INTERACTIVE SOUND SOURCE SEPARATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF MUSIC
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Nicholas J. Bryan
March 2014

This dissertation is online at: http://purl.stanford.edu/vx785cj6510

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Ge Wang, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Jonathan Abel

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Christopher Chafe

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Julius Smith, III

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

In applications such as audio denoising, music transcription, music remixing, and audio-based forensics, it is desirable to decompose a single-channel recording into its respective sources. One of the most promising and effective classes of methods to do so is based on non-negative matrix factorization (NMF) and related probabilistic latent variable models (PLVMs). Such techniques, however, typically perform poorly when no isolated training data is given and offer no mechanism to improve upon unsatisfactory results.

To overcome these issues, we present a new interaction paradigm and separation algorithm for single-channel source separation. The method works by allowing an end-user to roughly paint on time-frequency displays of sound. The rough annotations are then used to constrain, regularize, or otherwise inform an NMF/PLVM algorithm using the framework of posterior regularization and to perform separation. The output estimates are presented back to the user and the entire process is repeated in an interactive manner, until a desired result is achieved.

To test the proposed method, we developed and released an open-source software project embodying our approach, conducted user studies, and submitted separation results to a community-based signal separation evaluation campaign. For a variety of real-world tasks, we found that expert users of our proposed method can achieve state-of-the-art separation quality according to standard evaluation metrics, and inexperienced users can achieve good separation quality with minimal instruction. In addition, we show that our method can perform well with or without isolated training data and is relatively insensitive to model selection, thus improving upon past methods in a variety of ways.

Overall, these results demonstrate that our proposed approach is both a general and powerful separation method and motivates further work on interactive approaches to source separation. To download the application, code, and audio/video demonstrations, please see `http://ccrma.stanford.edu/~njb/thesis`.

This thesis is dedicated to
my parents, Carol and William Bryan,
who have given me everything
and asked for nothing in return.

Acknowledgements

This work would not be possible without the help, guidance, and encouragement of my countless teachers, advisors, colleagues, friends, and family. Thank you to my advisor Prof. Ge Wang for his inspiring teaching and mentorship as well as encouraging me to find my own research path. Thank you to Gautham J. Mysore for becoming an essential secondary advisor over the last two years of my graduate studies. Gautham helped me take my initial thesis proposal and develop it into a true body of work, resulting in several conference and workshop papers together.

Thank you to my dissertation committee including Julius O. Smith III, Jonathan S. Abel, Chris Chafe, and Bernard Widrow, who have all always been exemplary mentors, teachers, and scholars. Thank you to the numerous other CCRMA and music department faculty and staff, including John Chowning, Malcolm Slaney, Dave Berners, Jonathan Berger, Marina Bosi, Jay Kadis, Carr Wilkerson, Fernando Lopez-Lezcano, Sasha Leitman, Debbie Barney, Mario Champagne, Nette Worthey, Bissera Pentcheva, Takako Fujioka, Tom Rossing, Erik Ulman, Thomas Grey, and George Barth. In addition, thank you to David Kerr for years of collaboration, help, and amazing audio/visual work.

Thank you to my student colleagues at CCRMA, including Baek San Chang, Turner Kirk, Jieun Oh, Eunjoon Cho, Dennis Sun, François Germain, Hongchan Choi, Jorge Herrera, Diana Siwiak, Blair Bohannan, Rob Hamilton, Juhan Nam, Jack Perng, Greg Sell, Miriam Kolar, Travis Skare, Esteban Maestre, John Granzow, Spencer Salazar, Luke Dahl, Patty Huang, Keun Sup Lee, SongHui Chon, Michael Berger, Sylvain Le Groux, Mauricio Rodriguez, and Romain Michon.

Thank you to Paris Smaragdis for tremendous inspiration, research, and guidance at Adobe Research and beyond. Thank you to Rebecca Fiebrink for introducing me to the

topic of interactive machine learning and for your inspiring work on the subject. Thank you to my colleagues at Adobe Research over the past two years (in addition to Gautham and Paris), including Minje Kim, Matt Hoffman, David Salesin, Joel Brandt, Tom Lieber, Mira Dontcheva, Wilmot Li, and Andy Moorer. Thank you to my undergraduate professors at the University of Miami, including Ken Pohlmann, Paul Wilson, Joe Abbati, Colby Leider, Rueven Lask, Manohar Murthi, Miroslav Kubat, and Kamal Premaratne.

Thank you to my family. Thank you to my mother and father, Carol and William Bryan, to whom this thesis is dedicated. Thank you to my brother, Andrew Bryan, who has been a tremendous role model and shared many of the same experiences of graduate school, albeit years and distance apart. Thank you to my grandfather and grandmother, Jim and Lili Bryan, for helping support me in my first year at Stanford. Thank you to my many grandparents, cousins, aunts, and uncles. Thank you to my friends in Minnesota, Florida, California, and elsewhere. Lastly, thank you to my girlfriend, Suegol Malek, who has always been loving, supportive, understanding, and caring throughout my pursuit of graduate studies since the first day we met and also spent countless hours editing this thesis.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Over the past several decades, there has been a large research interest in the problem of single-channel sound source separation. Such work focuses on the task of separating a single mixture recording into its respective sources and is motivated by the fact that real-world sounds are inherently constructed by many individual sounds (e.g., human speakers, musical instruments, background noise, etc.). While source separation is a difficult and ill-defined mathematical problem, the topic is highly motivated by many outstanding problems in audio signal processing and machine learning, including the following:

- Speech denoising and enhancement – the task of removing background noise (e.g., wind, babble, etc.) from recorded speech and improving speech intelligibility for human listeners and/or automatic speech recognizers.
- Content-based analysis and processing – the task of extracting and/or processing audio based on semantic properties of the recording such as tempo, rhythm, and/or pitch.
- Music transcription – the task of notating an audio recording into a musical representation such as a musical score, guitar tablature, or other symbolic notation.
- Audio-based forensics – the task of examining, comparing, and evaluating audio

recordings for scientific and/or legal matters.

- Audio restoration – the task of removing imperfections such as noise, hiss, pops, and crackles from (typically old) audio recordings.
- Music remixing and content creation – the task of creating a new musical work by manipulating the content of one or more previously existing recordings.

In this work, we are acutely interested in the application of music remixing and content creation and wish to expand upon a recording engineer's ability to manipulate recorded sound. The modern day practice of music remixing and related compositional techniques emerged from the musique concrète movement of the 1940s and 1950s [1] and, again in the 1960s and 1970s from Jamaican dance hall music [2]. Since then, music remixing has heavily influenced the development of disco, hip-hop, and pop music and has had far-reaching consequences towards the aesthetic and artistic nature of music creation. To create a remix, a composer will typically take a previously existing monophonic or stereophonic mixture recording (e.g., pop song), select a subset of the material (e.g., drum break, bass line, guitar solo, etc.), process the material (e.g., change the pitch, tempo, rhythm, etc.), and mix the content together with new, alternative material.

This process, however, is severely limited by the fact that any selected subset of the existing recordings will be composed of a mixture of sound sources (e.g., drums, bass, guitar, and vocals). That is, unless a sound source is soloed within the original recording or the composer has access to the original source recordings of the given song, the creative process is compromised. As a result, we are motivated to develop a source separation approach for this application, which is general-purpose, of high-quality, can separate single-channel and/or stereo-channel recordings, and is easily useable by recording engineers, musicians, and similar end-users.

1.2 General Approaches to Source Separation

Given the above goal, we can first explore the many existing technical approaches to source separation and analyze which are best suited to our needs. Popular methods include microphone array processing [3], adaptive signal processing [4], independent component analysis

[5], computational auditory scene analysis [6], sinusoidal modeling [7, 8], classical denoising and enhancement [9, 10, 11, 12], and, more recently, non-negative matrix factorization (NMF) [13, 14, 15, 16] and related probabilistic latent variable models (PLVMs) [17, 18].

Out of these, NMF/PLVM-based separation methods are among the most promising, general-purpose, single-channel (and/or stereo-channel) separation techniques and are well suited to our needs. These methods operate by modeling audio spectrogram data or equivalently the magnitude of the short-time Fourier transform (STFT) of an audio recording as a linear combination of prototypical spectral components over time. The prototypical spectral content for each sound source is typically learned from data, used to estimate the contribution of each source within an unknown mixture, and eventually perform separation.

In many cases, these methods can achieve good separation results by leveraging isolated recordings of individual sound sources (training data) to learn individual source models and then separate an unknown mixture of similar sounding sources using the trained models [19]. When no training data is available, however, the methods are not useable without further assumptions. In addition, even with training data, these techniques can be plagued by a lack of separation, sound artifacts, musical noise, and other undesirable effects, and offer little to no mechanism to improve upon unsatisfactory separation, limiting the general usability of the methods.

As a result of these issues, a significant research effort has gone into improving NMF/PLVM-based separation methods by incorporating additional information to inform the separation process. Common methods of informed source separation include incorporating spatial information [20], temporal dynamics [21, 22], musical notation or score information [23, 24, 25, 26, 27], and user-guidance [28, 29, 30, 31, 32, 33]. In this work, we build upon past user-guided separation approaches and propose a closely related approach.

1.3 Proposed Approach

Our proposed method works by allowing end-users to perform single-channel source separation by roughly painting on time-frequency visualizations of sound, as shown in Fig. 1.1 (Upper Middle). We then incorporate the painting annotations into an NMF/PLVM-based

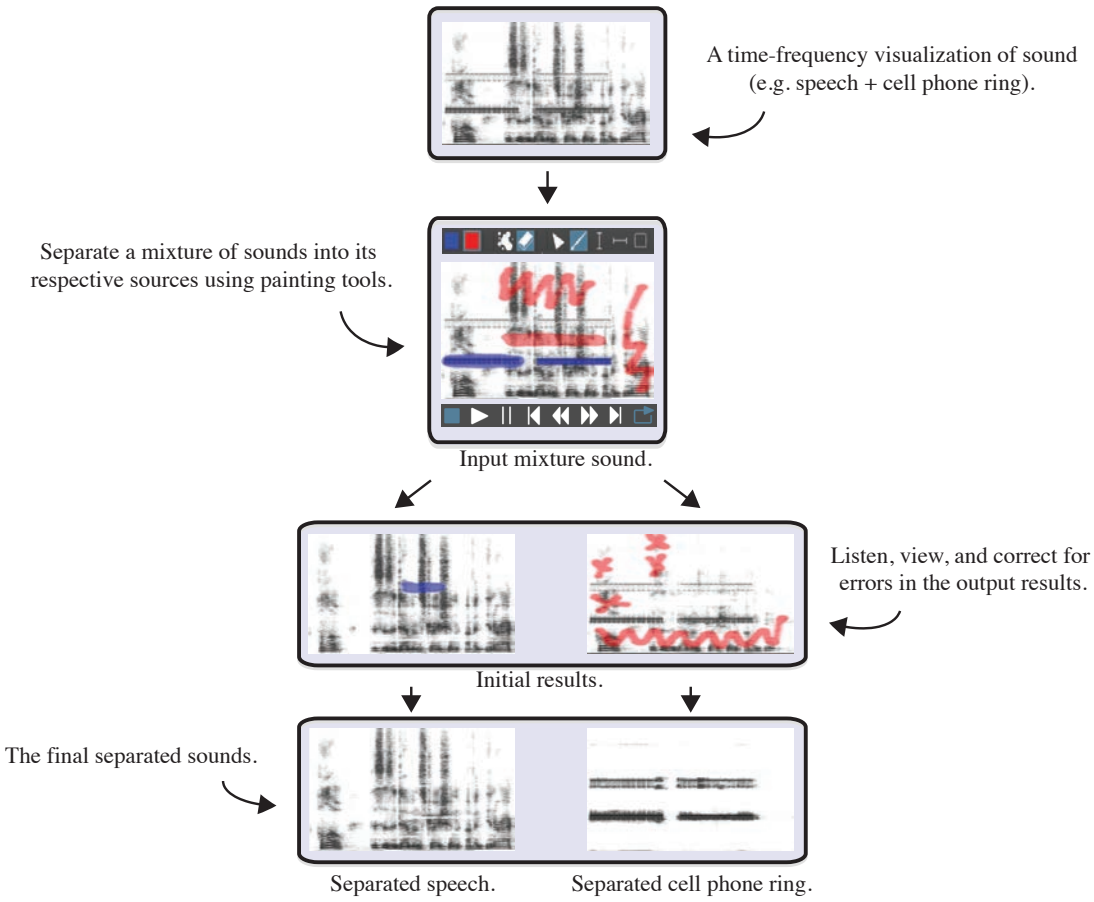


Figure 1.1: The proposed interactive source separation methodology. (Top) A time-frequency display of sound depicting recorded speech mixed with a cell phone (horizontal lines) ringing in the background. (Upper Middle) Using our proposed method, a user can separate distinct sound sources by roughly painting on time-frequency displays. Color is used to denote sound source and opacity is used as a confidence level. (Lower Middle) Once separated, fine-tuning is performed by painting on intermediate separation output estimates. Painting on one output track at a particular point pushes the sound into the other track(s) in an intelligent way. (Bottom) The final separated output recordings.

separation algorithm using the framework of posterior regularization [34, 35], perform an initial separation, and allow a user to listen to the separated outputs, as shown in Fig. 1.1 (Lower Middle). If the results are unsatisfactory, the user can then annotate errors in the output estimates or further annotate the input, and iteratively re-run the process in a quick, interactive manner until a desired result is achieved, as shown in Fig. 1.1 (Bottom). This

overarching approach encompasses both a new interaction paradigm and separation algorithm, which are used in synchrony to iteratively achieve high-quality separation results and together form an *interactive* approach to source separation.

For evaluation, we developed and released an open-source software project embodying our approach, conducted user studies, and compared the separation quality of our work against past methods. We also submitted separation results to a community-based source separation evaluation campaign and achieved state-of-the-art separation quality according to standard evaluation metrics for a variety of real-world tasks. In addition, we found that our method can perform well with or without isolated training data and is relatively insensitive to model selection. To download the developed software, source code, and audio/video demonstrations, please see `http://ccrma.stanford.edu/~njb/thesis`.

In the remainder of this chapter, we discuss related work on user-guided source separation and interactive machine learning in Section 1.4 and Section 1.5. We then discuss the core contributions of this thesis in Section 1.6, followed by an outline of the remaining chapters in Section 1.7, and a brief overview of the mathematical notation in Section 1.8.

1.4 User-Guided Source Separation

Research on user-guided source separation has grown significantly over the past five years and includes multiple efforts to apply prior information to NMF/PLVM-based methods and other source separation methods. Typically, the design process for user-guided separation algorithms consists of two formal elements. First, the particular modality of user-input must be decided. Popular forms of input include singing, humming, or other audio-based data as well as graphical annotation-based input. Second, once the form of user-input is decided, the algorithmic mechanism to incorporate the user-input into a new or existing separation algorithm must be designed. This typically involves mapping the user-input into some form of optimization constraint or regularization parameter, but it is highly dependent on the separation method used.

Below, we discuss user-guidance in the context of NMF/PLVM-based separation methods in Section 1.4.1 and then discuss user-guided approaches that are used with alternative separation methods in Section 1.4.2.

1.4.1 User-Guided NMF/PLVM-Based Separation Methods

When we discuss user-guided NMF/PLVM-based separation methods, we divide the user-input methods into audio-based and graphical-based user-input.

In terms of audio-based input, we see one of the first examples of user-guided NMF/PLVM-based separation in the work of Smaragdis and Mysore [28, 29]. In this case, a user is employed to sing or hum an input query signal. The frequency content and timing information of the query signal is then used to select or separate the query source from a mixture. In closely related work, Fitzgerald [36] takes a similar approach, but extends the user-guidance to handle multichannel signals. While an appealing and natural interaction, such audio-based interaction is difficult to elicit in a precise manner, limiting separation quality and motivating alternative approaches.

In terms of graphical annotation-based input, we note several interesting annotation methods and corresponding separation algorithms. In the work of Wang and Plumbley [37], human guidance and interaction is used to guide NMF without training data via listening to and graphically grouping basis vectors. The grouped basis vectors are then used to better reconstruct each sound source within a mixture.

In the work of Ozerov et al. [30, 38], a user is employed to annotate timing activations of isolated sound sources from an input mixture recording to inform a separation algorithm. The timing information is then applied in the form of hard equality constraints (zero values) into an NMF-based model. Also of interest, Kirchhoff et al. use similar user-guided timing information for the purpose of polyphonic music transcription [39].

The benefit of this type of interaction is that it is simple for a user to perform and typically easy to implement. Unfortunately, if there is no training data available or there are no isolated time regions within a given recording, different methods must be used. Also, in many cases, simple timing information is not enough to improve the separation quality to a suitable level.

Alternatively, Durrieu et al. [31] and Lefèvre et al. [32] discuss two different methods that allow a user to annotate a time-frequency visualization of sound to improve separation (the most similar to our proposed approach). Durrieu's method employs an end-user to

annotate the fundamental frequency of musical sources to improve results, while the latter method of Lefèvre allows a user to annotate binary time-frequency regions of different sources in a spectrogram. Both methods are based on a form of non-negative matrix factorization and can perform well, but also have certain limitations. In particular, for sounds that do not have a fundamental frequency, the method of Durrieu et al. [31] becomes less useful. Also, the method of Lefèvre was only developed for binary annotations and is oriented towards gathering data to train an automatic, user-free system.

1.4.2 Alternative User-Guided Separation Methods

In addition to NMF/PLVM-based user-guided separation techniques, it is useful to discuss other promising alternative approaches to user-guided separation. The most prominent alternative approaches are those that allow user-guidance on time-frequency displays (e.g., paint brush, box selection, etc.) and use some alternative method of separation. This past work includes AudioSculpt [40, 41], Audio Brush [42], SPEAR [43], and a host of many related recent commercial audio editing tools, including Celemony's Melodyne [44], Sony's Spectral Layers [45], Adobe Audition [46], and Izotope's RX [47].

Typically, these approaches operate by having some form of spectral representation of sound (e.g., STFT, phase vocoder, sinusoidal modeling, etc.). Then, a user is employed to manually manipulate (typically) each time-frequency point of the representation, essentially allowing a user to craft a custom time-varying filter to remove any unwanted sound sources from a mixture. In many cases, however, this type of selection is tedious, needs to be quite precise, and requires significant manual work, limiting the separation quality.

To combat this problem, some editors, such as SPEAR or Spectral Layers, provide "smart" selection tools, which blend user-guidance with machine intelligence such as automatically selecting sinusoidal peak tracks, harmonics, noise regions, etc. Others, including Melodyne, Audition, and RX, seem to incorporate learning-like abilities to perform separation that adapts to user-input, albeit typically for specific problem cases such as polyphonic pitch-based separation, denoising, or dereverberation. Unfortunately, many of the latter approaches are proprietary and do not provide open access to their method of operation, making comparison difficult.

Even so, all such past work prompts important questions on how user-guidance can be used to inform and improve separation algorithms. Questions include: What is the best and most informative method of user-guidance for source separation? What is the best way to algorithmically incorporate this guidance? Can simple user-guided approaches outperform more complicated, fully automatic approaches? What is the limit on separation quality if a user is willing to spend infinite time aiding a separation algorithm? Overall, this past work sets the foundation of our proposed approach discussed in Chapter 3, in addition to our discussion below on interactive machine learning.

1.5 Interactive Machine Learning

Interactive machine learning (IML) is an emerging research area that focuses on the intersection between human-computer interaction (HCI) and machine learning. IML has found recent success across several domains and generally consists of researching and developing machine learning techniques from an end-users' perspective [48]. In many cases, this reduces to thinking about how end-users can 1) both create and deploy machine learning algorithms as opposed to scientists and engineers, and 2) inform a learning algorithm in some way to learn faster, learn more accurately, and/or learn tasks not previously possible without user-interaction.

While challenging, an HCI approach to machine learning carries significant potential and motivation. Firstly, by simply allowing end-users to both train and deploy algorithms that learn, we give machine learning algorithms an orders-of-magnitude larger audience. We also make them more useful in that end-users can arbitrarily define what concept or task is to be learned for a given algorithm. Thirdly, in a user-interactive scenario, a learning algorithm can elicit more information from a user (e.g., feature-labeled training data) compared to what is ordinarily provided in the standard machine learning scenario (i.e., class-labeled training data). With more information, it is then possible to train machine learning algorithms to learn faster, learn more accurately, and/or learn tasks not previously possible—all with the simple guidance of a human user.

Early works citing success of IML include Fails and Olsen [48], who leverage user-feedback for training image classifiers; and Cohn et al. [49], who leverage IML for document clustering and scientific discovery. More recent works, which are also encouraging, include Stumpf et al. [50], who discuss the idea of IML in a broad machine learning context; Talbot et al. [51], who use an interactive visualization tool for training multiple classifiers; Fogarty et al. [52], who leverage IML for image search; Fiebrink [53], who uses IML for musical instrument design; Settles [54], who leverages IML for natural language processing tasks in conjunction with active learning; and Amershi et al. [55], who leverage IML for network alarm triage.

With this context in mind and a basic understanding of our proposed approach, it is useful to consider our proposed interactive separation approach as a specific instance of an IML algorithm. As such, note that our proposed approach to source separation benefits from user-interaction in the same way general IML algorithms benefit. That is, we allow end-users to both train and deploy a learning algorithm, making our method significantly more useful and general compared to approaches without user-interaction. We also leverage far more information than simple class-labeled training data in our separation algorithm (i.e., user-feedback encoded as optimization constraints), allowing us to outperform past approaches without user-interaction.

Given this background, we now outline the core contributions of this work, outline background information in Chapter 2, and then continue this discussion in Chapter 3.

1.6 Contributions

The overall contribution of this thesis is a general-purpose, high-quality source separation approach and algorithm that leverages interactive user-feedback to perform separation. The specific contributions are:

- A new interaction paradigm for incorporating user-feedback into NMF/PLVM-based separation algorithms by painting on time-frequency displays of sound,
- A new NMF/PLVM-based separation algorithm that incorporates painting-based user-annotations, which allows separation with or without the use of isolated training data

and is insensitive to model selection,

- Evaluation showing that expert users of our approach can achieve state-of-the-art separation quality on a variety of real-world tasks,
- Evaluation showing that inexperienced users of our approach can achieve good separation quality with minimal instruction,
- An open-source, freely available, cross-platform software tool that embodies the interaction paradigm and separation algorithm.

In addition to these outlined contributions, publications leading up to this thesis include:

- An extended abstract (workshop) publication of our initial approach [56],
- A conference publication outlining the core algorithmic and mathematical contribution [57],
- A conference publication outlining algorithmic extensions and additional geometric interpretation [58],
- A conference publication outlining a modified algorithm applied to separating polyphonic instrumental music [59],
- Published separation results in the fourth signal separation evaluation campaign (SiSEC) [60],
- An extended abstract (workshop and demonstration) publication on our open-source software project [61],
- A conference paper on the developed software system and core human-computer interaction contributions [62].

1.7 Outline

In Chapter 2, we begin our discussion by providing background and foundational information related to non-negative matrix factorization and related probabilistic models. We

outline similarities and (slight) differences between the two approaches and discuss how they are used for the purpose of single-channel source separation.

In Chapter 3, we discuss our interactive approach to source separation, contributions regarding interaction design, and the application of user-feedback for source separation algorithms. We also describe goals of the work, analogies from contrasting domains, and principles of design, including differences between user-guided and interactive approaches to source separation.

In Chapter 4, we discuss how we leverage the proposed interaction design for the purpose of informing an NMF/PLVM-based separation method. This chapter yields the majority of the mathematical contribution of this work and is the most technical in nature.

In Chapter 5, we discuss our open-source software project that embodies our interactive approach to source separation.

In Chapter 6, we discuss evaluation of the proposed method, including results from initial experiments, user-testing, and the fourth signal separation evaluation campaign (SiSEC), which show that the proposed method can achieve state-of-the-art separation quality.

Finally, in Chapter 7, we discuss conclusions and a variety of future work.

1.8 Notation

For a reference on all notation used throughout this thesis, please see Appendix 1.8. For a limited overview, please see below.

1.8.1 Vector and Matrix Notation

In terms of linear algebra notation, we use lowercase symbols for scalars (e.g., a, b, c), lowercase bold symbols for vectors (e.g., $\mathbf{a}, \mathbf{b}, \mathbf{c}$), and uppercase bold symbols for matrices and tensors (e.g., $\mathbf{A}, \mathbf{B}, \mathbf{C}$). By default all vectors are column vectors. To denote a row vector, we write $\mathbf{x}^{\mathrm{T}}$, where $^{\mathrm{T}}$ is a vector or matrix transpose.

