neural information processing systems*, volume 20, pages 385–392. Citeseer, 2007.

[29] V. Emiya, R. Badeau, and B. David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6), 2007.

[30] M. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–588, 1995.

[31] T. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1:209–230, 1973.

[32] C. Févotte, N. Bertin, and J.L. Durrieu. Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis. *Neural Computation*, 21(3):793–830, 2009.

[33] C. Févotte and A.T. Cemgil. Nonnegative matrix factorizations as probabilistic inference in composite models. In *Proc. 17th European Signal Processing Conf. (EUSIPCO), Glasgow, Scotland*, 2009.

[34] E. Fox, E. Sudderth, M. Jordan, and A. Willsky. Developing a tempered HDP-HMM for systems with state persistence. Technical report, MIT Laboratory for Information and Decision Systems, 2007.

[35] J.L. Gauvain and C.H. Lee. MAP estimation of continuous density HMM: theory and applications. In *Proceedings of the workshop on Speech and Natural Language*, page 190. Association for Computational Linguistics, 1992.

[36] A. Gelman. Exploratory data analysis for complex models. *Journal of Computational and Graphical Statistics*, 13(4):755–779, 2004.

[37] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[38] S.M. Gerrish and D.M. Blei. A language-based approach to measuring scholarly impact. In *ICML*, 2010.

[39] Celemony Software GmbH. Melodyne. `http://www.celemony.com/cms/`.

[40] T. Griffiths and M. Steyvers. Finding scientific topics. *Proc. National Academy of Science*, 2004.

[41] T.L. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process. In *Advances in Neural Information Processing Systems 17 (NIPS)*, pages 475–482. MIT Press, 2005.

[42] G. Grindlay and D.P.W. Ellis. Multi-voice polyphonic music transcription using eigeninstruments. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2009.

[43] W. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[44] M. Hoffman, D. Blei, and P. Cook. Content-based musical similarity computation using the hierarchical Dirichlet process. In *Int'l Conf. on Music Information Retrieval*, 2008.

[45] M. Hoffman, P. Cook, and D. Blei. Data-driven recomposition using the hierarchical Dirichlet process hidden Markov model. In *Int'l Computer Music Conf.*, 2008.

[46] M. Hoffman and P.R. Cook. Feature-based synthesis: mapping acoustic and perceptual features onto synthesis parameters. In *Proceedings of the International Computer Music Conference (ICMC06)*, volume 4, 2006.

[47] M.D. Hoffman, D.M. Blei, and P.R. Cook. Easy as CBA: a simple probabilistic model for tagging music. In *Proceedings of the 10th International Conference on Music Information Retrieval*, 2009.

[48] M.D. Hoffman, D.M. Blei, and P.R. Cook. Finding latent sources in recorded music with a shift-invariant HDP. In *Proc. Digital Audio Effects (DAFx-09)*, 2009.

[49] M.D. Hoffman, D.M. Blei, and P.R. Cook. Bayesian nonparametric matrix factorization for recorded music. In *ICML*, 2010.

[50] M.D. Hoffman, P.R. Cook, and D.M. Blei. Bayesian spectral matching: Turning Young MC into MC Hammer via MCMC sampling. In *Int'l Computer Music Conf.*, 2009.

[51] J.H. Jensen, D.P.W. Ellis, M.G. Christensen, and S.H. Jensen. Evaluation of distance measures between gaussian mixture models of mfccs. In *Proc. ISMIR*, pages 107–108, 2007.

[52] M. Jordan. An introduction to probabilistic graphical models. 2009.

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

[54] Bent Jørgenson. *Statistical properties of the generalized inverse-Gaussian distribution*. Springer-Verlag, New York, 1982.

[55] J.F.C. Kingman. *Poisson processes*. Oxford University Press, USA, 1993.

[56] A. Klapuri. *Signal processing methods for the automatic transcription of music*. PhD thesis, Tampere University of Technology, Finland, 2004.

[57] A. Lazier and P. Cook. MOSIEVIUS: Feature driven interactive audio mosaicing. In *International Conference on Digital Audio Effects (DAFx)*, 2003.

[58] D.D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13 (NIPS)*, pages 556–562. MIT; 1998, 2001.

[59] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob., Vol. 1*, 1966.

[60] M. Mandel and D. Ellis. LabROSA's audio classification submissions, mirex 2008 website. http://www.music-ir.org/mirex/2008/abs/AA_AG_AT_MM_CC_mandel.pdf.

[61] P. Manning, C. Raghavan and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[62] D. McEnnis, C. McKay, I. Fujinaga, and P. Depalle. jAudio: A feature extraction library. In *Proceedings of the International Conference on Music Information Retrieval*, pages 600–603, 2005.