To index vectors, we use x_i to index the ith element of a vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \tag{1.1}$$

$\mathbf{x}_i$ to index the ith column vector of a matrix

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & ... & \mathbf{x}_N \\ | & | & & | \end{bmatrix}, \tag{1.2}$$

$\mathbf{x}_j^{\mathrm{T}}$ to index the jth row vector of a matrix

$$\mathbf{X} = \begin{bmatrix} — & \mathbf{x}_1^{\mathrm{T}} & — \\ — & \mathbf{x}_2^{\mathrm{T}} & — \\ & \vdots & \\ — & \mathbf{x}_M^{\mathrm{T}} & — \end{bmatrix}, \tag{1.3}$$

X_{ij} to index the element at the ith row and jth column of a matrix

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1N} \\ X_{21} & X_{22} & \dots & X_{2N} \\ \vdots & \vdots & \cdots & \vdots \\ X_{M1} & X_{M2} & \dots & X_{MN} \end{bmatrix}, \tag{1.4}$$

and X_{ijk} to index a particular element of a three-dimensional tensor $\mathbf{X}$.

We also denote $\mathbf{I}$ to be a square identity matrix, $\mathbf{1}$ to be a column vector or matrix of ones, $\mathbf{0}$ to be a column vector or matrix of zeros, R as the set of all real-valued numbers, R_+ as the set of all non-negative valued numbers, C as the set of all complex-valued numbers, Δ^n as an n-simplex or the set $\{(t_0, \dots, t_n) \in \mathrm{R}^{n+1} | \sum_{i=0}^{n} t_i = 1, t_i \geq 0 \ \forall i\}$, $\odot$ to

be element-wise matrix or vector multiplication, $/$ to be element-wise vector or matrix division, and $\geq, \leq, >, <$ to all be element-wise comparisons. Finally, if any vector or matrix index should be outside the bounds of the corresponding vector or matrix dimensions, a value of zero is returned.

1.8.2 Random Variables and Probability Notation

In terms of random variable notation, we denote $X(w)$ or X as a random variable, x as an outcome value of the random variable X, $\mathbf{X}$ as a set of random variables, $\mathbf{x}$ as an outcome value for a set of random variables $\mathbf{X}$, $\mathbf{x}_i$ as the ith random variable in the random variable set $\mathbf{X}$, and use the notation $X \sim distribution(\Theta)$ to denote that a random variable is distributed according to a particular distribution with parameters Θ. We also denote $Val(X)$ as the set of values that the random variable X can take, $|Val(X)|$ as the number of elements in $Val(X)$, x^i as the ith outcome value of the random variable X, $\sum_x$ as the sum over all possible values that the random variable X can take and $\sum_{x \in X_s}$ as the sum over a subset s of all possible values that the random variable X can take.

In terms of probability notation, we define $p(X = x) := p(\{w : X(w) = x\})$ to be a probability evaluated at the set of outcomes for which the random variable X takes on the value x or more simply, the probability that the random variable X takes on the value x. We also define $p(x)$ to be shorthand for $p(X = x)$, which is sometimes called the probability mass function. We then define $p(X)$ to denote a distribution over the random variable X.

Then, we define $p(X, Y)$ to be joint probability distribution between the random variables X and Y, $p(X|Y)$ to be the conditional distribution as a function of the random variables X given Y, and $p(X; \theta)$ to be a distribution over the random variable X parameterized by θ. Also note that throughout this work, we will 1) occasionally violate our own notation for clarity and 2) we will only consider discrete random variables and probability distributions. For further notation information, please see Appendix 1.8.

Chapter 2

Background and Foundational Information

2.1 Problem Formulation

The problem of single-channel source separation involves taking a single mixture signal and separating it into its respective sources. For our case, we assume the mixture signal $\mathbf{x} \in \mathbf{R}^{N_\tau}$ is a finite, discrete-time signal that is created via,

$$\mathbf{x} = \sum_{i=1}^{N_s} \mathbf{x}_i, \tag{2.1}$$

where we further define $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_s} \in \mathbf{R}^{N_\tau}$ to be pre-mixed individual sound sources, N_τ to be the total number of samples of each signal, and τ to be the sampling period. Note, in contrast to general multi-channel separation mixing models, we absorb all mixing scale factors into the pre-mixed source signals.

Given only the observed mixture signal $\mathbf{x}$, the separation algorithm must then accurately estimate the individual sources $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N_s}$ as shown in Fig. 2.1 for two sources. This problem is typically formulated as a large, underdetermined inverse estimation problem and is inherently ill-posed (i.e., far more unknown variables than equations, no unique solution), making it technically challenging. Many different approaches, however, have been

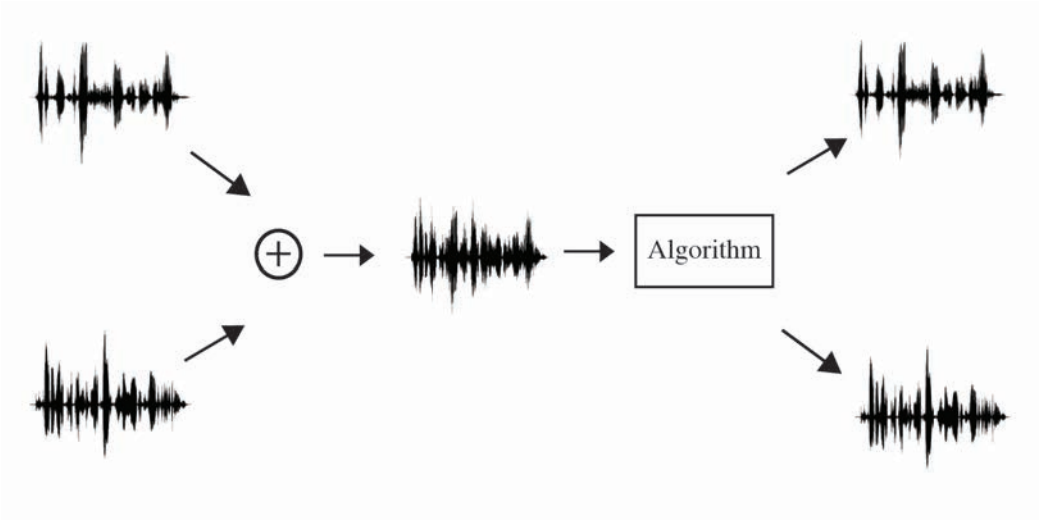


Figure 2.1: Single-channel source separation diagram for two source signals.

proposed to overcome this issue, as we discussed in Chapter 1, including NMF/PLVM-based separation techniques.

2.2 Non-Negative Matrix Factorization and Related Probabilistic Latent Variable Separation Methods

Non-negative matrix factorization (NMF) and related probabilistic latent variable models (PLVMs) are data-driven machine learning techniques that we use for the purpose of source separation. At a high level, when we use NMF/PLVMs for source separation, we decompose audio spectrogram data, or equivalently the magnitude of the short-time Fourier transform (STFT) of an audio recording, as a linear combination of the outer product of prototypical spectral components times vectors of amplitude over time. The spectral components for each sound source and their gains are learned from data and the result is used to estimate the contribution of each source within an unknown mixture over time, and eventually perform separation.

NMF/PLVM methods can also be thought of as basis decomposition or dictionary-based methods and are closely related to sparse coding [63, 64], principal component analysis [65], singular value decomposition [66], independent subspace analysis methods [5, 67,

68], and related matrix factorization methods. In addition to their audio applications, both NMF and PLVMs are also commonly used for processing images, text, and other data types and collectively have gained a significant research interest over the past decade.

2.2.1 Block Diagram

To separate a single-channel recording into its respective sources using NMF/PLVMs, we follow the general separation block diagram illustrated in Fig. 2.2. We begin with a time-domain audio signal $\mathbf{x}$ and transform the signal into a time-frequency domain complex-valued matrix $\mathbf{X}$ using the STFT. We then perform NMF/PLVM estimation on the magnitude of this matrix, use the results from the NMF/PLVM to filter and separate the input signal in the frequency domain, and finally transform the filtered time-frequency domain signal back to the time domain using the inverse STFT and the original signal phase $\angle \mathbf{X}$.

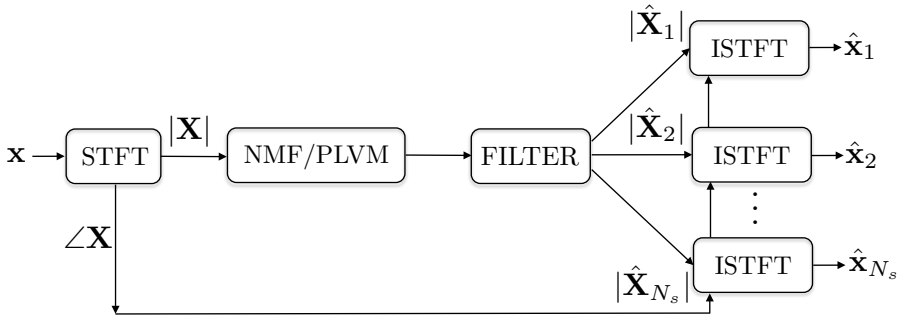


Figure 2.2: Block diagram of NMF/PLVM-based separation methods.

Given this block diagram, we now individually discuss each of the larger computational blocks, including the STFT in Section 2.3.2, NMF in Section 2.4, and PLVMs in Section 2.5. We then discuss how we model mixture sounds using NMF/PLVMs in Section 2.6 and how we separate mixture sounds in Section 2.7.

2.2.2 General References

For readers familiar with NMF/PLVM-based separation methods, this background chapter may be unnecessary for understanding further chapters. For readers interested in additional

supplementary material, there are several books on the subject of source separation that may be of interest, including the text of Common and Jutten [69], the text edited by Makino et al. [70], and the text edited by Wang [71].

In addition, please refer to the works of Julius O. Smith III [72, 73, 8] for signal processing background material; Boyd and Vandenberghe for optimization background material [74]; and Hastie et al. [75], Koller and Friedman [76] and/or Bishop [77] for general probability and machine learning background material.

2.3 Short-Time Fourier Transform Processing

The first stage of our separation process is the short-time Fourier transform (STFT). The STFT is a powerful signal processing tool that is used to transform a time domain signal into complex amplitude values as a function of time and frequency. When applied to finite discrete signals, the forward STFT can be thought of as a process that transforms a time-domain vector $\mathbf{x}$ into a complex time-frequency domain matrix $\mathbf{X}$. Once transformed, the time-frequency signal can be analyzed, visualized, processed, and/or inverted back to the time-domain using the inverse STFT.

For the case of NMF/PLVM-based separation methods, we use the STFT for a variety of purposes. Firstly, we use the STFT time-frequency data $\mathbf{X}$ as the fundamental representation of sound that we model using an NMF/PLVM. Secondly, we use the STFT to implement time-varying linear filtering via overlap-add processing. Thirdly, we use the magnitude of the STFT for visualization purposes. We discuss the discrete Fourier transform, forward STFT, the inverse STFT, overlap-add processing, and STFT visualization in Sections 2.3.1, 2.3.2, 2.3.3, 2.3.4, and 2.3.5, respectively.

2.3.1 The Discrete Fourier Transform

To compute the STFT, we use the discrete Fourier transform (DFT) to convert short, sequential time-windowed segments of the input signal $\mathbf{x}$ into the frequency domain. We

define the discrete Fourier transform (DFT) via

$$y_k = \sum_{n=0}^{N-1} x_n e^{-j\omega_k n}, \quad k = 0, 1, \ldots, N-1, \tag{2.2}$$

where x_n is the n^{th} element of the input signal $\mathbf{x}$, y_k is the k^{th} element of the complex output signal $\mathbf{y}$ in the frequency domain, $\omega_k = 2\pi k/N, \ k = 0, 1, 2, \ldots, N-1$, and N is the DFT size in samples. We also define the inverse DFT (IDFT) via

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} y_k e^{j\omega_k n}, \quad n = 0, 1, \ldots, N-1, \tag{2.3}$$

where y_k is the k^{th} element of the complex input signal $\mathbf{y}$ in the frequency domain and x_n is the n^{th} element of the output signal $\mathbf{x}$ in the time domain.

In addition to Eq. (2.2), we can also formulate the DFT as a single complex-valued matrix multiplication operator. In this case, each column is a sampled sinusoid at a different frequency and is orthogonal to one another [72]. This provides us with the intuition that the DFT is a projection operator that projects the input signal onto a sampled set of complex sinusoids. Similar intuition is true for the IDFT and Eq. (2.3).

In practice, we implement the DFT using the fast Fourier transform (FFT) [78, 79, 8] and restrict the domain of the input to be real-valued. This is done for efficiency purposes and because we are only concerned with real-valued audio input signals. We define $\mathbf{y} \leftarrow \text{FFT}(\mathbf{x})$ as a forward FFT function and $\mathbf{x} \leftarrow \text{IFFT}(\mathbf{y})$ as an IFFT function. These algorithm definitions will be required when we outline the entire STFT procedure.

2.3.2 The Forward Short-Time Fourier Transform

Given the definition of the DFT, we define the discrete STFT via [8, 80]

$$X_{km} = \sum_{n=-\infty}^{\infty} [x_n w_{n-mR}] e^{-j\omega_k n}, \tag{2.4}$$

where x_n is the n^{th} element of an input signal vector $\mathbf{x}$, X_{km} is the $(k,m)^{\text{th}}$ element of the complex amplitude output time-frequency matrix $\mathbf{X}$, w_{n-mR} is the $n-mR^{\text{th}}$ element of the window vector $\mathbf{w}$, M is the window length in samples, N is the DFT size in samples, R is the hop size in samples, and $\omega_k = 2\pi k/N,\ k = 0, 1, 2, \ldots, N-1$. The forward STFT process is illustrated in Fig. 2.3.

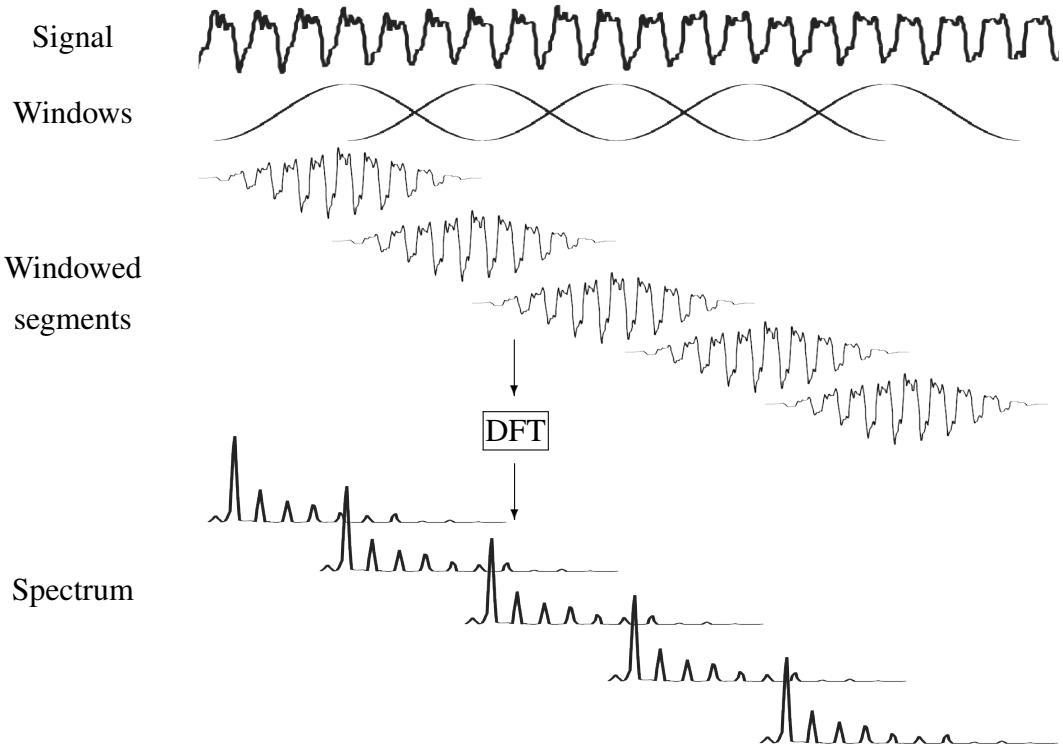


Figure 2.3: The short-time Fourier transform.

In terms of practical implementation, we outline the STFT in Algorithm 1. Alternatively, we can define a modified interface to the STFT function via

$$(|\,\mathbf{X}\,|, \angle\,\mathbf{X}) \leftarrow \text{STFT}(\mathbf{x}, P).$$

In this case, the input arguments consist of the time domain signal and a variable P, which represents all necessary STFT parameters. The output arguments are then two separate real-valued matrices that represent the magnitude $|\,\mathbf{X}\,|$ and phase $\angle\,\mathbf{X}$ of the output matrix. Note, because there is conjugate symmetry in the DFT for real-valued input signals, we

only store the first $N_f = N/2 + 1$ rows of the final STFT output, resulting in an output matrix $\mathbf{X} \in \mathrm{C}^{N_f \times N_t}$.

Algorithm 1 The Short-Time Fourier Transform (STFT)

Procedure STFT (
$\mathbf{x} \in \mathrm{R}^{N_\tau}$, // input signal vector
$\mathbf{w} \in \mathrm{R}^{M}$, // window function of size M
R, // hop size
N, // DFT size
)
initialize: $i \leftarrow 1$, $j \leftarrow 1$
repeat
// select segment of input
$\mathbf{s} \leftarrow [x_i \; x_{i+1} \; \ldots \; x_{i+M-1}]^{\mathrm{T}}$
// apply the window
$\mathbf{s}_w \leftarrow \mathbf{s} \odot \mathbf{w}$
// zero-phase zero-pad to length N
$\mathbf{s}_z \leftarrow [0 \ldots 0 \; \mathbf{s}_w^{\mathrm{T}} \; 0 \ldots 0]^{\mathrm{T}}$
// compute a size N FFT
$\mathbf{z} \leftarrow \mathrm{FFT}(\mathbf{s}_z)$
// only store the first half of the FFT output into the jth column of $\mathbf{Y}$
$\mathbf{y}_j \leftarrow [z_1 \; \ldots \; z_{N/2+1}]^{\mathrm{T}}$
// update indices
$i \leftarrow i + R$
$j \leftarrow j + 1$
until $i > N_\tau$
return: $\mathbf{Y} \leftarrow [\mathbf{y}_1 \ldots \mathbf{y}_{j-1}] \in \mathrm{C}^{N_f \times N_t}$

2.3.3 The Inverse Short-Time Fourier Transform

Once in the time-frequency domain, a signal can be analyzed, visualized, processed, and/or transformed back to the time-domain. To transform back into the time-domain, the inverse short-time Fourier transform (ISTFT) is used. We define the ISTFT via

$$x_n = \frac{1}{N} \sum_{m=-\infty}^{\infty} \sum_{k=0}^{N-1} X_{km} e^{j\omega_k n},$$

where the input to the STFT is a complex-valued matrix $\mathbf{X}$ and the output is a real-valued vector $\mathbf{x}$.

The inverse STFT process is illustrated in Fig. 2.4 and defined in implementation form in Algorithm 2. Following our forward STFT algorithm, we must reflect the complex-valued time-frequency domain input matrix to maintain conjugate symmetry before the inverse DFTs. To do so, we define $\mathbf{X} \leftarrow \text{REFLECT}(\mathbf{X})$ to perform the appropriate conjugate symmetric reflection, which expands the number of rows of the input from $N_f = N/2 + 1$ to N rows (the number of columns stays the same). We also define a modified interface to the ISTFT function via

$$\mathbf{x} \leftarrow \text{ISTFT}(|\,\mathbf{X}\,|, \angle\,\mathbf{X}, P),$$

where the output argument is the same, but the input arguments are modified to accept the time-frequency domain magnitude $|\,\mathbf{X}\,|$, the phase $\angle\,\mathbf{X}$, and all STFT parameters P.

Constant Overlap-Add

To perfectly reconstruct an input time domain signal $\mathbf{x}$ from the STFT signal $\mathbf{X}$ using the ISTFT (assuming no modifications), we must obey the constant overlap add (COLA) property. The COLA property is defined by

$$\sum_{m=-\infty}^{\infty} w_{n-mR} = 1, \forall n$$

and is dependent on the window shape, the window size, and the hop size. We can also write this in algorithmic form via $\mathbf{x} \equiv \text{ISTFT}(\text{STFT}(\mathbf{x}, P), P)$.

Common windows include the rectangular, Hann, Hamming, Blackman, Bartlett, and Kaiser windows. For more information on COLA windows and the short-time Fourier transform in general, please see the work of Smith [8].

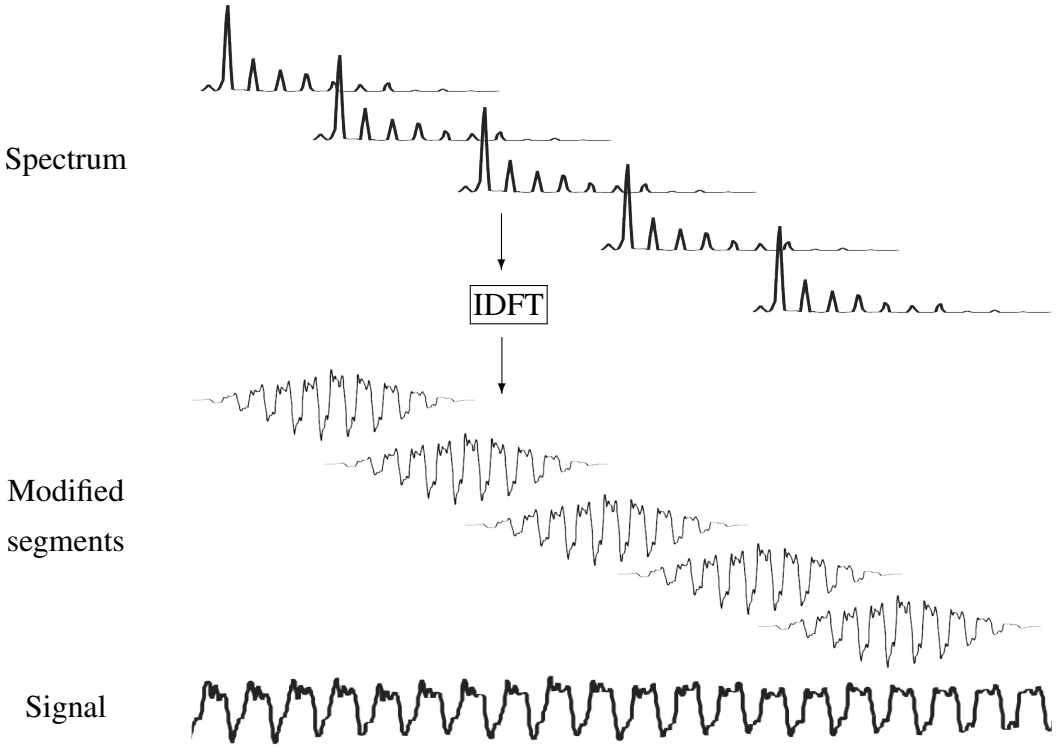


Figure 2.4: The inverse short-time Fourier transform.

Algorithm 2 The Inverse Short-Time Fourier Transform (ISTFT)

Procedure ISTFT (
$\mathbf{X} \in \mathrm{C}^{N_f \times N_t}$, // input signal vector
R // hop size
)
initialize: $\mathbf{y} \leftarrow \mathbf{0} \in \mathrm{R}^{N_\tau}$, $i \leftarrow 1$, $j \leftarrow 1$
// reflect to enforce conjugate symmetry
$\mathbf{X} \leftarrow \text{REFLECT}(\mathbf{X})$
repeat
// inverse FFT of the j^{th} column of $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_{N_t}]$
$\mathbf{s} \leftarrow \text{IFFT}(\mathbf{x}_j)$
// overlap-add to the output vector
$[y_i \; y_{i+1} \; \dots \; y_{i+M-1}]^{\mathrm{T}} \leftarrow [y_i \; y_{i+1} \; \dots \; y_{i+M-1}]^{\mathrm{T}} + \mathbf{s}$
// update indices
$i \leftarrow i + R$
$j \leftarrow j + 1$
until $i > N_t$
return: $\mathbf{y}$

2.3.4 Time-Varying Linear Filtering via Overlap-Add

When we employ the STFT to process and modify a given sound, we perform what is called overlap-add (OLA) processing. OLA processing is an efficient way of implementing time-varying linear filtering (acyclic convolution) in the frequency domain. It is justified via the convolution theorem [8] or the fact that convolution in the time domain corresponds to multiplication in the frequency domain.

To perform time-varying linear filtering using OLA, we first compute the STFT of an input signal, resulting in a time-frequency domain matrix $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_{N_t}] \in \mathrm{C}^{N_f \times N_t}$. We then multiply each column $\mathbf{x}_m$ of $\mathbf{X}$ with a different filter frequency response $\mathbf{f}_m$. Finally, we perform the ISTFT on the filtered signal. In matrix notation, the entire frequency

Algorithm 3 Time-Varying Overlap-Add Processing

Procedure OLA (
$\mathbf{x} \in \mathrm{R}^{N_\tau \times 1}$, // input signal vector
$\mathbf{F} \in \mathrm{C}^{N_f \times N_t}$, // time-varying filter matrix
P // STFT parameters
)
// compute the forward STFT of the input signal
$\mathbf{X} \leftarrow \text{STFT}(\mathbf{x}, P)$
// filter the input signal in the time-frequency domain
$\mathbf{Y} = \mathbf{X} \odot \mathbf{F}$
// compute the inverse STFT of the filtered signal
$\mathbf{y} \leftarrow \text{ISTFT}(\mathbf{Y}, P)$
return: $\mathbf{y}$

domain filtering operation for all time can be written as

$$\mathbf{Y} = \mathbf{X} \odot \mathbf{F}, \tag{2.5}$$

where $\mathbf{F} = [\mathbf{f}_1 \dots \mathbf{f}_{N_t}] \in \mathrm{C}^{N_f \times N_t}$, $\odot$ is element-wise matrix multiplication, and $\mathbf{Y}$ denotes the filtered time-frequency domain output matrix. When done correctly, this allows us to filter the input signal with a filter that changes coefficients every R samples. For our purposes, we only use real-valued filter frequency responses.

Assuming we know the matrix of all filter frequency responses $\mathbf{F}$ and the STFT parameters P, we can write the OLA processing procedure in the form of Algorithm 3. As we will see, this process will be expanded to incorporate an NMF/PLVM parameter estimation algorithm to automatically estimate $\mathbf{F}$.

When performing OLA processing, we must carefully consider several details. Details include 1) the choice of the DFT size N, the window size M, the hop size R, and the shape of the window $\mathbf{w}$, which must be chosen to work together and 2) the shape of each applied filter frequency response (i.e., the columns of $\mathbf{F}$), which must be time-limited. If any of these details are implemented incorrectly, a variety of unwanted audible effects can easily occur.

Weighted Overlap-Add

Complementary to OLA processing, it is also possible to use the weighted overlap-add (WOLA) processing to perform time-varying "instantaneous nonlinear spectral processing" [8]. WOLA processing is near identical to standard OLA processing, but with a small change. The change is that a second "synthesis" window function is applied after each inverse DFT within the STFT. This is used to minimize problems caused by nonlinear processing such as discontinuities, time-aliasing, and other audible artifacts.

When using both an analysis and synthesis window, the two windows must have a cumulative constant overlap-add effect. As a result, we must design both the analysis and synthesis WOLA windows together so that the modified COLA constraint

$$\sum_{m} w_{n-mR}^{2} = const.\ \forall n \tag{2.6}$$

is satisfied. This is in contrast to Eq. (2.5), which is required for OLA processing. To design a window for WOLA, we can simply take standard windows (e.g., Hann, Hamming, etc.) used for OLA processing and square root each element of the window signal.

While WOLA processing is typically not used to implement time-varying linear filtering, as we require, it is sometimes useful in the context of NMF/PLVM separation methods. This is because the second synthesis window helps mitigate problems caused by estimating the filter frequency response matrix $\mathbf{F}$ via NMF/PLVMs. This can be seen as a quick, efficient, and approximate solution to applying invalid filter frequency responses (columns of $\mathbf{F}$), which works reasonably well in practice, but is not optimal.

2.3.5 Visualization

In addition to using the STFT to process sound, we also use it for visualization purposes. To do so, we compute the magnitude of the time-frequency domain matrix $\mathbf{V} = |\,\mathbf{X}\,|$ and then rescale the result to an appropriate range for display. For rescaling, we use a logarithmic amplitude rescaling to convert the STFT magnitude to decibels (dB)

$$\mathbf{V}_{dB} = 20 \log_{10} \mathbf{V}, \tag{2.7}$$

where the logarithm is applied element-wise.

After this, the maximum is found and subtracted off $\mathbf{V}_{dB} = \mathbf{V}_{dB} - \max \mathbf{V}_{dB}$ to normalize the data. From there, a minimum amplitude threshold value is used to clip all values within a finite range (e.g., -60dB - 0dB). This result is linearly scaled within the range of zero to one and mapped to color values according to a standard color map (e.g., jet, black and white, hot, cool, etc.) and format (e.g., RGB values).

2.4 Non-Negative Matrix Factorization

In this section, we define non-negative matrix factorization, formulate it as an optimization problem, derive an algorithm to solve the optimization problem, and discuss a probabilistic interpretation in Sections 2.4.1, 2.4.2, 2.4.3, and 2.4.4, respectively.

2.4.1 Model

Non-negative matrix factorization is a process that approximates a single non-negative matrix as the product of two non-negative matrices. It is defined by

$$\mathbf{V} \approx \mathbf{W}\,\mathbf{H}, \tag{2.8}$$

where $\mathbf{V} \in \mathrm{R}_{+}^{N_f \times N_t}$ is a non-negative input matrix; $\mathbf{W} \in \mathrm{R}_{+}^{N_f \times N_z}$ is a matrix of basis vectors, basis functions, or dictionary elements; $\mathbf{H} \in \mathrm{R}_{+}^{N_z \times N_t}$ is a matrix of corresponding activations, weights, or gains; N_f is the number of rows of the input matrix; N_t is the number of columns of the input matrix; and N_z is the number of basis vectors [81]. Typically $N_z < N_f < N_t$, resulting in a compressed, low-rank approximation of the data $\mathbf{V}$.

Furthermore, we can say the following:

- $\mathbf{V} \in \mathrm{R}_{+}^{N_f \times N_t}$ – original non-negative input data matrix
  - Each column is an N_f-dimensional data sample
  - Each row represents a data feature
- $\mathbf{W} \in \mathrm{R}_{+}^{N_f \times N_z}$ – matrix of basis vectors, basis functions, or dictionary elements
  - A column represents a basis vector, basis function, or dictionary element
  - Each column is not orthonormal, but commonly normalized to one
- $\mathbf{H} \in \mathrm{R}_{+}^{N_z \times N_t}$ – matrix of activations, weights, encodings, or gains
  - A row represents the gain of a corresponding basis vector
  - Each row is not orthonormal, but sometimes normalized to one

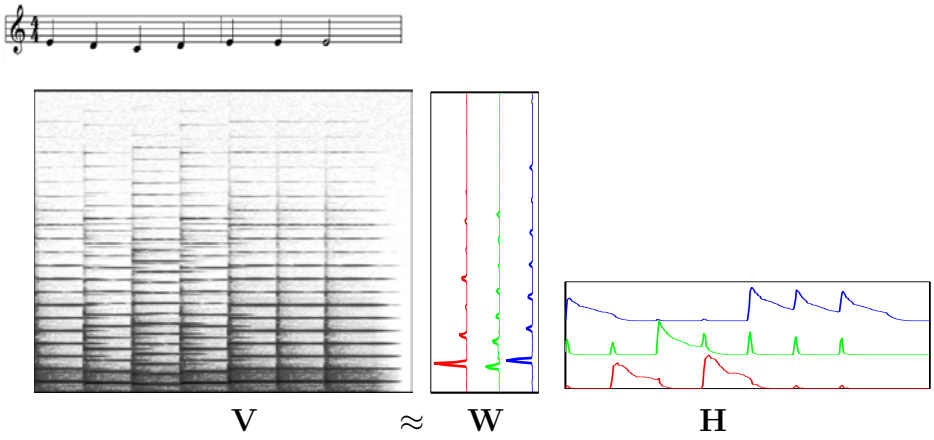


Figure 2.5: NMF of a piano performing "Mary Had a Little Lamb" for two measures with $N_z = 3$. Notice how the matrix $\mathbf{W}$ captures the harmonic content of the three pitches of the passage and the matrix $\mathbf{H}$ captures the time onsets and gains of the individual notes.

When used for audio applications, NMF is typically used to model spectrogram data or the magnitude of STFT data [14]. That is, we take a single-channel recording, transform it into the time-frequency domain using the STFT, take the magnitude or power $\mathbf{V}$, and then approximate the result as $\mathbf{V} \approx \mathbf{W}\,\mathbf{H}$. In doing so, NMF approximates spectrogram data as a linear combination of prototypical frequencies or spectra (i.e., basis vectors) over time.

This process can be seen in Fig. 2.5, where a two-measure piano passage of "Mary Had a Little Lamb" is shown alongside a spectrogram and an NMF factorization. Notice how $\mathbf{W}$ captures the harmonic content of the three pitches of the passage and $\mathbf{H}$ captures the time onsets and gains of the individual notes. Also note that N_z is typically chosen manually or using a model selection procedure such as cross-validation, and N_f and N_t are a function of the overall recording length and STFT parameters (transform length, zero-padding size, and hop size).

This leads to two related interpretations of how NMF models spectrogram data. The first interpretation is that the columns of $\mathbf{V}$ (i.e., short time-segments of the mixture signal) are approximated as a weighted sum of basis vectors as shown in Fig. 2.6 and Eq. (2.9).

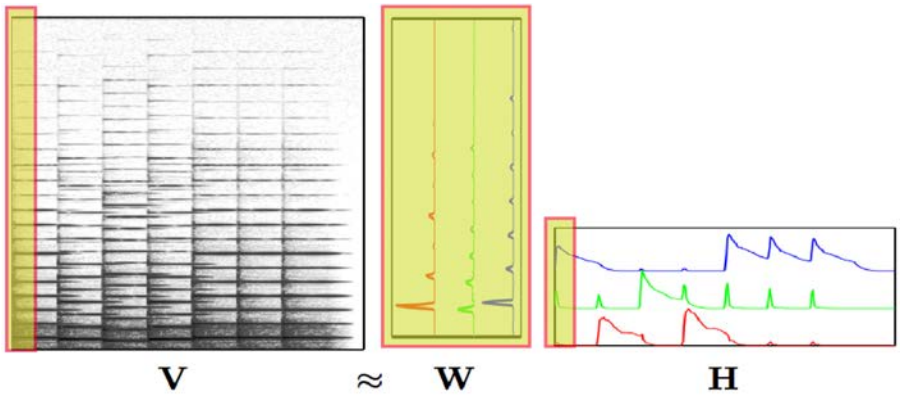


Figure 2.6: NMF interpretation I. The columns of $\mathbf{V}$ (i.e., short time-segments of the mixture signal) are approximated as a weighted sum or mixture of basis vectors $\mathbf{W}$.

$$\mathbf{V} \approx \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_{N_t} \\ | & | & & | \end{bmatrix} \approx \begin{bmatrix} \sum_{j=1}^{N_z} H_{j1}\,\mathbf{w}_j & \sum_{j=1}^{K} H_{j2}\,\mathbf{w}_j & \dots & \sum_{j=1}^{K} H_{jN_t}\,\mathbf{w}_j \end{bmatrix}. \tag{2.9}$$

The second interpretation is that the entire matrix $\mathbf{V}$ is approximated as a sum of matrix "layers," as shown in Fig. 2.7 and Eq. (2.10).

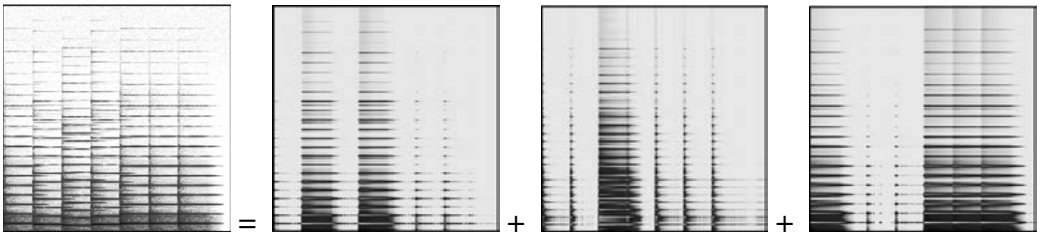


Figure 2.7: NMF interpretation II. The matrix $\mathbf{V}$ (i.e., the mixture signal) is approximated as a sum of matrix "layers."

$$
\begin{aligned}
\mathbf{V} &\approx \\
\begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_{N_t} \\ | & | & & | \end{bmatrix} &\approx \begin{bmatrix} | & | & & | \\ \mathbf{w}_1 & \mathbf{w}_2 & \dots & \mathbf{w}_{N_z} \\ | & | & & | \end{bmatrix} \begin{bmatrix} — & \mathbf{h}_1^{\mathrm{T}} & — \\ — & \mathbf{h}_2^{\mathrm{T}} & — \\ & \vdots & \\ — & \mathbf{h}_{N_z}^{\mathrm{T}} & — \end{bmatrix} \\
&\approx \mathbf{w}_1\,\mathbf{h}_1^{\mathrm{T}} + \mathbf{w}_2\,\mathbf{h}_2^{\mathrm{T}} + \ldots + \mathbf{w}_{N_z}\,\mathbf{h}_{N_z}^{\mathrm{T}}\,. \qquad (2.10)
\end{aligned}
$$

As we will see, both interpretations are useful and provide further insight into how NMF operates.

2.4.2 Optimization Formulation

To estimate the basis matrix $\mathbf{W}$ and the activation matrix $\mathbf{H}$ for a given input data matrix $\mathbf{V}$, NMF is formulated as an optimization problem. This is written as

$$
\begin{aligned}
&\underset{\mathbf{W},\mathbf{H}}{\arg\min} \;\; D(\mathbf{V} \,|\, \mathbf{W}\,\mathbf{H}) \\
&\text{subject to } \mathbf{W} \geq \mathbf{0}, \mathbf{H} \geq \mathbf{0},
\end{aligned} \qquad (2.11)
$$

where $D(\mathbf{V} \,|\, \mathbf{W}\,\mathbf{H})$ is an appropriately defined cost function between $\mathbf{V}$ and $\mathbf{W}\,\mathbf{H}$ and the inequalities $\geq$ are element-wise. It is also common to add additional equality constraints to require the columns of $\mathbf{W}$ to sum to one, which we enforce. When $D(\mathbf{V} \,|\, \mathbf{W}\,\mathbf{H})$ is additively separable, the cost function can be reduced to

$$
D(\mathbf{V} \,|\, \mathbf{W}\,\mathbf{H}) = \sum_{f=1}^{N_f} \sum_{t=1}^{N_t} d(V_{ft} | \, [\mathbf{W}\,\mathbf{H}]_{ft}). \qquad (2.12)
$$

where $[]_{ft}$ indexes its argument at row f and column t and $d(q|p)$ is a scalar cost function measured between q and p.

Popular cost functions include the Euclidean distance metric, Kullback-Lieber (KL) divergence, and the Itakura-Saito (IS) divergence. Both the KL and IS divergences have been found to be well suited for audio purposes. In this work, we focus on the case where

$d(q|p)$ is the generalized (non-normalized) KL divergence

$$d_{\mathrm{KL}}(q|p) = q \ln \frac{q}{p} - q + p, \tag{2.13}$$

because of its correspondence to the PLVMs we discuss later. This results in the following optimization formulation

$$\begin{aligned} &\arg\min_{\mathbf{W},\mathbf{H}} \sum_{f=1}^{N_f} \sum_{t=1}^{N_t} -V_{ft} \ln[\mathbf{W}\,\mathbf{H}]_{ft} + [\mathbf{W}\,\mathbf{H}]_{ft} + const. \\ &\text{subject to } \mathbf{W} \geq \mathbf{0}, \mathbf{H} \geq \mathbf{0}\,. \end{aligned} \tag{2.14}$$

Given this formulation, we notice that the problem is not convex in $\mathbf{W}$ and $\mathbf{H}$, limiting our ability to find a globally optimal solution to Eq. (2.14). It is, however, biconvex or independently convex in $\mathbf{W}$ for a fixed value of $\mathbf{H}$ and convex in $\mathbf{H}$ for a fixed value of $\mathbf{W}$, motivating the use of iterative numerical methods to estimate locally optimal values of $\mathbf{W}$ and $\mathbf{H}$, as discussed below.

2.4.3 Parameter Estimation

To solve Eq. (2.14), we must use an iterative numerical optimization technique and hope to find a locally optimal solution. Gradient descent methods are the most common and straightforward for this purpose, but typically are slow to converge. Other methods such as Newton's method, interior-point methods, conjugate gradient methods, and similar [74] can converge faster, but are typically much more complicated to implement, motivating alternative approaches.

The most popular alternative that has been proposed is by Lee and Seung [81, 13] and consists of a fast, simple, and efficient *multiplicative* gradient descent-based optimization procedure. The method works by breaking down the larger optimization problem into two subproblems and iteratively optimizes over $\mathbf{W}$ and then $\mathbf{H}$, back and forth, given an initial feasible solution. The approach monotonically decreases the optimization objective for both the KL divergence and Euclidean cost functions and converges to a local stationary point.

Algorithm 4 KL-NMF Parameter Estimation

Procedure KL-NMF (
$\mathbf{V} \in \mathbf{R}_{+}^{N_f \times N_t}$, // input data matrix
N_z // number of basic vectors
)
initialize: $\mathbf{W} \in \mathbf{R}_{+}^{N_f \times N_z}$, $\mathbf{H} \in \mathbf{R}_{+}^{N_z \times N_t}$ (e.g., random values > 0)
repeat
optimize over $\mathbf{W}$

$$\mathbf{W} \leftarrow \mathbf{W} \odot \frac{(\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}})\mathbf{H}^{\mathrm{T}}}{\mathbf{1}\,\mathbf{H}^{\mathrm{T}}} \tag{2.15}$$

optimize over $\mathbf{H}$

$$\mathbf{H} \leftarrow \mathbf{H} \odot \frac{\mathbf{W}^{\mathrm{T}}(\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}})}{\mathbf{W}^{\mathrm{T}}\,\mathbf{1}} \tag{2.16}$$

until convergence
return: $\mathbf{W}$ and $\mathbf{H}$

The approach is justified using the machinery of Majorization-Minimization (MM) algorithms [82]. MM algorithms are closely related to Expectation-Maximization (EM) algorithms, the latter of which we discuss in more detail below. In general, MM algorithms operate by approximating an optimization objective with a lower bound auxiliary function. The lower bound is then maximized instead of the original function, which is usually more difficult to optimize.

Algorithm 4 shows the complete iterative numerical optimization procedure applied to Eq. (2.14) with the KL divergence, where the division is element-wise, $\odot$ is an element-wise multiplication, and $\mathbf{1}$ is a vector or matrix of ones with appropriately defined dimensions [13]. For a full derivation of KL-NMF, please see the work of Lee and Seung [13].

Since Lee and Seung, numerous other researchers, including Dhillon and Sra [83], Cichocki et al. [84], Févotte et al. [16, 85], and Yang [86], have generalized this multiplicative update approach for different cost functions and scenarios. In most cases, the different cost functions require further proofs of convergence and use an MM, EM, or alternative approach for this purpose. When extending NMF to incorporate user-annotated painting annotations, we will take an EM approach, albeit with a modified optimization objective.

2.4.4 Probabilistic Interpretation

In addition to the linear algebra-based interpretation of NMF, we can also associate a probabilistic model to NMF. While the probabilistic model changes according to the cost function used, it is still useful. We discuss the probabilistic model that follows from using KL divergence as the cost function. The initial model was first discussed by Lee and Seung and then expanded upon in the work of Virtanen et al. [87], which more clearly outlines the model for audio purposes.

The probabilistic model is as follows: Each pixel of spectrogram data (e.g., V_{ft}) is generated by sampling independently and identically distributed (i.i.d.) Poisson random variables $V_{ft} \sim Poisson(\mu_{ft})$ with means $\mu_{ft} = [\mathbf{W}\,\mathbf{H}]_{ft}$. The probability mass function of a Poisson distributed random variable is given by

$$p(x; \mu_{ft}) = e^{-\mu_{ft}} \frac{\mu_{ft}^{x}}{x!}. \tag{2.17}$$

While simply stated, this model carries several layers of interpretability.

In particular, this formulation implies that the spectrogram data is quantized and made up of discrete random events, and a single event is the occurrence of a single 'unit' of sound energy or quantum. In addition, while not immediately obvious, this also implies that each pixel V_{ft} is generated by a latent variable model, where different latent components (e.g., basis) are mixed together (e.g., weights) to reconstruct the data. Because we more carefully outline latent variable models below, we limit our discussion of the full latent variable model here.

While this interpretation helps establish a probabilistic intuition behind NMF, it is also unsettling. This is because each element or pixel of a spectrogram matrix is continuously valued and we are modeling them as discrete random variables. To resolve this issue, we can assume that the spectrogram data we are modeling is discretized by multiplying the data with an increasingly large scale factor and then quantized [88]. While this assumption is required to theoretically justify the probabilistic model, in practice, this assumption is rarely a problem and is typically not addressed further. We should note, however, that this assumption will also be required for the probabilistic latent variable models discussed in Section 2.5.

Finally, to show that this interpretation is equivalent to a KL divergence cost function for NMF, we can perform maximum-likelihood estimation to fit this probabilistic model to our data. We do so by forming the likelihood as a function of our model parameters $\mathbf{W}$ and $\mathbf{H}$ and parameterized by our data $\mathbf{V} \in \mathrm{R}^{N_f \times N_t}$. This results in

$$L(\mathbf{W}, \mathbf{H} \,|\, \mathbf{V}) = \prod_{f=1}^{N_f} \prod_{t=1}^{N_t} e^{-[\mathbf{W}\,\mathbf{H}]_{ft}} \frac{[\mathbf{W}\,\mathbf{H}]_{ft}^{V_{ft}}}{V_{ft}!} \tag{2.18}$$

We can then form the log-likelihood by taking the logarithm and by doing some slight rearranging, we get

$$\begin{aligned} \mathcal{L}(\mathbf{W}, \mathbf{H} \,|\, \mathbf{V}) &= \ln \prod_{f=1}^{N_f} \prod_{t=1}^{N_t} e^{-[\mathbf{W}\,\mathbf{H}]_{ft}} \frac{[\mathbf{W}\,\mathbf{H}]_{ft}^{V_{ft}}}{V_{ft}!} \\ &= \sum_{f=1}^{N_f} \sum_{t=1}^{N_t} -[\mathbf{W}\,\mathbf{H}]_{ft} + (\ln[\mathbf{W}\,\mathbf{H}]_{ft}^{V_{ft}}) - \ln(V_{ft}!) \\ &= \sum_{f=1}^{N_f} \sum_{t=1}^{N_t} V_{ft} \ln[\mathbf{W}\,\mathbf{H}]_{ft} - [\mathbf{W}\,\mathbf{H}]_{ft} + const. \end{aligned} \tag{2.19}$$

When we formally write this as a maximization problem, we get

$$\begin{aligned} &\arg\max_{\mathbf{W},\mathbf{H}} \quad \sum_{f=1}^{N_f} \sum_{t=1}^{N_t} V_{ft} \ln([\mathbf{W}\,\mathbf{H}]_{ft}]) - [\mathbf{W}\,\mathbf{H}]_{ft} + const. \\ &\text{subject to} \quad \mathbf{W} \geq \mathbf{0}, \mathbf{H} \geq \mathbf{0}\,. \end{aligned} \tag{2.20}$$

This optimization formulation is equivalent to solving Eq. (2.14) and therefore shows the equivalence of the probabilistic model formulation to the linear algebra-based KL divergence cost function formulation. This is not the end of our discussion on probabilistic models, however, as we now discuss the use of more explicit probabilistic latent variable models for source separation.

2.5 Probabilistic Latent Variable Models

In this section, we describe three common probabilistic latent variable models used for source separation in Section 2.5.1. We then focus on one model in particular, formulate an optimization problem to estimate the parameters of the model in Section 2.5.2, and derive an iterative algorithm to solve the optimization problem in Section 2.5.3. In Chapter 4, we will build off of both the model formation and parameter estimation procedure when we discuss our proposed separation algorithm.