[63] X. Meng and W. Wong. Simulating ratios of normalizing constants via a simple identity: A theoretical exploration. *Statistica Sinica*, 6:831–860, 1996.

[64] N. Metropolis, A. Rosenbluth, M. Rosenbluth, M. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[65] R. Miotto, L. Barrington, and G. Lanckriet. Improving auto-tagging by modeling semantic co-occurrences. In *Proceedings of the International Conference on Music Information Retrieval*, 2010.

[66] T.J. Mitchell and J.J. Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032, 1988.

[67] R. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

[68] R. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[69] Radford M. Neal and Geoffrey E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. MIT Press, 1999.

[70] J. Paisley and L. Carin. Nonparametric factor analysis with beta process priors. In *Proc. 26th Int'l Conf. on Machine Learning*, 2009.

[71] E. Pampalk. *Computational Models of Music Similarity and their Application to Music Information Retrieval*. PhD thesis, Vienna University of Technology, Austria, 2006.

[72] E. Pampalk and M. Gasser. An implementation of a simple playlist generator based on audio similarity measures and user feedback. In *Int. Conf. on Music Information Retrieval*. Citeseer, 2006.

[73] G.E. Poliner and D.P.W. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Applied Signal Processing*, 2007(1):154, 2007.

[74] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77:257–286, 1989.

[75] C. Roads. *The computer music tutorial*. The MIT Press, 1996.

[76] D. Schwarz. Concatenative sound synthesis: The early years. *Journal of New Music Research*, 35(1):3–22, 2006.

[77] J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

[78] K. Seyerlehner, A. Linz, G. Widmer, and P. Knees. Frame level audio similarity—a codebook approach. In *Proc. of the 11th Int. Conference on Digital Audio Effects (DAFx08), Espoo, Finland, September*, 2008.

[79] P. Smaragdis. Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. *Independent Component Analysis and Blind Signal Separation*, pages 494–499, 2004.

[80] P. Smaragdis and J.C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180, 2003.

[81] P. Smaragdis, B. Raj, and M. Shashanka. Sparse and shift-invariant feature extraction from non-negative data. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE Int'l Conf. on*, pages 2069–2072, 2008.

[82] E. Sudderth, A. Torralba, W. Freeman, and A. Willsky. Describing visual scenes using transformed Dirichlet processes. In *Advances in Neural Information Processing Systems 18*, 2005.

[83] V.Y.F. Tan and C. Févotte. Automatic relevance determination in nonnegative matrix factorization. In *Proc. Workshop on Signal Processing with Adaptative Sparse Structured Representations (SPARS09)*, 2009.

[84] Y. Teh, D. Gorur, and Z. Ghahramani. Stick-breaking construction for the Indian buffet process. In *11th Conf. on Artificial Intelligence and Statistics*, 2007.

[85] Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2007.

[86] R. Thibaux and M. Jordan. Hierarchical beta processes and the Indian buffet process. In *11th Conf. on Artificial Intelligence and Statistics*, 2007.

[87] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. Multilabel classification of music into emotions. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, 2008.

[88] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Towards musical query-by-semantic-description using the CAL500 data set. In *Proc. ACM SIGIR*, pages 439–446, 2007.

[89] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio Speech and Language Processing*, 16(2), 2008.

[90] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Towards musical query-by-semantic description using the CAL500 data set. In *ACM Special Interest Group on Information Retrieval Conference (SIGIR '07)*, 2007.

[91] G. Tzanetakis. *Manipulation, Analysis and Retrieval Systems for Audio Signals*. PhD thesis, Princeton University, Princeton, New Jersey, 2002.

[92] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

[93] T. Virtanen, A.T. Cemgil, and S. Godsill. Bayesian extensions to non-negative matrix factorisation for audio signal modelling. In *Proc. of IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP08)*, pages 1825–1828, 2008.

[94] M.J. Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

[95] C. Wang, D. Blei, and L. Fei-Fei. Simultaneous image classification and annotation. In *Proc. IEEE CVPR*, 2009.

[96] Ge Wang and Perry R. Cook. Chuck: A concurrent, on-the-fly, audio programming language. In *2003 Int'l Computer Music Conference*, 2003.

[97] K. Yoshii and M. Goto. Continuous pLSI and smoothing techniques for hybrid music recommendation. In *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR)*, 2009.

[98] K. Yoshii and M. Goto. Infinite latent harmonic allocation: A nonparametric Bayesian approach to multipitch analysis. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, 2010.

[99] A. Zils and F. Pachet. Musical mosaicing. In *International Conference on Digital Audio Effects (DAFx)*, 2001.