2.5.1 Models

The three probabilistic latent variable models we discuss can be viewed as variants of probabilistic latent component analysis (PLCA) developed by Smaragdis, Raj, and Shashanka [17, 18, 89, 90, 19, 91]. PLCA was developed as a straightforward extension of probabilistic latent semantic indexing (PLSI) or equivalently probabilistic latent semantic analysis (PLSA) [92] for arbitrary dimensions and is applied to modeling audio. Note, PLSI/PLSA is a popular probabilistic model used in the natural language processing (NLP) community for modeling topics, words, and documents.

The general PLCA model is defined as a factorized generative probabilistic latent variable model of the form

$$p(\mathbf{x}) = \sum_{z} p(z) \prod_{j=1}^{N} p(x_j|z), \tag{2.21}$$

where $p(\mathbf{x})$ is an N-dimensional probability distribution over observed data $\mathbf{x} = x_1, x_2, \ldots, x_N$, $p(z)$ is the probability distribution over a latent variable z, $p(x_j|z)$ are one-dimensional probability distributions, and all parameters of the distributions $\boldsymbol{\Theta}$ are implicit in the shorthand notation. The goal of using this model is to uncover a hidden or latent structure found within observed data, which is done by estimating the model parameters of the $p(z)$ and $p(x_j|z)$ distributions.

When modeling sound, typically discrete, one- or two-dimensional variants are employed and used to model audio spectrogram data. Each variant is defined by its own generative process and conditional independence assumptions. In many cases, the models are found to be very similar in practice, but have slight differences that make them useful

for different scenarios.

What is common across all variations is that they each interpret observed spectrogram data in a unique way. Rather than directly modeling the rows or columns of a spectrogram $\mathbf{V}$ with continuous distributions (e.g., Gaussians), discrete multinomial distributions are used to model the continuous valued data.

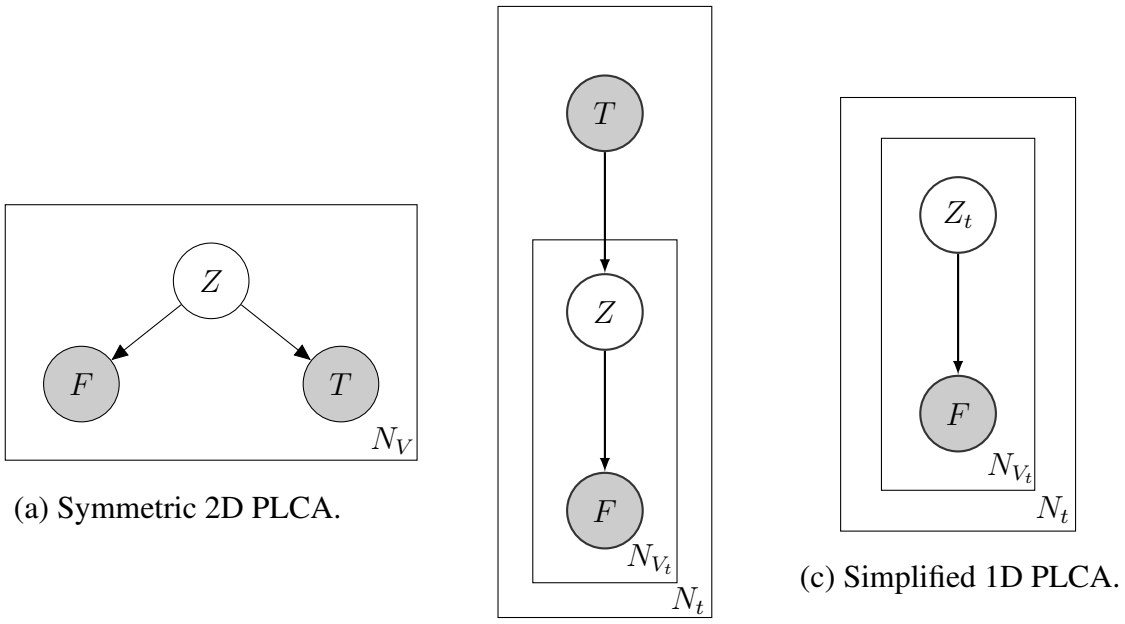
A multinomial distribution is defined by the probability mass function

$$p(x_1, x_2, \ldots, x_k; n, \boldsymbol{\pi}) = \begin{cases} \frac{n!}{x_1! x_2! \ldots x_k!} \pi_1^{x_1} \pi_2^{x_2} \ldots \pi_k^{x_k} & \sum_{i=1}^{k} x_i = n \\ 0 & \text{otherwise} \end{cases} \tag{2.22}$$

where n is the total number of independent trials of a random experiment with k possible outcomes; $x_1, x_2, \ldots, x_k$ are non-negative integers that represent the number of the outcomes of each of the k categories; the factorial terms are normalizing constants to make a proper probability; and $\boldsymbol{\pi} = (\pi_1, \pi_2, \cdots, \pi_k)$ denote the probability of each of the k outcomes, where $\pi_1, \pi_2, \cdots, \pi_k \geq 0$ and $\sum_{i=1}^{k} \pi_i = 1$. In essence, the multinomial distribution assigns the probability of any combination of outcomes for n independent trials. A random variable X that is distributed according to a multinomial distribution is denoted $X \sim Multinomial(n, \boldsymbol{\pi})$.

In addition to the definition of a multinomial distribution, we define the fundamental unit of discrete data that we model as an energy *quantum*, akin to NMF. An energy quantum is defined to be a finite unit of sound energy indexed by frequency $\{1, \ldots, N_f\}$ and/or time $\{1, \ldots, N_t\}$. We also define observed *data* as a collection of quanta in the form of a spectrogram matrix $\mathbf{V} \in \mathrm{R}_{+}^{N_f \times N_t}$. Carefully note, we do not observe individual quanta, but only sums of quanta indexed by time and frequency. The matrix $\mathbf{V}$ can also be thought of as a matrix of co-occurence values indexed by time and frequency.

As we will see, multinomial distributions (or similarly Poisson distributions) are a convenient way of modeling correlation information found within audio spectrogram data. Modeling correlation information helps us characterize distinct sound sources, discriminate one sound from another, and eventually tease apart multiple sources within a single mixture. Because of this, such models typically do not have a valid and/or accurate physical interpretation and can be the subject of concern to many. This, however, should be considered a philosophical difference in approach, rather than a limitation.



(a) Symmetric 2D PLCA.

(b) Asymmetric 2D PLCA.

(c) Simplified 1D PLCA.

Figure 2.8: Plate diagrams or graphical representations for three common latent variable models used for single-channel source separation. Observed random variables are represented by shaded circles, hidden random variables are represented by clear circles, repeated draws are represented by a surrounding box, and conditional dependencies are represented by directed arrows.

Symmetric Two-Dimensional Model

The first variant that we look at is the symmetric two-dimensional PLCA model [18]. In this case, we model each spectrogram data $\mathbf{V}$ via

$$p(f,t) = \sum_z p(z)p(f|z)p(t|z), \tag{2.23}$$

where $p(f,t)$ is a two-dimensional multinomial distribution representing the probability of sound at a particular time-frequency point, $p(f|z)$ is a multinomial distribution representing frequency basis vectors or dictionary elements for each source, $p(t|z)$ is a multinomial distribution representing gains, weighting, or activations indexed by time, and $p(z)$ is a multinomial distribution representing the overall weighting of each latent variable outcome.

A graphical representation of this model is shown in Fig. 2.8(a).

The complete generative process of this model used to create a spectrogram $\mathbf{V}$ is specified via:

1. For $n = 1, \ldots, N_V$ times, where $N_V = \sum_f \sum_t V_{ft}$,
   (a) Generate a latent variable $z^{(n)} \sim Multinomial(N_V, \boldsymbol{\sigma})$.
   (b) Generate a frequency $f^{(n)}|z^{(n)} \sim Multinomial(N_V, \mathbf{w}_z)$.
   (c) Generate a time $t^{(n)}|z^{(n)} \sim Multinomial(N_V, \boldsymbol{\pi}_z)$.
2. Set V_{ft} equal to the count of the occurrence of each outcomes value pair (f, t).

The multinomial parameter $\boldsymbol{\sigma}$ is a $N_z \times 1$-element non-negative vector, where all elements sum to one, $\mathbf{W} = [\mathbf{w}_1 \ldots \mathbf{w}_{N_z}]$ is an $N_f \times N_z$-element non-negative matrix where each column sums to one, and $\mathbf{\Pi} = [\boldsymbol{\pi}_1 \ldots \boldsymbol{\pi}_{N_z}]^{\mathrm{T}}$ is an $N_t \times N_z$-element non-negative matrix where each row sums to one. Collectively, we define all parameters of the complete model as $\mathbf{\Theta} = \{\mathbf{W}, \mathbf{\Sigma}, \mathbf{\Pi}\}$, where $\mathbf{\Sigma} = diag(\boldsymbol{\sigma})$.

Also note, the parameters of the multinomial distributions themselves represent normalized probability values. Because of this, it is common to use the overloaded notation $\mathbf{\Theta} = \{p(f|z), p(z), p(t|z)\}$ to represent the parameters $\mathbf{\Theta} = \{\mathbf{W}, \mathbf{\Sigma}, \mathbf{\Pi}\}$ respectively. This notation is often more straightforward and is now standard convention, but it can sometimes be confusing. We will try to be clear when use either notation throughout.

Fig. 2.9 depicts the use of this model to approximate a spectrogram of a piano playing "Mary Had a Little Lamb." In this case, three frequency distributions are used to model the three pitches (E, D, and C) of the two-measure passage. Notice how the frequency distributions $p(f|z)$ ideally capture the harmonic structure of the pitches and the corresponding elements of $p(t|z)$ and $p(z)$ capture the timing and amplitude levels of each note. Also notice the similarity between this figure and Fig. 2.5.

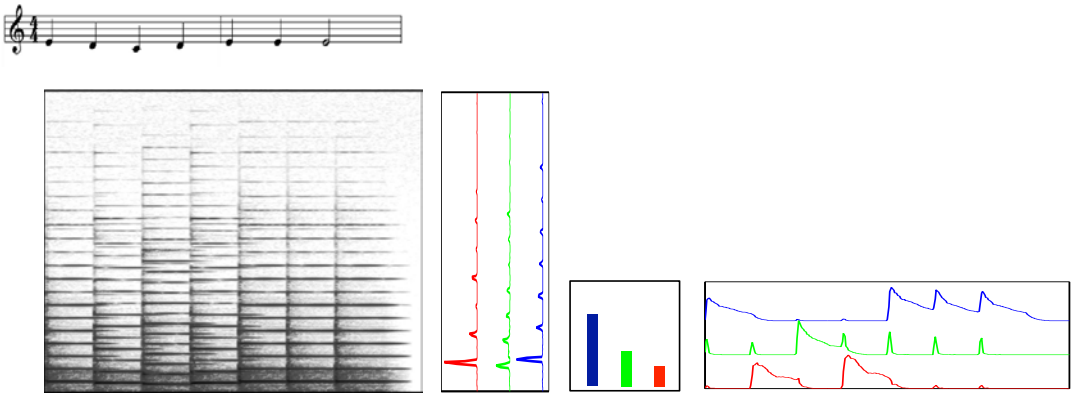


Figure 2.9: Symmetric two-dimensional PLCA of a piano performing "Mary Had a Little Lamb" for two measures with $N_z = 3$. Notice how the distribution $p(f|z)$ captures the harmonic content of the three distinct pitches of the passage and $p(z)$ and $p(t|z))$ capture the time onsets and gains of the individual notes.

Asymmetric Two-Dimensional Model

In the asymmetric two-dimensional PLCA model, a slight rearrangement of the symmetric factorization is used to approximate our spectrogram data $\mathbf{V}$. This model stems from the work of Raj and Smaragdis [17]. In this case, the factorized model is

$$p(f,t) = p(t) \sum_z p(f|z)p(z|t). \tag{2.24}$$

This model is very similar to the symmetric two-dimensional model discussed above, albeit with slightly different factors. In this model, $p(f,t)$ is a two-dimensional multinomial distribution representing the probability of sound energy at a given time-frequency point, $p(t)$ is a multinomial distribution representing the overall probability of sound energy at a given point in time, $p(f|z)$ is a multinomial distribution representing frequency basis vectors or dictionary elements for each source, and $p(z|t)$ is a multinomial distribution representing relative weighting of the latent component at a given time instance. A graphical representation of this model is illustrated in Fig. 2.8(b).

The complete generative process used to create a normalized spectrogram $\mathbf{V}$ is specified via:

1. For $t = 1, 2, \ldots, N_t$.

   (a) For $n = 1, \ldots, N_{V_t}$ times, where $N_{V_t} = \sum_f V_{ft}$,

      i. Generate a time $t^{(n)} \sim Multinomial(N_V, \boldsymbol{\sigma})$.

      ii. Generate a frequency $f^{(n)}|z^{(n)} \sim Multinomial(N_V, \mathbf{w}_z)$.

      iii. Generate a latent variable $z^{(n)}|t^{(n)} \sim Multinomial(N_V, \boldsymbol{\pi}_z)$.

2. Set V_{ft} equal to the count of the occurrence of each outcomes value pair (f, t).

The multinomial distribution parameter $\boldsymbol{\sigma}$ is a non-negative $N_t \times 1$ vector that sums to one, $\mathbf{W} = [\mathbf{w}_1 \ldots \mathbf{w}_{N_z}]$ is a $N_f \times N_z$ non-negative matrix with columns that sum to one, and $\boldsymbol{\Pi} = [\boldsymbol{\pi}_1 \ldots \boldsymbol{\pi}_{N_t}]$ is a non-negative $N_z \times N_t$-dimensional matrix with columns that sum to one. Collectively, we define all parameters of the complete model as $\boldsymbol{\Theta} = \{\mathbf{W}, \boldsymbol{\Sigma}, \boldsymbol{\Pi}\}$, where $\boldsymbol{\Sigma} = diag(\boldsymbol{\sigma})$.

As before, the parameters of the multinomial distributions themselves represent normalized probability values. Because of this, it is common to use the overloaded notation $\boldsymbol{\Theta} = \{p(f|z), p(z), p(z|t)\}$ to represent the parameters $\boldsymbol{\Theta} = \{\mathbf{W}, \boldsymbol{\Sigma}, \boldsymbol{\Pi}\}$, respectively.

Simplified One-Dimensional Model

In the simplified one-dimensional model developed by Raj and Smaragdis [17], each time frame $\mathbf{V}_t$ of the spectrogram $\mathbf{V}$ is modeled independently. As a result, the random variable representing time is eliminated and the data for each time slice $\mathbf{V}_t$ is approximated via

$$p(f_t) = \sum_{z_t} p(f_t|z_t)p(z_t). \tag{2.25}$$

In this model, $p(f_t)$ is a one-dimensional multinomial distribution representing the probability of sound energy at a given frequency point indexed by time t, $p(f_t|z_t)$ is a multinomial distribution representing frequency basis vectors or dictionary elements for each source, and $p(z_t)$ is a multinomial distribution representing weighting of the latent component z_t. The plate diagram of the model for one time-frame is shown Fig. 2.8(c).

The generative process used to create a complete spectrogram $\mathbf{V}$ is specified via:

1. For $t = 1, 2, \ldots, N_t$.

(a) For $n = 1, \ldots, N_{V_t}$ times, where $N_{V_t} = \sum_f V_{ft}$,

i. Generate a latent variable $z_t^{(n)} \sim Multinomial(N_V, \boldsymbol{\sigma})$.

ii. Generate a frequency $f_t^{(n)} | z_t^{(n)} \sim Multinomial(N_V, \mathbf{w}_z)$.

2. Set V_{ft} equal to the count of the occurrence of each outcomes value pair (f, t).

Note, each column of the spectrogram is treated separately. The multinomial distribution parameter $\boldsymbol{\sigma}$ is a N_{z_t}-dimensional non-negative vector that sums to one and $\mathbf{W} = [\mathbf{w}_1 \ldots \mathbf{w}_{N_z}]$ is a $N_f \times N_z$ non-negative matrix with columns that sum to one. Collectively, we define all parameters of the complete model as $\boldsymbol{\Theta} = \{\mathbf{W}, \boldsymbol{\Pi}\}$ or, alternatively, using the overloaded notation $\boldsymbol{\Theta} = \{p(f|z), p(z)\}$.

Comparison

When comparing these models, we can see that the biggest differences arise from how each model deals with time, resulting in contrasting conditional independence assumptions. In the symmetric two-dimensional factorization, both time and frequency are dealt with identically, resulting in a straightforward interpretation of each of the model factors. In the two-dimensional asymmetric and simplified one-dimensional factorization, however, time and frequency are dealt with differently.

In the two-dimensional asymmetric factorization, the distribution relating the latent component and time (i.e., $p(z|t)$) is normalized over the values of the latent component and not time. For our purposes, this results in a less intuitive factorization because the distribution $p(z|t)$ no longer represents the amplitude envelope of each component as it does in the symmetric factorization because it is normalized in each time frame. In the one-dimensional factorization, an additional independence is added (i.e., each column $\mathbf{V}_t$ of a spectrogram is generated independently) and the distribution relating time and the latent component is eliminated.

While this is useful for some applications [93, 21], it is beneficial for us to maintain a dependence between time and the latent component. Because of this, we focus on the two-dimensional symmetrical factorization in the remaining sections and chapters.

2.5.2 Optimization Formulation

Given the symmetric two-dimensional PLCA model, we now must fit the parameters of our model $\mathbf{\Theta}$ to data $\mathbf{V}$. While maximum-likelihood (ML) is the standard method used to estimate parameters of probabilistic models, the latent variables in our model make direct ML difficult. More specifically, when we formulate the log-likelihood of our model as a function of the parameters given our data $\mathbf{V}$, we get

$$\begin{aligned}
\mathcal{L}(\mathbf{\Theta}|\,\mathbf{V}) &= \ln p(\mathbf{V}\,|\mathbf{\Theta}) \\
&= \ln p(V_{11}, V_{12}, \ldots, V_{ft}|\mathbf{\Theta}) \\
&= \ln \frac{(\sum_f \sum_t V_{ft})!}{V_{11}!V_{12}!\ldots V_{ft}!} \prod_{f=1}^{N_f}\prod_{t=1}^{N_t} p(f,t)^{V_{ft}} \\
&= \ln \frac{(\sum_f \sum_t V_{ft})!}{V_{11}!V_{12}!\ldots V_{ft}!} \prod_{f=1}^{N_f}\prod_{t=1}^{N_t} \big[\sum_z p(f|z)p(z)p(t|z)\big]^{V_{ft}} \\
&= \sum_{f=1}^{N_f}\sum_{t=1}^{N_t} V_{ft} \ln \big[\sum_z p(f|z)p(z)p(t|z)\big] \qquad (2.26)
\end{aligned}$$

where the model parameters, in this case, are represented by $\mathbf{\Theta} = \{p(f|z), p(t|z), p(z)\}$ according to standard convention.

When we try to maximize this function, problems arise because of the logarithm of the sum. To mitigate this problem, we use an expectation-maximization (EM) algorithm. EM algorithms are the canonical optimization method for estimating model parameters of latent variable models. When we use EM, we form the log-likelihood, and then form an auxiliary function $\mathcal{F}(q, \mathbf{\Theta})$ that lower bounds the log-likelihood. We then maximize the lower bound, instead of directly optimizing the log-likelihood [94, 77].

In this way, we indirectly maximize the log-likelihood function and are able to solve an easier optimization problem. As noted in the literature [94, 77], this process guarantees the parameter estimates $\mathbf{\Theta}$ to monotonically increase the lower bound, and consequently increase the likelihood until convergence to a local stationary point. The complete generalized EM is presented in Algorithm 5 and follows from the description of Bishop [77]. In our notation, all observed random variables are denoted by $\mathbf{X}$, all unobserved random

variables are denoted by $\mathbf{Z}$, and all model parameters are denoted by $\mathbf{\Theta}$. We also define the posterior distribution $p(\mathbf{Z} \mid \mathbf{X}, \mathbf{\Theta})$ to be the distribution over the latent variables $\mathbf{Z}$ given the observed variables $\mathbf{X}$ and model parameters $\mathbf{\Theta}$.

Also note, there are alternative interpretations and meta-algorithms for EM algorithms (e.g., computing the expected value of the complete-data log-likelihood with respect to the posterior distribution of the unobserved variables). In such cases, it is common for the expectation step to only require step 4(a) (i.e., computing the posterior distribution $p(\mathbf{Z} \mid \mathbf{X}, \mathbf{\Theta})$) and not step 4(b) because $q(\mathbf{Z})$ is optimal when equal to the posterior. When we discuss our proposed algorithmic modifications to PLCA to incorporate user-feedback, however, we will require this formulation and will add additional optimization constraints to Eq. (2.30).

Algorithm 5 The Generalized Expectation-Maximization Algorithm

1. Formulate the log-likelihood of your model, knowing the model parameters $\mathbf{\Theta}$, the observed data $\mathbf{X}$, and the latent variables $\mathbf{Z}$,

$$\mathcal{L}(\mathbf{\Theta}|\,\mathbf{X}) \;=\; \ln p(\mathbf{X}\,|\mathbf{\Theta}) = \ln \sum_{\mathbf{Z}} p(\mathbf{X},\mathbf{Z}\,|\mathbf{\Theta}).$$

2. Form an auxiliary function that lower bounds the log-likelihood via

$$\begin{aligned}\mathcal{L}(\mathbf{\Theta}|\,\mathbf{X}) &= \ln p(\mathbf{X}\,|\mathbf{\Theta}) \\ &= \mathcal{F}(q,\mathbf{\Theta}) + \mathrm{KL}(q||p),\end{aligned} \tag{2.27}$$

where
$$\mathcal{F}(q,\mathbf{\Theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln\left\{\frac{p(\mathbf{X},\mathbf{Z}\,|\mathbf{\Theta})}{q(\mathbf{Z})}\right\}, \tag{2.28}$$

$$\begin{aligned}\mathrm{KL}(q||p) &= \mathrm{KL}(q(\mathbf{Z}) \,||\, p(\mathbf{Z}\,|\,\mathbf{X},\mathbf{\Theta})) \\ &= -\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln\left\{\frac{p(\mathbf{Z}\,|\,\mathbf{X},\mathbf{\Theta})}{q(\mathbf{Z})}\right\},\end{aligned} \tag{2.29}$$

$q(\mathbf{Z})$ or q is a discrete, free distribution, $\mathrm{KL}(q||p)$ is the Kullback-Leibler divergence and $\mathcal{F}(q,\mathbf{\Theta})$ is the lower bound as a result of $\mathrm{KL}(q||p)$ being non-negative.

3. Provide an initial guess of the model parameters $\mathbf{\Theta}$.

4. Maximize the lower bound via the following two-stage coordinate ascent procedure until convergence. The superscript ^{n} denotes iteration number.

   Expectation Step

   (a) Evaluate $p(\mathbf{Z}\,|\,\mathbf{X},\mathbf{\Theta})$, the posterior distribution over the latent variables $\mathbf{Z}$ given the observed variables $\mathbf{X}$ and model parameters Θ.

   (b) Maximize the lower bound $\mathcal{F}(q,\mathbf{\Theta})$ or equivalently minimize $\mathrm{KL}(q||p)$ with respect to q

$$\begin{aligned}q^{n+1} &= \arg\max_{q} \mathcal{F}(q,\mathbf{\Theta}^n) \\ &= \arg\min_{q}\ \mathrm{KL}(q||p).\end{aligned} \tag{2.30}$$

   Maximization Step

   (c) Maximize the lower bound with respect to $\mathbf{\Theta}$

$$\mathbf{\Theta}^{n+1} \;=\; \arg\max_{\mathbf{\Theta}} \mathcal{F}(q^{n+1},\mathbf{\Theta}). \tag{2.31}$$

2.5.3 Parameter Estimation

When we apply the above procedure to estimate the parameters of our model, we first form the lower bound auxiliary function $\mathcal{F}(q, \mathbf{\Theta})$ resulting in

$$\mathcal{F}(q, \mathbf{\Theta}) \;=\; \sum_{f=1}^{N_f} \sum_{t=1}^{N_t} V_{ft} \Big[\sum_{z} q(z) \ln \big[p(z) p(f|z) p(t|z) \big] \Big] + const. \tag{2.32}$$

Given the lower bound, we must derive our E and M step update procedures. To compute the E step, we must first maximize $\mathcal{F}(q, \mathbf{\Theta})$ with respect to q. Fortunately, we know this is maximized when q is equal to the posterior distribution $q = p(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{\Theta})$. So, instead of explicitly maximizing $\mathcal{F}(q, \mathbf{\Theta})$ with respect to q, all we have to do is compute the posterior distribution of our model using Bayes' theorem, resulting in

$$p(z|f,t) \;\leftarrow\; \frac{p(z)p(f|z)p(t|z)}{\sum_{z'} p(z')p(f|z')p(t|z')}. \tag{2.33}$$

This completes the E step update.

Then, given this posterior, we must compute the M step update. To do so, we must maximize $\mathcal{F}(q, \mathbf{\Theta})$ with respect to the model parameters $\mathbf{\Theta}$. In our case, we can perform this maximization analytically. To do so, we first simplify the lower bound via

$$\begin{aligned}
\mathcal{F}(q, \mathbf{\Theta}) \;&=\; \sum_{f=1}^{N_f} \sum_{t=1}^{N_t} V_{ft} \Big[\sum_{z} q(z|f,t) \ln \big[p(z) p(f|z) p(t|z) \big] \Big] + const. \\
&=\; \sum_{f=1}^{N_f} \sum_{t=1}^{N_t} V_{ft} \Big[\sum_{z} p(z|f,t) \ln \big[p(z) p(f|z) p(t|z) \big] \Big] + const. \\
&=\; \sum_{f=1}^{N_f} \sum_{t=1}^{N_t} V_{ft} \Big[\sum_{z} p(z|f,t) \big[\ln p(z) + \ln p(f|z) + \ln p(t|z) \big] \Big] + const.
\end{aligned} \tag{2.34}$$

Note, this final lower bound in Eq. (2.34) is equivalent to the expected value of the log-likelihood with respect to the posterior distribution.

Given Eq. (2.34), we maximize this with respect to the model parameters $\mathbf{\Theta} = \{p(z), p(f|z), p(t|z)\}$. This requires the use of Lagrange multipliers, $\pi, \boldsymbol{\kappa}$, and $\boldsymbol{\nu}$, to

enforce proper probability distributions, resulting in the Lagrangian

$$\begin{aligned}\Psi(\boldsymbol{\Theta}, \pi, \boldsymbol{\kappa}, \boldsymbol{\nu}) &= \sum_f \sum_t V_{ft} \Big[\sum_z p(z|f,t) \ln p(z) \Big] + \pi \Big(1 - \sum_z p(z) \Big) + \\ & \sum_f \sum_t V_{ft} \Big[\sum_z p(z|f,t) \ln p(f|z) \Big] + \sum_z \kappa_z \Big(1 - \sum_f p(f|z) \Big) + \\ & \sum_f \sum_t V_{ft} \Big[\sum_z p(z|f,t) \ln p(t|z) \Big] + \sum_z \nu_z \Big(1 - \sum_f p(t|z) \Big). \quad (2.35)\end{aligned}$$

Maximizing the Lagrangian $\Psi(\boldsymbol{\Theta}, \pi, \boldsymbol{\kappa}, \boldsymbol{\nu})$ with respect to $p(z)$, $p(f|z)$, and $p(t|z)$ and setting the results to zero, yields

$$\sum_f \sum_t V_{ft} p(z|f,t) - \pi p(z) = 0 \quad (2.36)$$

$$\sum_t V_{ft} p(z|f,t) - \kappa_z p(f|z) = 0 \quad (2.37)$$

$$\sum_f V_{ft} p(z|f,t) - \nu_z p(t|z) = 0. \quad (2.38)$$

Eliminating the Lagrange multipliers then yields the final M step update equations

$$p(z) = \frac{\sum_f \sum_t V_{ft} p(z|f,t)}{\sum_{z'} \sum_{f'} \sum_{t'} V_{f't'} p(z|f',t')} \quad (2.39)$$

$$p(f|z) = \frac{\sum_t V_{ft} p(z|f,t)}{\sum_{f'} \sum_{t'} V_{f't'} p(z|f',t')} \quad (2.40)$$

$$p(t|z) = \frac{\sum_f V_{ft} p(z|f,t)}{\sum_{f'} \sum_{t'} V_{f't'} p(z|f',t')}. \quad (2.41)$$

This completes the M step update.

With the E and M steps together, we write the complete iterative two-stage EM procedure in Algorithm 6. In this case, we again use the shorthand notation $p(z), p(f|z)$, and $p(t|z)$ for the model parameters. Note, we denote the posterior distribution via $q(z|f,t)$ as opposed to $p(z|f,t)$ for future purposes. Also notice, both E and M steps are solved in

closed form.

Algorithm 6 PLCA Parameter Estimation

Procedure PLCA (

$\mathbf{V} \in \mathbf{R}_{+}^{N_f \times N_t}$, // observed normalized data

N_z // number of latent variable outcomes

)

initialize: feasible $p(z)$, $p(f|z)$, and $p(t|z)$

repeat

expectation step

for all z, f, t **do**

$$q(z|f,t) \leftarrow \frac{p(z)p(f|z)p(t|z)}{\sum_{z'} p(z')p(f|z')p(t|z')} \tag{2.42}$$

end for

maximization step

for all z, f, t **do**

$$p(f|z) \leftarrow \frac{\sum_t V_{ft} q(z|f,t)}{\sum_{f'} \sum_{t'} V_{f't'} q(z|f',t')} \tag{2.43}$$

$$p(t|z) \leftarrow \frac{\sum_f V_{ft} q(z|f,t)}{\sum_{f'} \sum_{t'} V_{f't'} q(z|f',t')} \tag{2.44}$$

$$p(z) \leftarrow \frac{\sum_f \sum_t V_{ft} q(z|f,t)}{\sum_{z'} \sum_{f'} \sum_{t'} V_{f't'} q(z'|f',t')} \tag{2.45}$$

end for

until convergence

return: $p(f|z)$, $p(t|z)$, $p(z)$, and $q(z|f,t)$

2.5.4 Relationship Between NMF and PLCA

Given both the NMF and PLCA probabilistic models and corresponding parameter estimation algorithms, it is useful to establish the relationship between the two approaches. This is because both methods are used for the same purpose, have similar probabilistic interpretations, and have similar parameter estimation algorithms. There are slight differences, however, and it is useful to clarify.

So, to compare the methods, we first analyze how NMF and PLCA both model spectrogram data $\mathbf{V}$. In the NMF probabilistic model, the spectrogram $\mathbf{V}$ is modeled as i.i.d. Poisson distributions with mean $\mu_{ft} = [\mathbf{W}\,\mathbf{H}]_{ft}$ for each element V_{ft}. In the symmetric two-dimensional PLCA model, the spectrogram $\mathbf{V}$ is modeled according to an i.i.d. generative process of multinomial distributed random variables.

While these two models seem at odds, they are intimately related. We see the relation when we analyze the probability of i.i.d. Poisson random variables conditioned on their sum N. When we do so, we see that

$$p(V_{11}, V_{12}, \ldots, V_{ft} | Y = N) \sim Multinomial(N, \mathbf{\Theta}) \tag{2.46}$$

where $V_{11}, V_{12}, \ldots, V_{ft}$ are i.i.d. Poisson random variables, following the derivation from Steel [95]. That is to say, the probability of i.i.d. Poisson random variables conditioned on their sum is a multinomial distribution and $N = \sum_f \sum_t V_{ft}$. As a result, we see that the NMF Poisson model and the PLCA multinomial models are identical, except for a conditioning on the overall number of events N. For a full derivation of this relation, please see Appendix B.

We also see the connection between the two probabilistic models when we derive the log-likelihood of our PLCA model using the alternative model parameter notation

$\boldsymbol{\Theta} = \{\mathbf{W}, \boldsymbol{\Sigma}, \boldsymbol{\Pi}\}$. In this case, we derive the log-likelihood as

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\Theta}|\,\mathbf{V}) &= \ln p(\mathbf{V}\,|\boldsymbol{\Theta}) \\
&= \ln p(V_{11}, V_{12}, \ldots, V_{ft}|\boldsymbol{\Theta}) \\
&= \ln \frac{(\sum_f \sum_t V_{ft})!}{V_{11}!V_{12}!\ldots V_{ft}!} \prod_{f=1}^{N_f}\prod_{t=1}^{N_t} \big[\sum_z W_{fz}\Sigma_z\Pi_{tz}\big]^{V_{ft}} \\
&= \sum_{f=1}^{N_f}\sum_{t=1}^{N_t} V_{ft} \ln\big[\sum_z W_{fz}\Sigma_z\Pi_{tz}\big] + const. \\
&= \sum_{f=1}^{N_f}\sum_{t=1}^{N_t} V_{ft} \ln[\mathbf{W}\,\boldsymbol{\Sigma}\boldsymbol{\Pi}]_{ft} + const. \\
&= \sum_{f=1}^{N_f}\sum_{t=1}^{N_t} V_{ft} \ln[\mathbf{W}\,\mathbf{H}]_{ft} + const., \qquad (2.47)
\end{aligned}$$

where in the last step, we set $\mathbf{H} = \boldsymbol{\Sigma}\boldsymbol{\Pi}$. As shown, this is identical to our NMF maximization objective Eq. (2.14), albeit with a missing additive term $\sum_{f=1}^{N_f}\sum_{t=1}^{N_t}[\mathbf{W}\,\mathbf{H}]_{ft}$. The missing term is the result of the assumption that we observe the overall sum of the spectrogram N as well as the fact that it is equal to one, as a result of the sum-to-one constraints on the model parameters.

In addition to comparing the two probabilistic models formulations, it is also useful to compare the MM KL-NMF parameter estimation algorithm and the EM PLCA parameter estimation algorithm. To do so, we can notate Algorithm 6 using matrix notation and rearrange the update questions to match those of Algorithm 4 [90]. Algorithm 7 shows the multiplicative update rules of PLCA where $\mathbf{W}$ is a matrix of probability values such that $p(f|z)$ is the f^{th} row and z^{th} column, $\mathbf{H}$ is a matrix of probability values such that $p(t|z)p(z)$ is the z^{th} row and t^{th} column, $\mathbf{1}$ is an appropriately sized matrix of ones, $\odot$ is element-wise multiplication, and the division is element-wise. Given proper initialization and normalization, these updates are near numerically identical to the multiplicative update equations for KL-NMF.

Given this close relationship, we can say that for practical purposes KL-NMF and PLCA are equivalent. As a result, we define Algorithm 8 to represent a general NMF/PLCA

algorithm using matrix notation. When we refer to NMF/PLVMs throughout the remaining duration of this work, we will refer to this procedure.

Algorithm 7 PLCA Parameter Estimation in Matrix Notation

Procedure PLCA(

$\mathbf{V} \in \mathbf{R}_{+}^{N_f \times N_t}$, // observed normalized data

N_z // number of basic vectors

)

initialize: $\mathbf{W} \in \mathbf{R}_{+}^{N_f \times N_z}$, $\mathbf{H} \in \mathbf{R}_{+}^{N_z \times N_t}$ (e.g., random probability values)

repeat

$$\mathbf{Z} \leftarrow \frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}} \tag{2.48}$$

$$\mathbf{W} \leftarrow \mathbf{W} \odot \frac{\mathbf{Z}\,\mathbf{H}^{\mathrm{T}}}{\mathbf{1}\,\mathbf{H}^{\mathrm{T}}} \tag{2.49}$$

$$\mathbf{H} \leftarrow \mathbf{H} \odot (\mathbf{W}^{\mathrm{T}}\,\mathbf{Z}) \tag{2.50}$$

until convergence

return: $\mathbf{W}$ and $\mathbf{H}$

Algorithm 8 General NMF/PLCA Parameter Estimation

Procedure NMF-PLCA(

$\mathbf{V} \in \mathbf{R}_{+}^{N_f \times N_t}$, // observed normalized data

N_z // number of basic vectors or latent components)

$(\mathbf{W}, \mathbf{H}) \leftarrow$ KL-NMF($\mathbf{V}$, N_z)

or

$(\mathbf{W}, \mathbf{H}) \leftarrow$ PLCA($\mathbf{V}$, N_z)

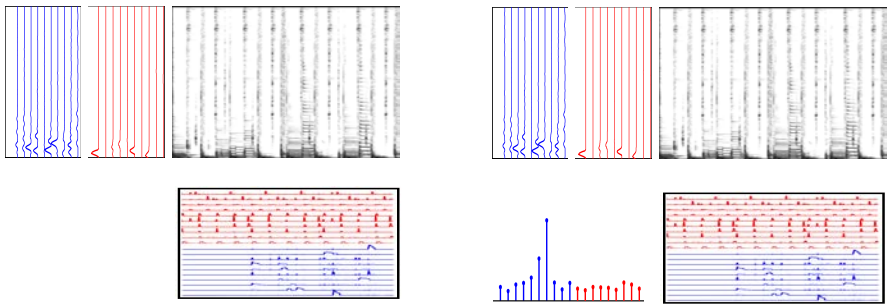
return: $\mathbf{W}$ and $\mathbf{H}$

2.6 Modeling Mixtures

Now that we have discussed how NMF/PLVM methods both model audio and perform parameter estimation given a spectrogram $\mathbf{V}$, we must discuss how these methods are used to model *mixture* sounds and eventually perform separation.

The general idea when modeling mixture sounds is to partition the parameters of a large NMF/PLVM model into distinct, non-overlapping groups as illustrated in Fig. 2.10. Once partitioned, we associate each group of parameters to one of the individual sources present in the mixture. Then, we estimate each group of parameters in one of three different ways: unsupervised, supervised, or semi-supervised parameter estimation.

In unsupervised parameter estimation, we estimate all parameters of our model simultaneously (i.e., $\mathbf{W}$ and $\mathbf{H}$). In supervised parameter estimation, we perform a series of unsupervised estimations to pre-learn each group of basis vectors for each source independently from isolated recordings, combine them together to form $\mathbf{W}$, and then estimate an unknown $\mathbf{H}$ from mixture data given the fixed, pre-computed $\mathbf{W}$. In semi-supervised separation, we perform a series of unsupervised estimations to pre-learn some of the basis vectors groups in $\mathbf{W}$ independently from isolated recordings, and then estimate the remaining unknown basis vectors of $\mathbf{W}$ and all elements of $\mathbf{H}$ from mixture data with the pre-learned portion of $\mathbf{W}$ held fixed. We discuss each of these procedures in more detail below.



$$\mathbf{V} \approx \mathbf{W}\,\mathbf{H} \approx \begin{bmatrix} \mathbf{W}_1 & \mathbf{W}_2 \end{bmatrix} \begin{bmatrix} \mathbf{H}_1^{\mathrm{T}} \\ \mathbf{H}_2^{\mathrm{T}} \end{bmatrix}$$

$$\mathbf{V} \approx \sum_{z \in Val(Z)} p(z)p(f|z)p(t|z) \approx \sum_{z \in Val(Z_1)} p(z)p(f|z)p(t|z) + \sum_{z \in Val(Z_2)} p(z)p(f|z)p(t|z)$$

Figure 2.10: (Left) Modeling a mixture sound (e.g., drums and bass) using NMF, where the model parameters are partitioned into two distinct groups. (Right) Modeling a mixture sound using PLCA, where the model parameters are analogously partitioned.

2.6.1 Unsupervised Parameter Estimation

When we estimate all of the parameters of an NMF/PLVM simultaneously, we call the procedure unsupervised parameter estimation. The term unsupervised is in reference to the fact that the model is attempting to learn or cluster distinct sound sources within a mixture without any supervision or additional information. We can interpret Fig. 2.10 as unsupervised parameter estimation, assuming all parameters are estimated simultaneously.

Unfortunately, however, when we simultaneously estimate all of the parameters of a model, it is difficult to obtain an ideal segmentation or grouping of the parameters so that distinct groupings explain distinct sound sources. This is particularly problematic when we allow each sound source to be explained by more than one basis vector or latent component. This is because there is nothing within the NMF or PLCA optimization objective to encourage or discourage one group of parameters to only explain one particular sound. As a result, the basis vectors that are associated with one sound source will explain another and vice versa, resulting in little to no separation.

This issue is illustrated geometrically in Fig. 2.11, where we display three-dimensional normalized spectrogram data (i.e., spectra with low, middle, and high frequencies) of two sound sources and their mixture using a standard 2-simplex diagram. We additionally show three basis vectors for each source, the convex hulls for each source, and the convex hull of the combined model. The convex hull of a given source represents the general geometrical region within the simplex of that source. It is defined by the set $U = \{\sum_{z \in Z_s} \theta_z \, \mathbf{w}_z \mid \theta_z \geq 0, \sum_{z \in Z_s} \theta_z = 1\}$, where $\mathbf{w}_z$ is a column of $\mathbf{W}$ that sums to one and Z_s is the set of values of z for source s.

In the ideal situation, the basis vectors for each source ($\bigtriangledown$ and $\bigtriangleup$) and their corresponding convex hulls would surround their respective spectra data points (x and $+$). Instead, the convex hulls for each source surround the mixture spectra ($\star$). This means that the learned basis vectors for each source are inaccurate and do not actually represent the frequency content of the individual sources. This will result in poor separation quality, requiring alternative approaches or algorithmic extensions to unsupervised parameter estimation.

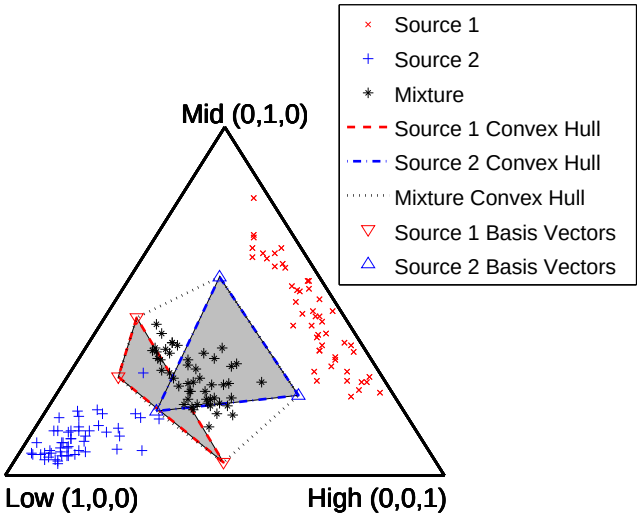


Figure 2.11: A 2-simplex of normalized three-frequency spectra of two sources using unsupervised parameter estimation. The convex hulls and basis vectors of each source are shown, along with the mixture convex hull. The bottom-left, top-middle, and bottom-right corners of the simplex represent low, middle, and high frequencies respectively. Notice, the convex hulls of each source overlap and do not outline their respective spectra.

2.6.2 Supervised Parameter Estimation

To overcome this problem, typically we use supervised or semi-supervised parameter estimation. The general idea of supervised separation is to use isolated training data (recordings) of each sound source within a mixture to pre-learn the individual basis vectors for each source using unsupervised estimation. Then, given an unknown mixture recording of similar sources, the pre-learned basis vectors are combined to form a single, larger model. Given the pre-computed basis vectors, a new set of corresponding gains or weights are estimated to fit the mixture [19].

The complete process is illustrated in Fig. 2.12 for NMF. As shown, we see that supervised parameter estimation requires $N_s + 1$ factorization, where N_s is the total number of sources within the mixture. One unsupervised factorization is needed to train each source model independently and one final supervised parameter estimation is needed to model the mixture.

For the case of the last factorization, the optimization objective for the final mixture is modified to be

$$\begin{aligned} &\arg\min_{\mathbf{H}} \;\; D(\mathbf{V} \,|\, \mathbf{W}\,\mathbf{H}) \\ &\text{subject to } \mathbf{H} \geq \mathbf{0}, \end{aligned} \tag{2.51}$$

where we only optimize over the variables $\mathbf{H}$. A similar modification to the PLCA parameter estimation algorithm can be used and results in a modified log-likelihood expression in which the pre-learned frequency distributions are held fixed.

This procedure is outlined in Algorithm 9, with the help of a slightly modified NFM/PLCA parameter estimation procedure outlined in Algorithm 10. The latter algorithm allows us to provide an optional input parameter $\mathbf{W}_1$ to our NMF/PLCA parameter estimation algorithm in order to specify a set of pre-learned basis vectors that are held fixed throughout the mixture-based parameter estimation process.

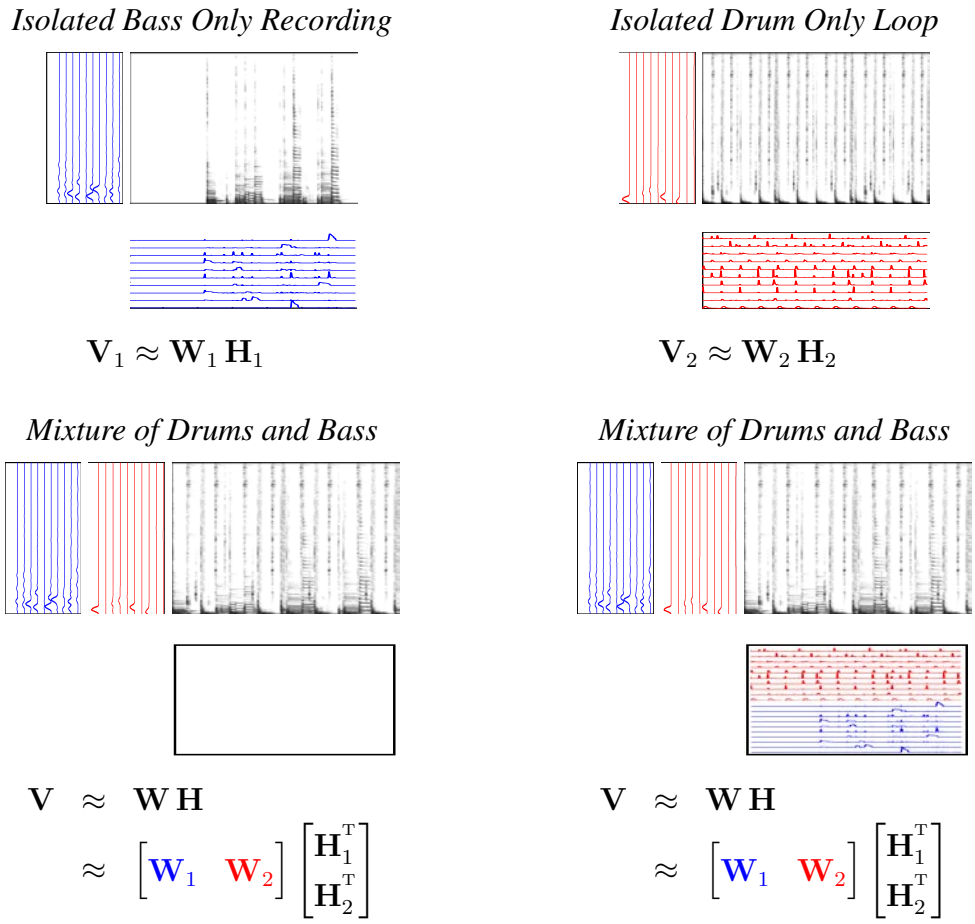


Figure 2.12: Supervised NMF performed on drum and bass guitar recordings. (Top) Independent NMF models are estimated: one using a recording of bass guitar and the other using a recording of drums. (Bottom Left) The basis vectors of the bass guitar and drum model are concatenated. (Bottom Right) The gains or weights of the new, larger NMF model are estimated, while the pre-computed basis vectors are held fixed.

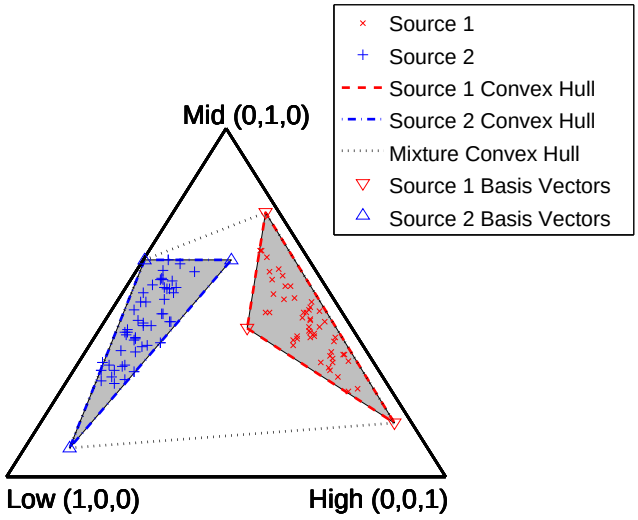


Figure 2.13: A 2-simplex of normalized three-frequency spectra of two sources using supervised parameter estimation. The convex hulls and basis vectors of each source are shown, along with the mixture convex hull. The bottom-left, top-middle, and bottom-right corners of the simplex represent low, middle, and high frequencies respectively. Notice, the convex hulls of each source do not overlap and outline their respective spectra.

We further illustrate this process using a simplex diagram as shown in Fig. 2.13. In this case, the learned basis vectors for each sound source ideally surround the spectra data points of their respective sources. It should be noted, however, that such ideal convex hulls still do not necessarily imply the learned models will achieve high-quality good separation. The groups of basis vectors are not orthogonal to each other and the combined basis vectors of the two sources can be used to perfectly model any mixture sound within the combined convex hull (black, dotted), even though the individual hulls only occupy small subregions. In addition, in most real-world scenarios, the hulls will overlap and make the problem even more difficult.

Finally, while we note that supervised parameter estimation significantly improves upon unsupervised estimation, it requires isolated training data for each individual sound source within a mixture. In a wide variety of settings, this is not possible, motivating alternative approaches. We discuss one approach below and will also discuss a new, alternative approach using our proposed user-guided methodology in Chapter 4.

Algorithm 9 Supervised NMF/PLCA Parameter Estimation

Procedure SUP-NMF-PLCA ($\mathbf{V}_s \, \forall s = 1, \ldots, N_s$, // training spectrograms
$\mathbf{V}_m$, // mixture spectrogram
N_z/N_s // number of basic vectors per source s
)
// Compute the prototypical frequency spectra for each source s
for $s = 1, \ldots, N_s$ **do**
$(\mathbf{W}_s, \mathbf{H}_s) \leftarrow$ NMF-PLCA($\mathbf{V}_s$, N_z/N_s)
end for
// Concatenate pre-learned basis vectors
$\mathbf{W} = [\mathbf{W}_1 \, \mathbf{W}_2 \ldots \mathbf{W}_{N_s}]$
// Compute mixture factorization
$(\mathbf{W}, \mathbf{H}) \leftarrow$ NMF-PLCA($\mathbf{V}_m$, N_z, $\mathbf{W}$)
return: $\mathbf{W}$ and $\mathbf{H}$

Algorithm 10 NMF-PLCA Parameter Estimation with Optional Predefined Basis Vectors

Procedure NMF-PLCA (

$\mathbf{V} \in \mathbf{R}_{+}^{N_F \times N_T}$, // input data matrix

N_K, // number of basic vectors

$\mathbf{W}_1 \in \mathbf{R}_{+}^{N_F \times N_{K_1}}$ where $N_{K_1} + N_{K_2} = N_K$ and $N_{K_1}, N_{K_2} \geq 0$ // optional, fixed basis

)

initialize: $\mathbf{W}_2 \in \mathbf{R}_{+}^{N_F \times N_{K_2}}$, $\mathbf{H}_1 \in \mathbf{R}_{+}^{N_{K_1} \times N_T}$, $\mathbf{H}_2 \in \mathbf{R}_{+}^{N_{K_2} \times N_T}$ // random values > 0

define: $\mathbf{W} := [\mathbf{W}_1 \, \mathbf{W}_2]$ and $\mathbf{H} := [\mathbf{H}_1 \, \mathbf{H}_2]$

repeat

optimize over $\mathbf{W}$

$$\mathbf{W}_2 \leftarrow \mathbf{W}_2 \odot \frac{(\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}})\mathbf{H}_2{}^{\mathrm{T}}}{\mathbf{1}\,\mathbf{H}_2{}^{\mathrm{T}}} \tag{2.52}$$

optimize over $\mathbf{H}$

$$\mathbf{H} \leftarrow \mathbf{H} \odot \frac{\mathbf{W}^{\mathrm{T}}(\frac{\mathbf{V}}{\mathbf{W}\,\mathbf{H}})}{\mathbf{W}^{\mathrm{T}}\,\mathbf{1}} \tag{2.53}$$

until convergence

return: $\mathbf{W}$ and $\mathbf{H}$

2.6.3 Semi-Supervised Parameter Estimation

The most common alternative approach to completely supervised parameter estimation is semi-supervised parameter estimation. The procedure is near identical to that of supervised separation with the exception that one or more source sources lack isolated training data and are not pre-learned, as shown in Fig. 2.14. This results in a modified optimization objective,

$$\begin{aligned} &\underset{\mathbf{H},\mathbf{W}_2}{\arg\min} \;\; D(\mathbf{V} \,|\, \mathbf{W}\,\mathbf{H}) \\ &\text{subject to } \mathbf{H} \geq \mathbf{0}, \mathbf{W}_2 \geq 0, \end{aligned} \tag{2.54}$$

where $\mathbf{W} := [\mathbf{W}_1 \, \mathbf{W}_2]$, $\mathbf{W}_1$ is a fixed, known matrix of basis vectors and $\mathbf{W}_2$ is an unknown matrix of basis vectors. We also say that we have N_{s_t} training recordings, which is less than the N_s possible training recordings. This process is illustrated in Fig. 2.14 and outlined in Algorithm 11.

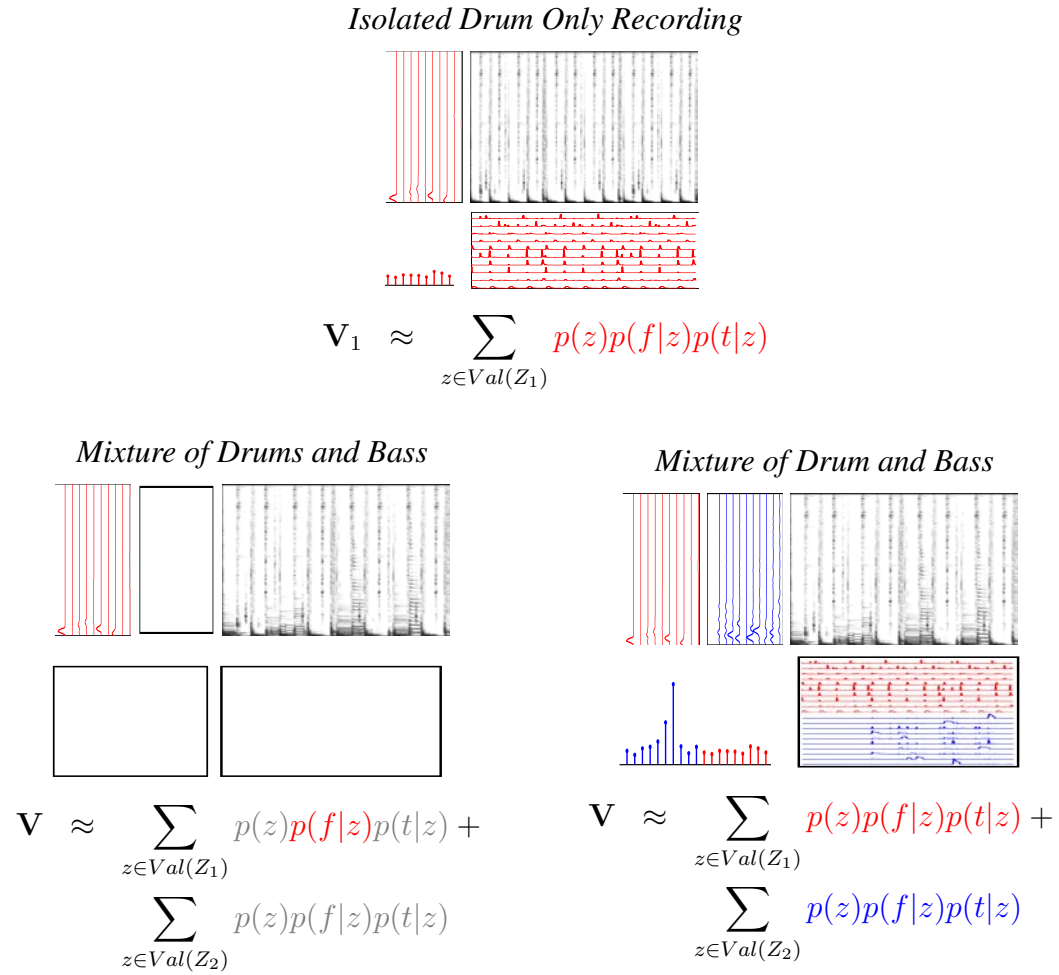


Figure 2.14: Semi-supervised PLCA performed on drum and bass guitar recordings. (Top) A single factorization of an isolated drum recording. (Bottom Left) The basis vectors of the bass guitar and drum model are concatenated. (Bottom Right) The gains or weights of the new, larger NMF model are estimated, while the pre-computed basis vectors are held fixed.

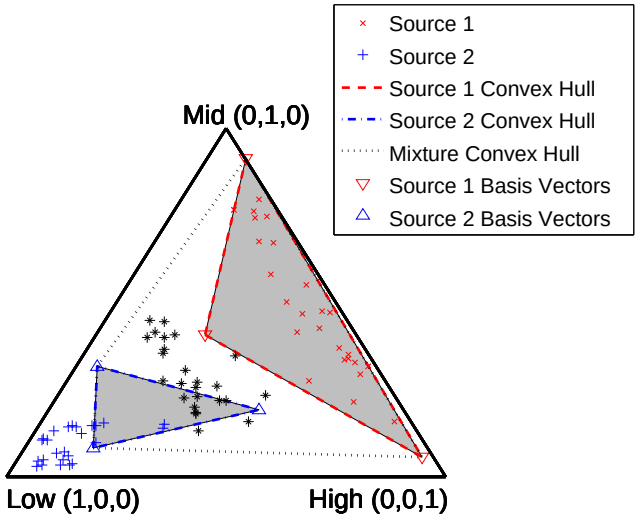


Figure 2.15: A 2-simplex of normalized three-frequency spectra of two sources using semi-supervised parameter estimation. The convex hulls and basis vectors of each source are shown, along with the mixture convex hull. The bottom-left, top-middle, and bottom-right corners of the simplex represent low, middle, and high frequencies respectively. Notice, the convex hulls of each source do not overlap, but the hull of the source learned from the mixture does not precisely outline its respective spectra.

In Fig. 2.15, we also illustrate the semi-supervised parameter estimation process using a simplex diagram. In this case, the basis vectors for the first sound source are learned from isolated training data and the basis vectors for the second source are learned from a mixture recording. Notice how the convex hull of the first source more accurately outlines its spectra data points and the convex hull of the second source less accurately outlines its spectra data points. This tells us that we have a better representation of the first source compared to the second and that the collective group of basis vectors will result in lower quality separation compared to when supervised parameter estimation is used.

Algorithm 11 Semi-Supervised NMF/PLCA Parameter Estimation

Procedure SEMI-SUP-NMF-PLCA ($\mathbf{V}_s \forall s = 1, \ldots, N_{s_t}$, // N_{s_t} training spectrograms
$\mathbf{V}_m$, // mixture spectrogram
N_z/N_s // number of basic vectors per source s
)
// Compute the prototypical frequency spectra for available source s
for $s = 1, \ldots, N_{s_t}$ **do**
$(\mathbf{W}_s, \mathbf{H}_s) \leftarrow$ NMF-PLCA($\mathbf{V}_s$, N_z/N_s)
end for
// Concatenate learned basis vectors
$\mathbf{W}_t = [\mathbf{W}_1 \, \mathbf{W}_2 \ldots \mathbf{W}_{N_{s_t}}]$
// Compute mixture factorization
$(\mathbf{W}, \mathbf{H}) \leftarrow$ NMF-PLCA($\mathbf{V}_m$, N_z, $\mathbf{W}_t$)
return: $\mathbf{W}$ and $\mathbf{H}$

2.7 Separating Mixtures

Once we compute a model of a mixture sound using unsupervised, supervised, or semi-supervised parameter estimation, we can finally perform separation. We denote two methods to do so as source synthesis and source filtering. In the case of source synthesis, the STFT magnitude of each source within a mixture is reconstructed directly from the estimated mixture model. In the case of source filtering, the STFT magnitude for each separate source is generated by filtering the original mixture as a function of the NMF/PLVM model akin to Weiner filtering [96, 97]. Both methods are discussed below.

2.7.1 Source Synthesis

For source synthesis, we reconstruct the time-frequency magnitude of a given source by multiplying together the portion of an estimated mixture model that corresponds to each source. When using NMF, this results in multiplying the basis vectors and activations of a

given source together

$$\hat{\mathbf{V}}_s = \mathbf{W}_s \mathbf{H}_s \tag{2.55}$$

where $\hat{\mathbf{V}}_s$ is the estimated time-frequency magnitude for source s, and $\mathbf{W}_s$ and $\mathbf{H}_s$ are the basis vectors and activations for source s. This process is analogous to the reverse of the supervised separation procedure and is depicted in Fig. 2.16. Alternatively, when using the probability notation, we write

$$\hat{p}(f, t|s) = \sum_{z \in Val(Z_s)} p(z)p(f|z)p(t|z), \tag{2.56}$$

where the summation is over all values of z that correspond to source s. Once the individual source magnitude spectrograms are constructed, the original mixture phase $\angle \mathbf{X}$ is used in conjunction with the inverse STFT to generate the final, separated time-domain signals $\hat{\mathbf{x}}_s$. An additional scaling factor may also be necessary to match the input mixture gain with the reconstructed outputs.

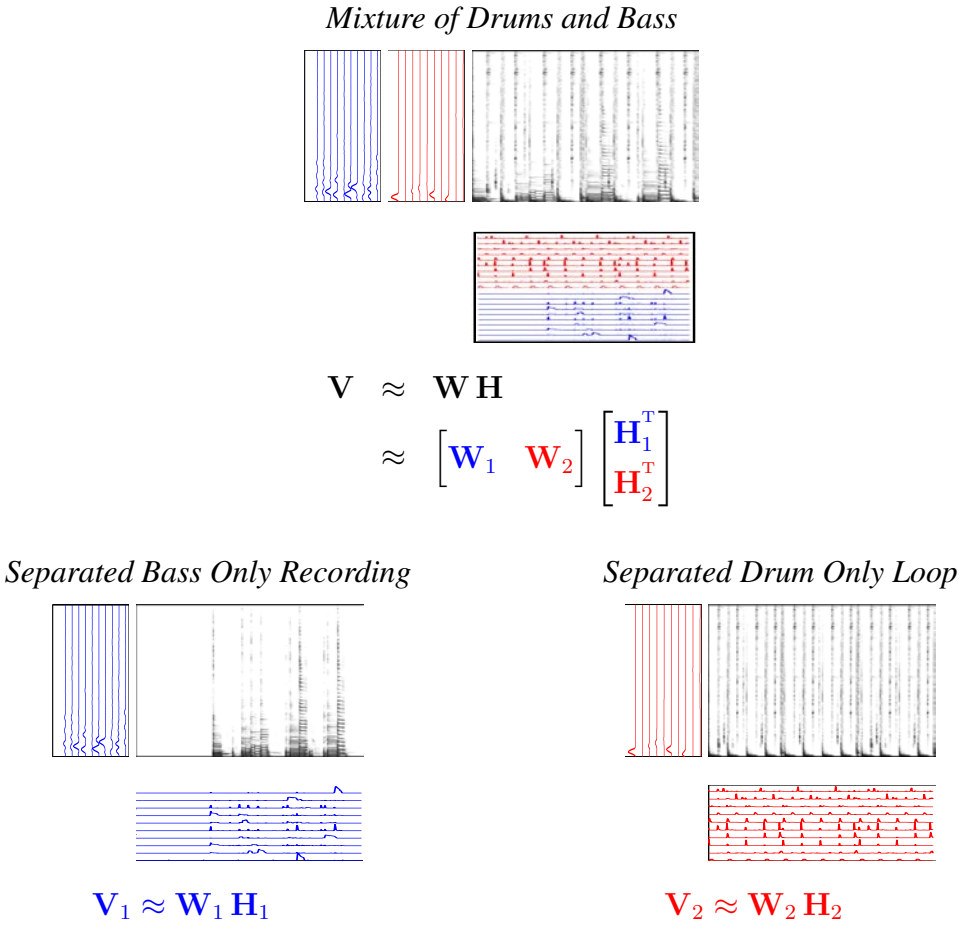


Figure 2.16: (Top) The mixture NMF (or PLCA), where groups of basis vectors and activations are associated with one sound source or another and estimated. (Bottom Left) The reconstructed bass source spectrogram. (Bottom Right) The reconstructed drum source spectrogram.

2.7.2 Source Filtering

In contrast to directly synthesizing the separated sources from the mixture model, we can filter the original mixture signal as a function of the NMF/PLVM output. This allows us to maintain the intricate detail of the mixture signal and simply filter out the unwanted sources from the mixture to generate the final separated sources. Typically, this approach results in significantly less audible artifacts in the separated sources, but also can result in

less separation.

To filter the mixture signal, we use the NMF/PLVM to generate the frequency response of a time-varying linear filter $\mathbf{F}_s$ for each source s. This filter is commonly called a masking filter, or soft mask filter, and is constructed by

$$\mathbf{F}_s = \frac{\mathbf{W}_s \mathbf{H}_s}{\mathbf{W} \mathbf{H}} \tag{2.57}$$

where $\mathbf{W}_s$ and $\mathbf{H}_s$ are the basis vectors and activations for source s. Each column of $\mathbf{F}_s$ represents the frequency response of a single filter and the column indices represent the time frames at which the filters are applied. In probability notation, the masking filter is generated via

$$\mathbf{F}_s = p(s|f,t) = \sum_{z \in Val(Z_s)} p(z|f,t) = \frac{\sum\limits_{z \in Val(Z_s)} p(z)p(f|z)p(t|z)}{\sum\limits_{z \in Val(Z)} p(z)p(f|z)p(t|z)}. \tag{2.58}$$

In essence, the posterior probability of a given source s is used as a ratio to filter the mixture at each time-frequency point. We then use the overlap-add procedure described in Section 2.3.4 to filter the original mixture time-frequency domain signal $\mathbf{X}$ via

$$\hat{\mathbf{V}}_s = \mathbf{F}_s \odot \mathbf{V} \tag{2.59}$$

and transform back to the time-domain using the ISTFT. For alternative approaches to generating masking filters, please see Fitzgerald and Rajesh [98] and others.

While this method performs well in practice, there are several problems with this approach (and source synthesis). Most notably, the generated filter frequency responses are not guaranteed to be time-limited, can have sharp discontinuities, and do not correspond to a realizable linear filter in the time-domain. As a result, when we transform the filter frequency responses into the time-domain, time-aliasing and other unwanted, audible artifacts can occur. In addition, because there is time-overlap between each applied filter (columns of $\mathbf{F}_s$), there are also constraints implied by the STFT that must be resolved.

To overcome these issues, any form of finite impulse response optimal filter design

(e.g., the window method) can be used to convert each column of $\mathbf{F}_s$ to a set of time-limited filters independently. Alternatively, STFT consistency constraints can be used to jointly estimate a set of filters $\mathbf{F}_s$ as described in the work of LeRoux et al. [99, 100]. For the purposes of this work, we use the baseline source filtering approach, in conjunction with overlap-add, to generate the final output sources.

2.8 Complete Separation Algorithm

Given that we have discussed everything required to perform NMF/PLVM-based separation, we now outline a complete separation algorithm. In this case, the algorithm definition is flexible in that it will perform unsupervised, supervised, or semi-supervised parameter estimation via an NMF or PLCA algorithm depending, on the input arguments. As a result, we do not need to individually reference the contrasting estimation procedures. Also note, for simplicity, the number of basis vectors or latent components N_z is the same for each sound source within a mixture. If desired, this can easily be generalized to vary according to sound source.

Algorithm 12 Complete NMF/PLCA Source Separation

Procedure NMF-PLCA-SEPARATION (

$\mathbf{x}_m \in \mathbf{R}^{N_\tau}$, // time-domain mixture signal

$\mathbf{x}_s \in \mathbf{R}^{N_{\tau_s}}\ \forall s \in \{1, ..., N_{s_t}\}$, // $0 \leq N_{s_t} \leq N_s$ time-domain training signals

N_z, // number of basic vectors

N_s, // number of sources

P // STFT parameters (hop size, DFT size, window size, window function)

)

// compute STFT and NMF-PLCA of training data

for all $s \in N_{s_t}$ **do**

$(\mathbf{X}_s, \angle \mathbf{X}_s) \leftarrow$ STFT($\mathbf{x}_s$, P)

$(\mathbf{W}_s, \mathbf{H}_s) \leftarrow$ NMF-PLCA($\mathbf{X}_s, N_z/N_s$)

end for

// compute STFT and NMF-PLCA of mixture signal data

$(\mathbf{X}_m, \angle \mathbf{X}_m) \leftarrow$ STFT($\mathbf{x}_m$, P)

$(\mathbf{W}, \mathbf{H}) \leftarrow$ NMF-PLCA($\mathbf{X}_m, N_z, [\mathbf{W}_1\, \mathbf{W}_2 \ldots \mathbf{W}_{N_{s_t}}]$)

// filter mixture according to NMF-PLCA output

for all $s \in N_s$ **do**

$\mathbf{F}_s \leftarrow \mathbf{W}_s\, \mathbf{H}_s \,/\, \mathbf{W}\, \mathbf{H}$ // compute filter

$\hat{\mathbf{X}}_s \leftarrow \mathbf{F}_s \odot \mathbf{X}$ // filter mixture

$\mathbf{x}_s \leftarrow$ ISTFT($\hat{\mathbf{X}}_s$, $\angle \mathbf{X}_m$, P)

end for

return: time-domain signals $\mathbf{x}_s,\ \forall s \in \{1, ..., N_s\}$

Chapter 3

An Interactive Approach

3.1 Introduction

Given the complete background of NMF/PLVM-based separation methods, we now begin discussion of our proposed interactive approach to source separation. We start from our discussion in Section 1.4 and Section 1.5, where we outlined related work of user-guided source separation and more generally interactive machine learning. As mentioned, there has been a recent interest in user-guided source separation methods both commercially and within the academic research community, spawning several intriguing and promising separation approaches.

Each approach, however, has its specific limitations and, collectively, all past approaches leave room for improvement for several general and significant reasons. Firstly, the separation quality of even current state-of-the-art methods, which typically focus on restricted problem domains (e.g., pitch-based separation, vocal extractions, etc.), do not achieve perfect separation, and commonly break down when applied to difficult situations. Secondly, in past user-guided methods there is little to no emphasis on user-feedback, refinement, and iteration, limiting how much an end-user can help inform the separation process (i.e., typically one-shot processes). And finally, it is unclear which previously discussed method of user-input or combination thereof performs best, if any.

Because of these observations, in this work we try to fully embrace such pitfalls and make one single, critical observation that guides our entire approach. That is, we take

the stance that any separation algorithm (user-guided or not) will never work perfectly–at least initially. Then, we seek to develop a mechanism to rapidly correct for mistakes in the separation estimate outputs and refine the results. If we can carefully design such a mechanism, even if it requires a significant effort on behalf of end-users, the approach will be useful for those who are willing to spend the time. In this way, we place a significant emphasis on user-feedback, refinement, and iteration.

As a result, we have chosen to denote our work as *interactive*, as described by the Oxford dictionary as "allowing a two-way flow of information between a computer and a computer-user," rather than simply a one-way user-guided interaction. While a subtle (and perhaps seemingly trivial) distinction, we believe two-way interactivity is immensely beneficial and deserving of the distinction. In addition, this difference aligns our proposed work with the research topic of interactive machine learning.

Given this high-level, philosophical approach, we now outline our primary design goals for our proposed separation algorithm in Section 3.2, the proposed interaction design in Section 3.3, and user-interaction analogies in Section 3.4. In Chapter 7, we further reflect on the benefits and problems of our proposed approach.

3.2 Goals

The primary design goals of our proposed separation approach are to:

- Provide a professional-level audio editing tool for recording engineers, musicians, and similar users to perform source separation for applications such as music remixing and content creation.
- Allow end-users to use drawing and painting-like tools to control and interact with the separation process in an intuitive, precise, and deliberate manner.
- Employ some form of NMF/PLVM separation technique to algorithmically perform separation.
- Emphasize the use of fast, iterative user-feedback to improve separation quality over time.

- Achieve high-quality separation results, even if a significant effort and time is required on behalf of the end-user.
- Evaluate the work with inexperienced and expert users alike and compare to alternative and past approaches.

Following these design goals, we first outline the interaction paradigm below.

3.3 Interaction Design

To separate a single-channel recording, we allow end-users to annotate or paint on time-frequency or spectrogram displays of sound. As opposed to exactly annotating each pixel of the image, however, a user is instructed to *roughly* paint on salient, time, frequency, or time-frequency regions that appear to correspond to one sound or another. We use color to denote sound source and opacity as a measure of confidence or strength, giving a degree of robustness to imprecise annotations.

Given the annotations, we perform an initial separation using an informed NMF/PLVM-based separation algorithm, which we discuss in Chapter 4. Then, we allow the user to listen to the separated outputs. If the results are unsatisfactory, the user can annotate errors in the output estimates or further annotate the input, and iteratively re-run the process in a quick, interactive manner until a desired result is achieved. Ideally, the time between user-annotation and updated separation results will be on the order of seconds or less (Fails and Olsen [48] mention a goal of five seconds or faster for image classification).

This interaction is depicted in a sequence of spectrogram displays in Fig. 1.1 and as a single user-interface in Fig. 3.1. For simplicity, we focus on the task of simultaneously separating one sound into two. To separate more than two sources (e.g., drums + vocals + guitar), the complete separation process can be repeated several times (e.g., first separate drums from vocals + guitar, then separate vocals from guitar). Note, however, there is nothing limiting the algorithm or interface from simultaneously separating more than two sources. Finally, we also depict the general, interactive machine learning feedback loop of our proposed system in Fig. 3.2.

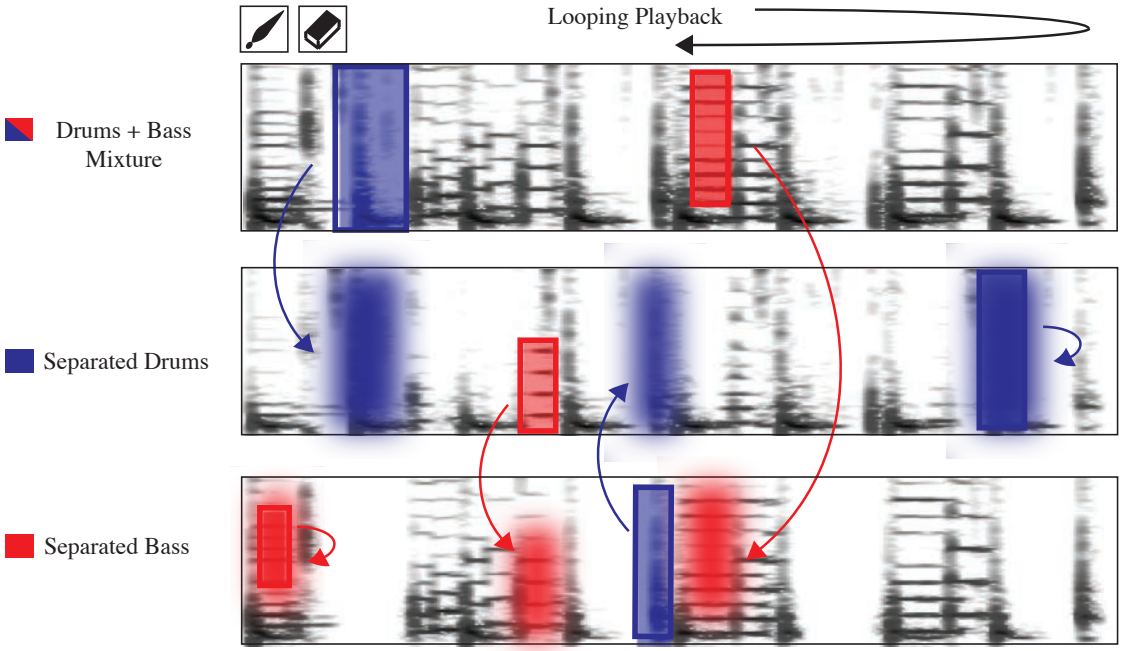


Figure 3.1: The proposed interactive source separation user-interface. (Top) The input mixture recording. (Middle) The first separated sound source (e.g., drums). (Bottom) The second separated sound source (e.g., bass guitar). (All) Painting on the input mixture or separated outputs will intelligently guide the separation process and dynamically update the output results. Annotation color is used to denote sound source and opacity is used as a measure of strength or confidence. Note, a single local annotation can have a global effect on the separation quality.

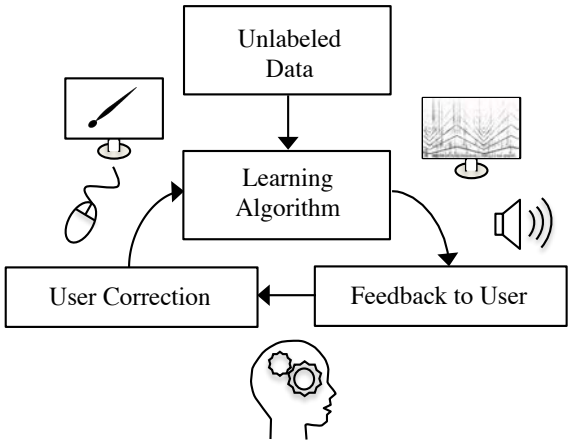
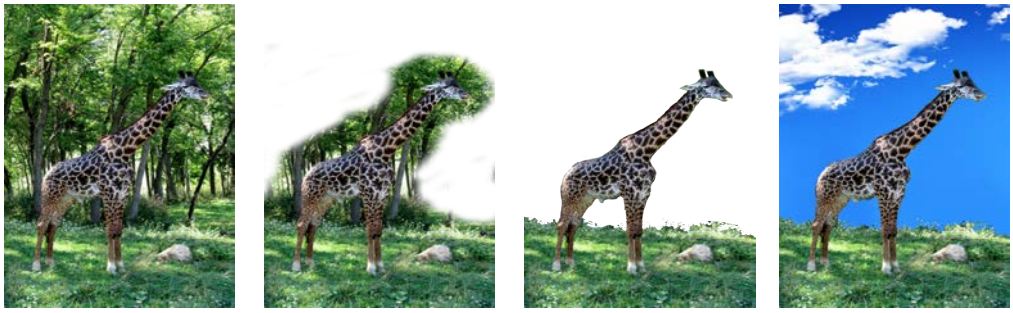


Figure 3.2: Interactive machine learning feedback-loop of our proposed method.

3.4 Analogies

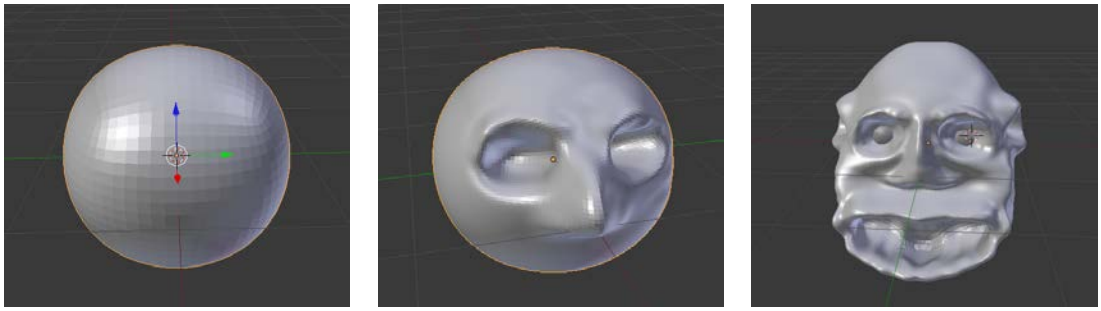
In designing this interaction, it was useful for us to think about two constructive analogies from related media content creation paradigms. In the first analogy, we can compare our proposed interaction to the process of extracting a layer (foreground, background, or other type of layer) in an image editing program. The process of extracting an image layer begins by a user defining what they wish to separate, taking a selection tool, clicking on the background a few times, and then clicking "extract."

If the layer is not completely separated, the process is repeated over and over in a rapid manner until a desirable result is achieved. This iterative process, combined with smart or intelligent selection tools, can be extremely powerful as demonstrated in the work of Rother et al. [101] and is something we strive towards. For our case of source separation, the "images" are time-frequency representations of sound, which have transparency (i.e., superposition property of sound) and consist of objects or sources that can have physically disjoint parts (i.e., harmonics of a voice are disconnected in time-frequency regions).



(a) Initial picture. (b) Initial separation. (c) Total separation. (d) Remixed image.

Figure 3.3: To extract the background layer of an image, a user will separate small portions of the background at a time in an iterative fashion to achieve the final, polished result. Only when the feedback loop between the user's action and the computer's response is sufficiently fast can a user effectively virtually sculpt (image license [102]).



(a) Initial 3D object. (b) Lightly sculpted character. (c) Sculpted 3D monster.

Figure 3.4: To sculpt a three-dimensional animated character, a user will begin with primitive shapes, and then gradually add and subtract material to sculpt the geometric mesh.

In the second analogy, we liken our proposed separation interaction to three-dimensional sculpting. In the process of three-dimensional sculpting, such as those of Maya or Blender, a user will typically take primitive geometric shapes, carefully add and subtract material (using manual or more intelligent tools), and over the course of many hours, create their finished product. This fast and iterative environment, which allows a user to push, pull, and generally manipulate content, is something we again strive towards with our proposed interaction and believe is critical in achieving high-quality separation results.

Chapter 4

Algorithm

4.1 Introduction

With the proposed interaction design specified, we can now discuss how we algorithmically incorporate the user-annotations to inform our separation algorithm. In short, the basic idea is to couple the user-annotations with a modified NMF/PLVM-based separation algorithm. Then, if the user-annotations change, the separation algorithm is dynamically re-run from scratch to consider the updated annotations, and the results are presented back to the user. In this way, we very simply provide a feedback mechanism without having to explicitly model the user, their actions, or anything similar such as would be required by reinforcement learning.

To couple the time-frequency user-annotations to an NMF/PLVM-based separation algorithm, we first render all the painting annotations from both the input and output displays of Fig. 3.1 into matrices (one for each sound source). We denote the annotation matrices as $\mathbf{\Lambda}_s \in \mathbf{R}^{N_f \times N_t},\ \forall s \in \{1, .., N_s\}$ as shown in Fig. 4.1. Alternatively, we denote all annotation matrices together in a single, large tensor $\mathbf{\Lambda} \in \mathbf{R}^{N_f \times N_t \times N_s}$ or $\mathbf{\Lambda} \in \mathbf{R}^{N_f \times N_t \times N_z}$, indexable by time, frequency, and source or latent component. Note $\Lambda_{ftz} = \Lambda_{fts}$ for all values of z that correspond to source s.

We then map these matrices to supplementary (regularization) parameters or constraints into the NMF/PLVM-based optimization objective. We do this to encourage or discourage one group of frequency components (source) to explain a given time-frequency point of the

mixture sound. Note, the mapping must be a function of time, frequency, and sound source. In this way, we are still able to leverage the computational power of the probabilistic model and incorporate expert user knowledge to guide the process.

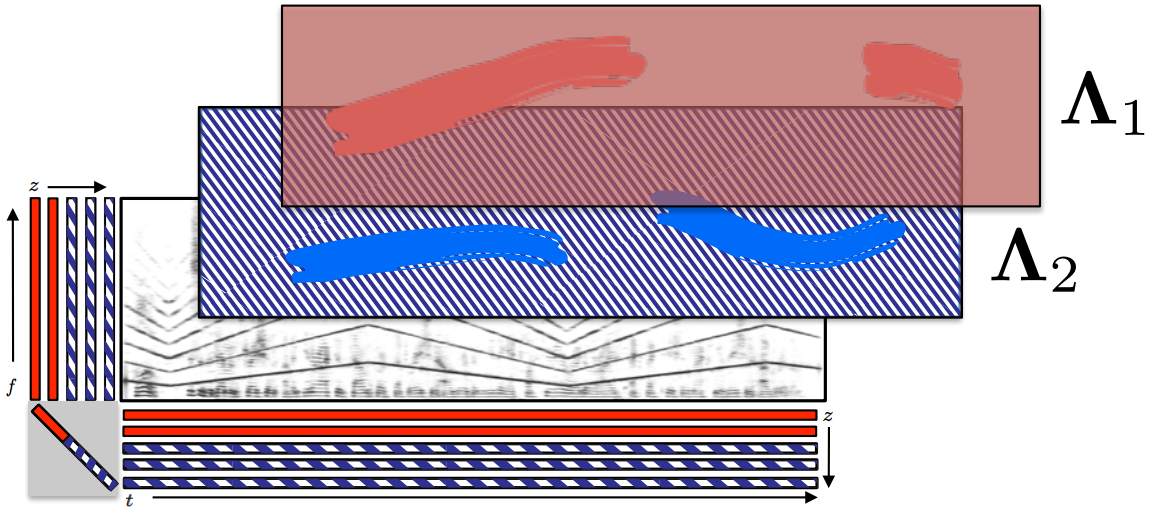


Figure 4.1: NMF/PLVM with time-frequency user-annotations. All annotations across both the input mixture and output results are rendered into two matrices, one for each sound source. The optimal gray box depicts parameters that are estimated for PLCA, which can optionally be absorbed into the timing activations as is the case for NMF.

By using this approach, we blend past manual time-frequency selection approaches with more recently developed automatic learning-based separation algorithms and receive several benefits. Firstly, this approach allows local annotations placed at one time-frequency point to propagate throughout the recording via the NMF/PLVM procedure and have a global effect on the separation estimation. Secondly, this minimizes the need for us to have a complete set of annotations, as required by many past approaches, and allows a user to get away with only annotating the salient time, frequency, or time-frequency regions of a sound.

Thirdly, this allows us to perform NMF/PLVM-based separation with or without any explicit isolated training data. This is because we can use the annotations in conjunction with supervised, semi-supervised, or even unsupervised parameter estimation. In each case,

the modification to the optimization objective is only applied to the final mixture and the training steps remain the same.

Fourthly, the time-frequency annotations we elicit can actually be more informative than isolated training data. This is because training data is only annotated according to time and for our algorithm, some or all data is annotated according to time and frequency. Because of this, it's not surprising that our approach can outperform standard supervised, semi-supervised, and unsupervised NMF/PLVM methods (as we demonstrate in Chapter 6).

4.2 A Probabilistic Latent Variable Model with Time-Frequency Constraints

Considering this approach, we now build off of our discussion of PLVMs in Section 2.5 and develop a new PLCA-based algorithm to incorporate the time-frequency user-annotations. For clarity, we restate the form of the symmetric two-dimensional PLCA model we use,

$$p(f,t) = \sum_z p(z)p(f|z)p(t|z). \tag{4.1}$$

Compared to a modified NMF formulation, incorporating optimization constraints as a function of time, frequency, and sound source into the factorized PLCA model is particularly interesting, motivating our focus.

Incorporating prior information into this model, and PLVMs in general, can be done in several ways. The most commonly used methods are by direct observations (i.e., setting probabilities to zero, one, etc.) or by incorporating Bayesian prior probabilities on model parameters. Direct observations do not give us enough control, so we consider incorporating Bayesian prior probabilities. For the case of Eq. (4.1), this would result in independently modifying the factor terms $p(f|z), p(t|z)$, or $p(z)$. Common prior probability distributions used for this purpose include Dirichlet priors [29, 28], gamma priors [36], and others.

Given that we would like to incorporate the user-annotations as a function of time,

frequency, and sound source, however, we notice that this is not easily accomplished using standard priors. This is because our model is *factorized* and each factor is only a function of one variable and (possibly) conditioned by another, making it difficult to construct a set of prior probabilities that, when jointly applied to $p(f|z), p(t|z)$, and/or $p(z)$, would encourage or discourage one source or another to explain a given time-frequency point.

We can see this more clearly when we consider PLCA to be the following simplified estimation problem:

$$X(f,t) \approx \phi(z)\phi(f,z)\phi(t,z), \tag{4.2}$$

where $X(f,t)$ is observed data that we model as the product of three distinct functions or factors $\phi(z)$, $\phi(f,z)$, and $\phi(t,z)$. Note, each factor has different input arguments and each factor has different parameters that we wish to estimate via EM. Also, forget for the moment that the factors must be normalized probabilities.

Given this model, if we wish to incorporate additional information, we could independently modify:

1. $\phi(z)$ to incorporate past knowledge of the variable z
2. $\phi(f,z)$ to incorporate past knowledge of the variable f and z
3. $\phi(t,z)$ to incorporate past knowledge of the variable t and z

This way of manipulation allows us to maintain our factorized form and can be thought of as prior-based regularization. If we would like to incorporate additional information/regularization that is a function of all three variables z, f, t, however, we must do something else. The first option would be to try to simultaneously modify all factors together to impose regularization that is a function of all three variables. This is unfortunately very difficult–both conceptually difficult to construct and practically difficult to algorithmically solve.

This motivates the use of posterior regularization (PR). PR provides us with an algorithmic mechanism via EM to incorporate constraints that are complementary to prior-based regularization. Instead of modifying the individual factors of our model as we saw before, we directly modify the posterior distribution of our model.The posterior distribution of our model, very loosely speaking, is a function of all random variables of our model. It is

natively computed within each E step of EM and is required to iteratively improve the estimates of our model parameters. In this example, the posterior distribution would be akin to $\phi(z, f, t)$, which is a function of t, f, and z, as required. We now formally discuss PR below, beginning with a general discussion and concluding with the specific form of PR we employ within our approach.

4.2.1 Posterior Regularization

The framework of posterior regularization, first introduced by Graça, Ganchev, and Taskar [34, 103, 103, 35], is a relatively new mechanism for injecting rich, typically data-dependent constraints into latent variable models using the EM algorithm. In contrast to standard Bayesian prior-based regularization, which applies constraints to the model parameters of a latent variable model in the maximization step of EM, posterior regularization applies constraints to the posterior distribution (distribution over the latent variables, conditioned on everything else) computation in the expectation step of EM. The method has found success in many natural language processing tasks, such as statistical word alignment, part-of-speech tagging, and similar tasks that involve latent variable models.

We see the basic idea of posterior regularization more clearly when we analyze the generalized EM algorithm presented in Section 2.5.2. In this case, what we do is constrain the distribution q in some way when we maximize the auxiliary bound $\mathcal{F}(q, \boldsymbol{\Theta})$ with respect to q in the expectation step of an EM algorithm, resulting in a

$$q^{n+1} \quad = \quad \arg\min_{q} \ \mathrm{KL}(q||p) + \Omega(q), \tag{4.3}$$

where $\Omega(q)$ constrains the possible space of q.

Note, the only difference between Eq. (4.3) and our past discussion on EM is the added term $\Omega(q)$. If $\Omega(q)$ is set to zero, we get back the original formulation and easily solve the optimization by setting $q = p$ without any computation (except computing the posterior p). Also note, to denote the use of constraints in this context, the term "weakly-supervised" was introduced by Graça [103] and is similarly adopted here.

This method of regularization is in contrast to prior-based regularization, where the

modified maximization step would be

$$\boldsymbol{\Theta}^{n+1} = \arg\max_{\boldsymbol{\Theta}} \mathcal{F}(q^{n+1}, \boldsymbol{\Theta}) + \Omega(\boldsymbol{\Theta}), \tag{4.4}$$

where $\Omega(\boldsymbol{\Theta})$ constrains the model parameters $\boldsymbol{\Theta}$ as opposed to Eq. (2.31). The only difference between Eq. (4.4) and our past discussion regarding EM is the added term $\Omega(\boldsymbol{\Theta})$. If $\Omega(\boldsymbol{\Theta})$ is set to zero, we get back the original formulation.

4.2.2 Linear Grouping Expectation Constraints

Given the general framework of posterior regularization, we need to define a meaningful penalty $\Omega(q)$ for which we map our annotations. We do this by mapping the annotation matrices to linear grouping constraints on the latent variable z. To do so, we first notice that Eq. (4.3) decouples for each time-frequency point for our specific model. Because of this, we can independently solve Eq. (4.3) for each time-frequency point, making the optimization much simpler. When we rewrite our E step optimization this using vector notation, we get

$$\begin{aligned} \arg\min_{\mathbf{q}} \quad & -\mathbf{q}_{ft}^{\mathrm{T}} \ln \mathbf{p}_{ft} + \mathbf{q}_{ft}^{\mathrm{T}} \ln \mathbf{q}_{ft} \\ \text{subject to} \quad & \mathbf{q}_{ft}^{\mathrm{T}} \mathbf{1} = 1,\ \mathbf{q}_{ft} \geq 0, \end{aligned} \tag{4.5}$$

where q and $p(z|f,t)$ for a given value of f and t is written as $\mathbf{q}_{ft}$ and $\mathbf{p}_{ft}$. Without any modification, we note q is optimal when equal to $p(z|f,t)$ as before.

We then apply our linear grouping constraints independently for each time-frequency point,

$$\begin{aligned} \arg\min_{\mathbf{q}} \quad & -\mathbf{q}_{ft}^{\mathrm{T}} \ln \mathbf{p}_{ft} + \mathbf{q}_{ft}^{\mathrm{T}} \ln \mathbf{q}_{ft} + \mathbf{q}_{ft}^{\mathrm{T}} \boldsymbol{\lambda}_{ft} \\ \text{subject to} \quad & \mathbf{q}_{ft}^{\mathrm{T}} \mathbf{1} = 1,\ \mathbf{q}_{ft} \geq 0, \end{aligned} \tag{4.6}$$

where we define $\boldsymbol{\lambda}_{ft} = [\Lambda_{ft1} \ldots \Lambda_{ft1} \Lambda_{ft2} \ldots \Lambda_{ft2}]^{\mathrm{T}} \in \mathbf{R}^{N_z}$ as the vector of user-defined penalty weights, $^{\mathrm{T}}$ is a matrix transpose, $\geq$ is element-wise greater than or equal to, and $\mathbf{1}$ is a column vector of ones. In this case, positive-valued penalties are used to decrease the probability of a given source, while negative-valued coefficients are used to increase the probability of a given source. Note, the penalty weights imposed on the group of values of

z that correspond to a given source s are identical, linear with respect to the z variables, and applied in the E step of EM, hence the name "linear grouping expectation constraints."

To solve the above optimization problem for a given time-frequency point, we form the Lagrangian

$$\mathcal{L}(\mathbf{q}_{ft}, \gamma) = -\mathbf{q}_{ft}^{\mathrm{T}} \ln \mathbf{p}_{ft} + \mathbf{q}_{ft}^{\mathrm{T}} \ln \mathbf{q}_{ft} + \mathbf{q}_{ft}^{\mathrm{T}} \boldsymbol{\lambda}_{ft} + \gamma(1 - \mathbf{q}_{ft}^{\mathrm{T}} \mathbf{1})$$

with γ being a Lagrange multiplier, take the gradient with respect to $\mathbf{q}$ and γ

$$\nabla_{\mathbf{q}_{ft}} \mathcal{L}(\mathbf{q}_{ft}, \gamma) = -\ln \mathbf{p}_{ft} + \mathbf{1} + \ln \mathbf{q}_{ft} + \boldsymbol{\lambda}_{ft} - \gamma \mathbf{1} = 0 \tag{4.7}$$

$$\nabla_{\alpha} \mathcal{L}(\mathbf{q}_{ft}, \gamma) = (1 - \mathbf{q}_{ft}^{\mathrm{T}} \mathbf{1}) = 0 \tag{4.8}$$

set equations Eq. (4.7) and Eq. (4.8) equal to zero, and solve for $\mathbf{q}_{ft}$, resulting in

$$\mathbf{q}_{ft} = \frac{\mathbf{p}_{ft} \odot \exp\{-\boldsymbol{\lambda}_{ft}\}}{\mathbf{p}_{ft}^{\mathrm{T}} \exp\{-\boldsymbol{\lambda}_{ft}\}} \tag{4.9}$$

where $\exp\{\}$ is an element-wise exponential function.

Notice the result is computed in closed-form and does not require any iterative optimization scheme as may be required in the general posterior regularization framework [34], minimizing the computational cost when incorporating the constraints. Also note, however, that this optimization must be solved for each time-frequency point of our spectrogram data for each E step iteration of our final EM parameter estimation algorithm.

4.2.3 Parameter Estimation

Now knowing the posterior-regularized expectation step optimization, we can derive a complete EM algorithm for a posterior-regularized two-dimensional PLCA model (PR-PLCA). The modification becomes only a small change to the original PLCA algorithm, which replaces equation Eq. (2.42) with

$$q(z|f,t) \leftarrow \frac{p(z)p(f|z)p(t|z)\tilde{\boldsymbol{\Lambda}}_{ftz}}{\sum_{z'} p(z')p(f|z')p(t|z')\tilde{\boldsymbol{\Lambda}}_{ftz'}} \tag{4.10}$$

where $\tilde{\mathbf{\Lambda}} = \exp\{-\mathbf{\Lambda}\}$. The entire algorithm is outlined in Algorithm 13. Notice, we continue to maintain closed-form E and M steps, allowing us to optimize further and draw connections to multiplicative non-negative matrix factorization algorithms.

Algorithm 13 PR-PLCA with Linear Grouping Expectation Constraints

Procedure PR-PLCA (

$\mathbf{V} \in \mathbf{R}_{+}^{N_f \times N_t}$, // observed normalized data

N_z, // number of basic vectors

N_s // number of sources

$\mathbf{\Lambda} \in \mathbf{R}^{N_f \times N_t \times N_z}$ // penalties

)

initialize: feasible $p(z)$, $p(f|z)$, and $p(t|z)$

precompute: $\tilde{\mathbf{\Lambda}} \leftarrow \exp\{-\mathbf{\Lambda}\}$

repeat

expectation step

for all z, f, t **do**

$$q(z|f,t) \leftarrow \frac{p(z)p(f|z)p(t|z)\tilde{\mathbf{\Lambda}}_{ftz}}{\sum_{z'} p(z')p(f|z')p(t|z')\tilde{\mathbf{\Lambda}}_{ftz'}} \tag{4.11}$$

end for

maximization step

for all z, f, t **do**

$$p(f|z) \leftarrow \frac{\sum_t V_{ft} q(z|f,t)}{\sum_{f'} \sum_{t'} V_{f't'} q(z|f',t')} \tag{4.12}$$

$$p(t|z) \leftarrow \frac{\sum_f V_{ft} q(z|f,t)}{\sum_{f'} \sum_{t'} V_{f't'} q(z|f',t')} \tag{4.13}$$

$$p(z) \leftarrow \frac{\sum_f \sum_t V_{ft} q(z|f,t)}{\sum_{z'} \sum_{f'} \sum_{t'} V_{f't'} q(z'|f',t')} \tag{4.14}$$

end for

until convergence

return: $p(f|z)$, $p(t|z)$, $p(z)$, and $q(z|f,t)$

4.2.4 Multiplicative Update Equations

To compare the proposed method to the multiplicative form of the PLCA algorithm outlined in Algorithm 7, we can rearrange the expressions in Algorithm 13 and convert to a multiplicative form following similar methodology to Smaragdis and Raj [90]. Rearranging the expectation and maximization steps, in conjunction with Bayes' rule, and $Z(f,t) = \sum_z p(z)p(f|z)p(t|z)\tilde{\mathbf{\Lambda}}_{ftz}$, we get

$$q(z|f,t) = \frac{p(f|z)p(t,z)\tilde{\mathbf{\Lambda}}_{ftz}}{Z(f,t)} \tag{4.15}$$

$$p(t,z) = \sum_f V_{ft} q(z|f,t) \tag{4.16}$$

$$p(f|z) = \frac{\sum_t V_{ft} q(z|f,t)}{\sum_t p(t,z)} \tag{4.17}$$

$$p(z) = \sum_t p(t,z) \tag{4.18}$$

Rearranging further, we get

$$p(f|z) = \frac{p(f|z)\sum_t \frac{V_{ft}\tilde{\mathbf{\Lambda}}_{ftz}}{Z(f,t)} p(t,z)}{\sum_t p(t,z)} \tag{4.19}$$

$$p(t,z) = p(t,z)\sum_f p(f|z)\frac{V_{ft}\tilde{\mathbf{\Lambda}}_{ftz}}{Z(f,t)} \tag{4.20}$$

which fully specifies the iterative updates. By putting Eq. (4.19) and Eq. (4.20) in matrix notation, we specify the multiplicative form of the proposed method in Algorithm 14.

Algorithm 14 PR-PLCA with Linear Grouping Expectation Constraints in Matrix Notation

Procedure PR-PLCA (

$\mathbf{V} \in \mathbf{R}_{+}^{N_f \times N_t}$, // observed normalized data

N_z, // number of basis vectors

N_s // number of sources

$\mathbf{\Lambda}_s \in \mathbf{R}^{N_f \times N_t},\ \forall\ s \in \{1, .., N_s\}$ // penalties

)

initialize: feasible $\mathbf{W} \in \mathbf{R}_{+}^{N_f \times N_z}$ and $\mathbf{H} \in \mathbf{R}_{+}^{N_z \times N_t}$

precompute:

for all s **do**

$$\tilde{\mathbf{\Lambda}}_s \leftarrow \exp\{-\mathbf{\Lambda}_s\} \tag{4.21}$$

$$\mathbf{X}_s \leftarrow \mathbf{V} \odot \tilde{\mathbf{\Lambda}}_s \tag{4.22}$$

end for

repeat

$$\mathbf{\Gamma} \leftarrow \sum_s (\mathbf{W}_s\, \mathbf{H}_s) \odot \tilde{\mathbf{\Lambda}}_s \tag{4.23}$$

for all s **do**

$$\mathbf{Z}_s \leftarrow \frac{\mathbf{X}_s}{\mathbf{\Gamma}} \tag{4.24}$$

$$\mathbf{W}_{(s)} \leftarrow \mathbf{W}_s \odot \frac{\mathbf{Z}_s\, \mathbf{H}_s^{\mathrm{T}}}{\mathbf{1}\, \mathbf{H}_s^{\mathrm{T}}} \tag{4.25}$$

$$\mathbf{H}_{(s)} \leftarrow \mathbf{H}_s \odot (\mathbf{W}_s^{\mathrm{T}}\, \mathbf{Z}_s) \tag{4.26}$$

end for

until convergence

return: $\mathbf{W}$ and $\mathbf{H}$

4.2.5 Computational Cost

Neglecting the pre-computation step in Algorithm 14, we consider the increase in computational cost at each EM iteration of the proposed method over the standard PLCA update equations in Algorithm 7. We notice that only Eq. (4.23) and Eq. (4.24) add computation compared to their counterpart of Eq. (2.48) in Algorithm 7 as a result of careful indexing of Eq. (4.25) and Eq. (4.26). Additionally, Eq. (2.48) of Algorithm 7 consists of an $O(N_f N_t N_z)$ matrix multiplication and an $O(N_f N_t)$ element-wise matrix division.

In contrast, Eq. (4.23) and Eq. (4.24) of Algorithm 14 consist of an $O(N_f N_t N_z)$ matrix multiplication, and an $O(N_s N_f N_t)$ element-wise matrix multiplication, division, and addition. In total, the difference is only an $O(N_f N_t N_s)$ element-wise matrix multiplication, division, and addition per EM iteration. As a result, the entire added cost per EM iteration for small N_s (typically two) is low and found to be acceptable in practice.

4.3 Modeling and Separating Mixtures

Now that we have discussed our proposed NMF/PLVM method given an arbitrary spectrogram $\mathbf{V}$, we must discuss how the method is used to model mixture sounds and perform separation. To do so, we follow the discussion of Section 2.6 and Section 2.7 directly to perform unsupervised, supervised, and semi-supervised separation, but in this case with user-interaction. Each of these three cases is discussed below.

4.3.1 Unsupervised

To perform separation with user-interaction in conjunction with what is typically thought of as unsupervised parameter estimation, we follow the illustration of Fig. 4.1 and Algorithm 14. That is, we partition the model parameters of our PLCA model into distinct groups, associate the groups with each sound source present in the mixture, and then simultaneously estimate all parameters (i.e., $\mathbf{W}$ and $\mathbf{H}$) of our model and leverage the user-annotations to steer the estimation process so that each group of parameters independently explains each sound source.

We depict the process of user-interaction in Fig. 4.2 using simplex diagrams. On the left, we show the use of standard unsupervised NFM/PLVM estimation using three basis vectors per sound source identical to Fig. 2.11 in Section 2.6.1. Note, ideally the basis vectors for each source ($\bigtriangledown$ and $\bigtriangleup$) and their corresponding convex hulls should surround their respective spectra data points (x and $+$). On the right, we show the use of our proposed interactive approach, where we interactively tune the regularization parameters to learn better basis vectors and timing activations (not depicted) from a mixture. While not perfect, the results on the right are a drastic improvement.

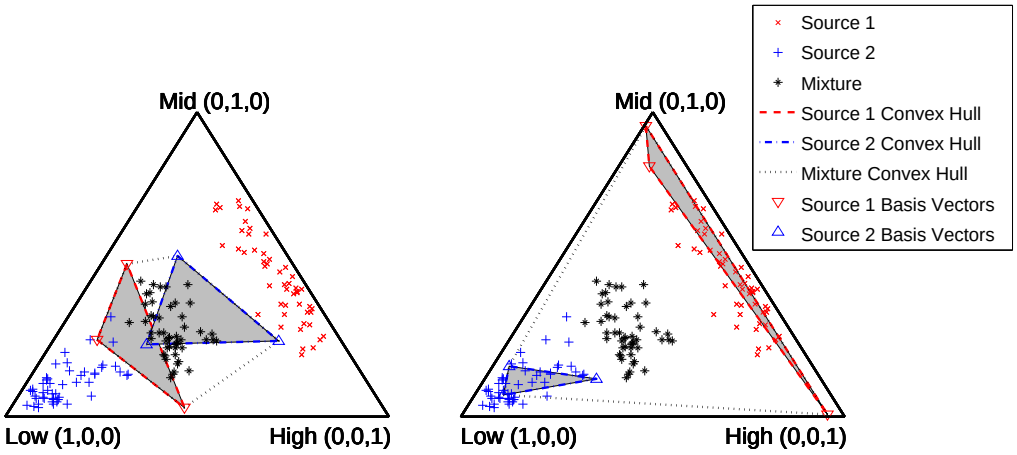


Figure 4.2: A 2-simplex diagram illustrating unsupervised separation with and without interaction. (Left) The basis vectors of the first source and the second source are learned in a unsupervised manner. (Right) With interaction, the proposed method results in better estimates of the basis vectors for both sound sources compared to standard PLCA.

4.3.2 Supervised Separation

To perform separation with user-interaction in conjunction with supervised parameter estimation, we can follow the procedure described in Section 2.6.2 with slight modification. That is, we can independently pre-learn groups of basis vectors for each sound source using isolated training data, concatenate all basis vectors together, and then perform parameter estimation on our mixture recording using Algorithm 14, albeit holding the pre-learned basis vectors fixed and considering the user-annotations. This is illustrated in Fig. 4.3.

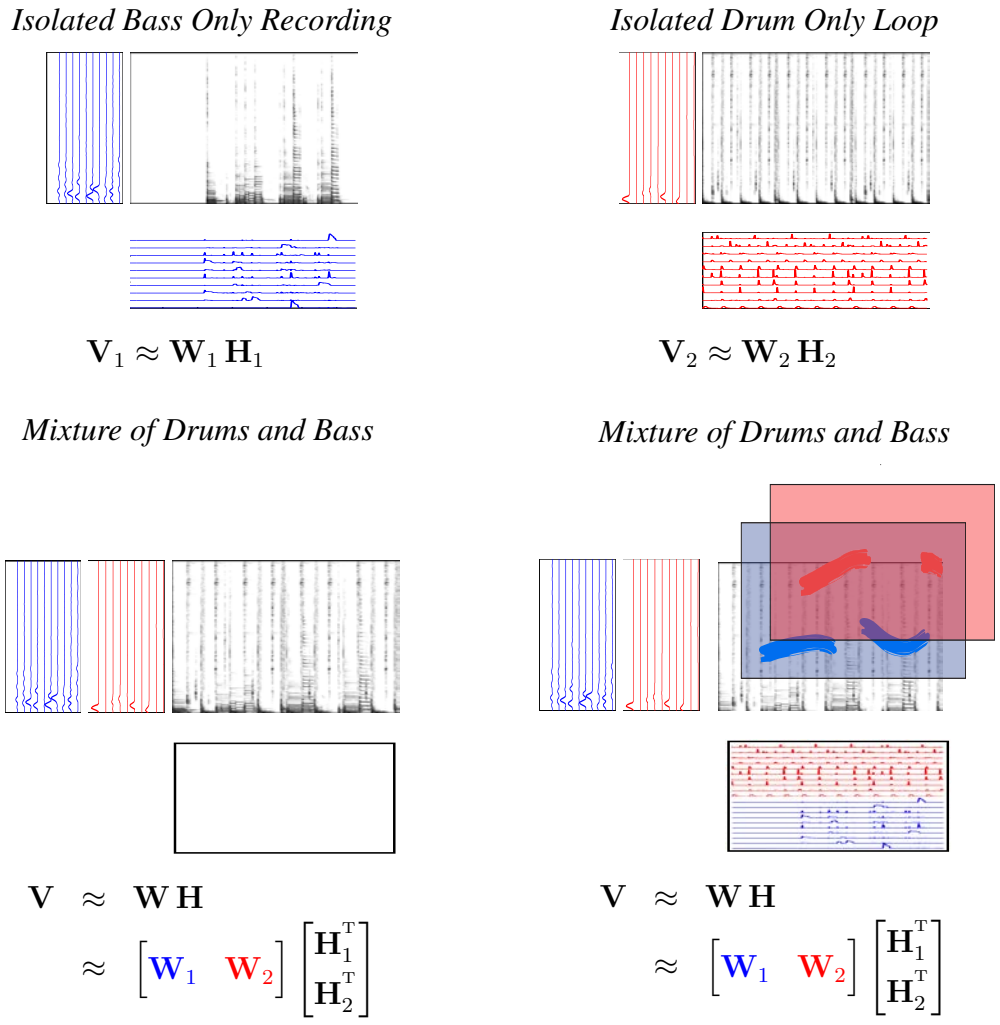


Figure 4.3: Supervised NMF/PLVM with user-interaction performed on drum and bass guitar recordings. (Top) Independent models are estimated: one using a recording of bass guitar and the other using a recording of drums. (Bottom Left) The basis vectors of the bass guitar and drum model are concatenated. (Bottom Right) The gains or weights of the new, larger model are estimated using the painting annotations, while the pre-computed basis vectors are held fixed.

We depict the process of supervised separation with user-interaction in Fig. 4.4 using simplex diagrams. In this case, we do not illustrate the adaptation of basis vectors, but how an individual mixture point is separated using pre-computed basis vectors. On the left,

we show the use of standard supervised separation of two sound sources, where we pre-learn the basis vectors for each source, and then attempt to separate a single mixture spectra point. The original, ground-truth separated source points are shown alongside the estimated separated sources. On the right, we interactively tune the regularization parameters so that the original and estimated source points align more precisely. While the original separation estimates are reasonably close to the ground-truth points, we can achieve even better results using the proposed method.

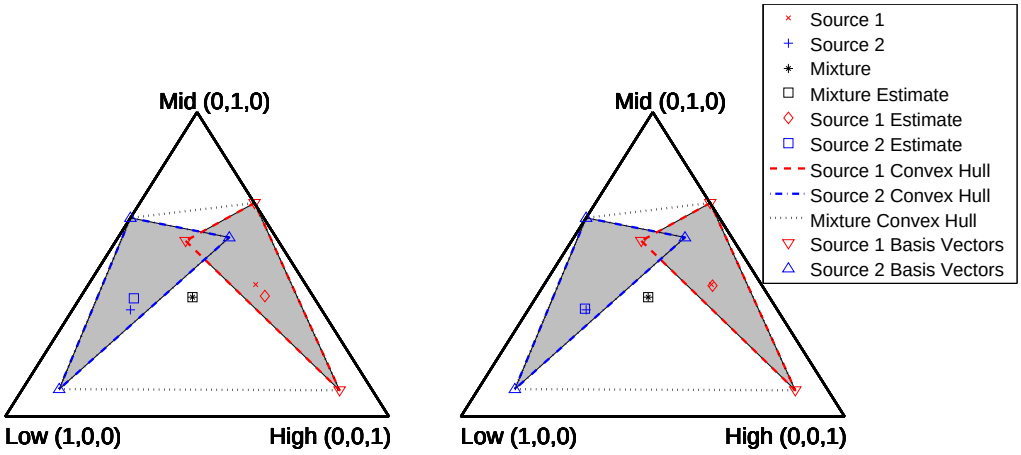


Figure 4.4: A 2-simplex diagram illustrating the reconstructions of the mixture and individual sources when using supervised separation with and without user-interaction. In both cases, the mixture is well reconstructed. However, the reconstruction error of the individual sources is noticeably higher using standard PLCA (left) compared to the proposed method (right).

4.3.3 Semi-Supervised

Finally, to perform separation with user-interaction in conjunction with standard semi-supervised separation, we follow our discussion from Section 2.6.3. In this case, we pre-learn one or more groups of basis vectors using isolated training data and then, given a mixture recording, learn the remaining basis vectors and all timing activations with the user-annotations using Algorithm 14, as illustrated in Fig. 4.5.

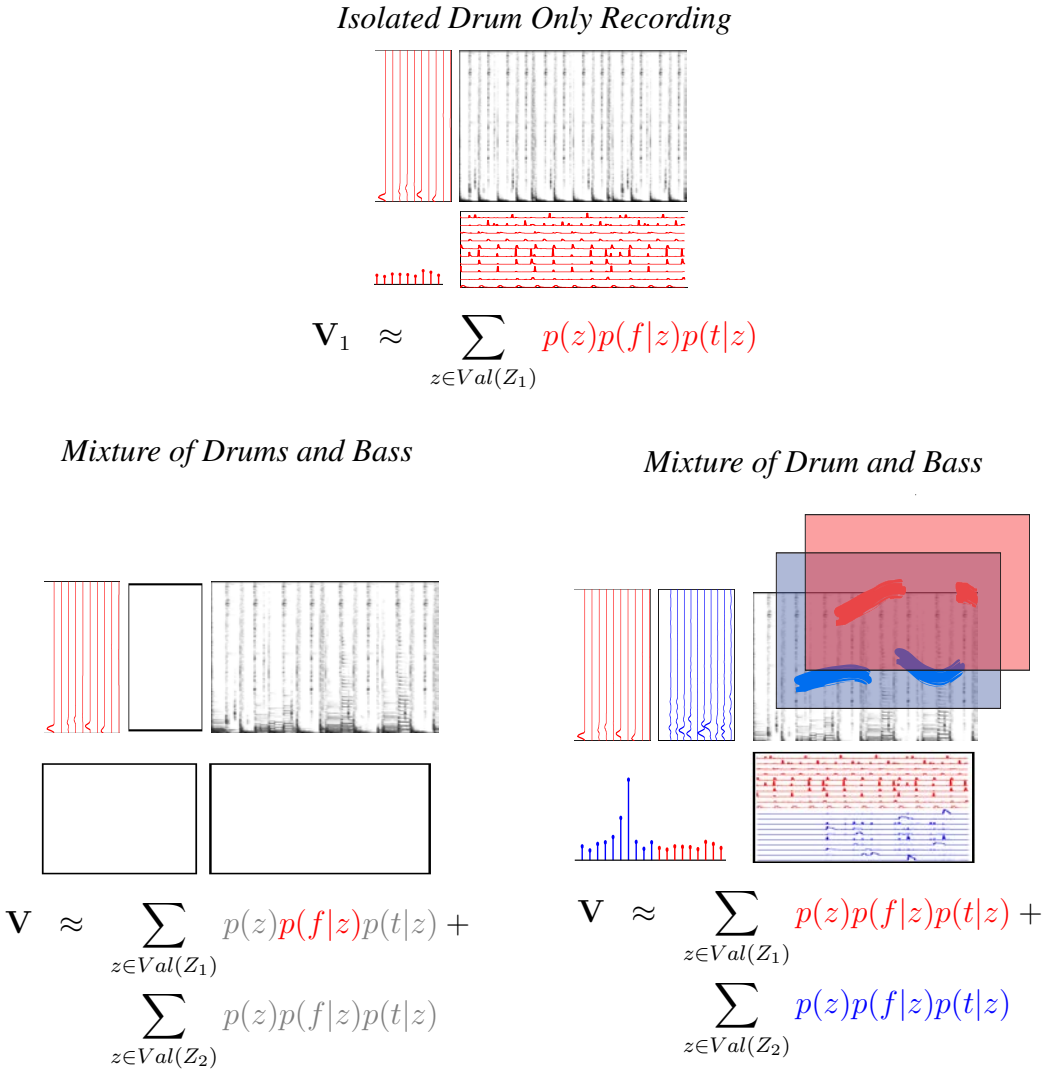


Figure 4.5: Semi-supervised NMF/PLVM with user-interaction performed on drum and bass guitar recordings. (Top) A single factorization of an isolated drum recording. (Bottom Left) The basis vectors of the bass guitar and drum model are concatenated. (Bottom Right) The gains or weights of the new, larger NMF model are estimated with user-interaction, while the pre-computed basis vectors are held fixed.

We also depict the process of semi-supervised separation with user-interaction in Fig. 4.6 using simplex diagrams. In this case, we illustrate the adaptation via only the basis vectors that are inferred from a mixture. On the left, we show the use of standard semi-supervised separation of two sound sources, where we pre-learn the basis vectors for one source and

infer the basis vectors for the other source from a mixture recording. On the right, we interactively tune the regularization parameters so that the inferred basis vectors for the second source more accurately bound the second source spectra points.

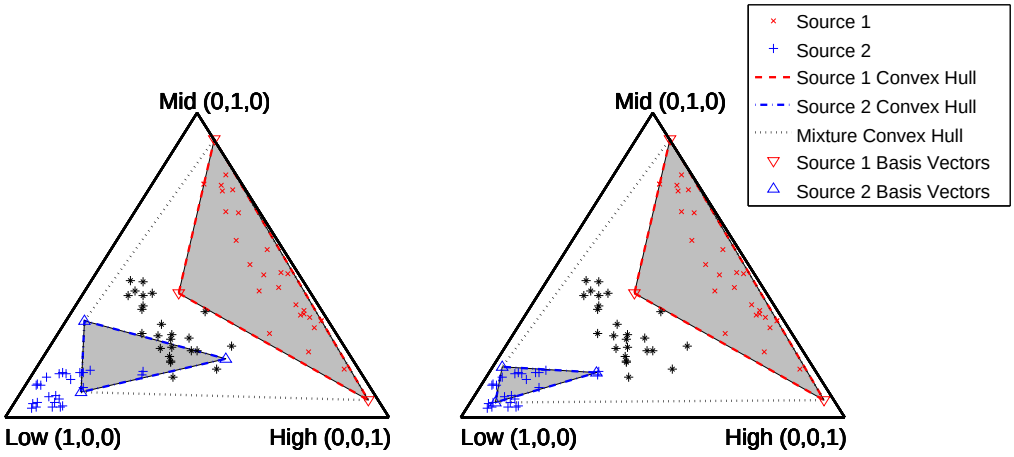


Figure 4.6: A 2-simplex diagram illustrating semi-supervised separation. The basis vectors of the first source are learned in a supervised manner, while the basis vectors of the second source are learned via semi-supervised learning. The proposed method (right) results in better estimates of the second source basis vectors compared to standard PLCA (left).

4.4 Compete Separation Algorithm

To perform our complete separation algorithm, we need to run a modified version of the original NMF/PLVM separation procedure, which allows us to both elicit and incorporate the user-annotations. The complete process is specified in Algorithm 15. As before, we define an alternative interface to the NMF/PLCA estimation procedure to allow an optional input argument of pre-learned basis vectors, which are held fixed. In addition, we also use the source filtering approach to separation and use the output of our NMF/PLVM model to create a time-frequency masking filter, as described in Section 2.7.2. The filter for each source is then applied to the original mixture recording to complete the separation process.

Algorithm 15 Complete Interactive NMF/PLVM Source Separation

Procedure INTERACTIVE-SEPARATION (

$\mathbf{x}_m \in \mathbf{R}^{N_\tau}$, // time-domain mixture signal

$\mathbf{x}_s \in \mathbf{R}^{N_{\tau_s}}\ \forall s \in \{1, ..., N_{s_t}\}$, // $0 \geq N_{s_t} \geq N_s$ time-domain training signals

N_z, // number of basis vectors

N_s, // number of sources

P // STFT parameters

)

precompute:

// compute STFT and NMF-PLCA of training data

for all $s \in N_{s_t}$ **do**

$(\mathbf{X}_s, \angle \mathbf{X}_s) \leftarrow$ STFT($\mathbf{x}_s$, P)

$(\mathbf{W}_s, \mathbf{H}_s) \leftarrow$ NMF-PLCA($\mathbf{X}_s, N_z/N_s$)

end for

// compute STFT of mixture signal data

$(\mathbf{X}_m, \angle \mathbf{X}_m) \leftarrow$ STFT($\mathbf{x}_m$, P)

repeat

input: user-annotated penalties $\mathbf{\Lambda}_s \in \mathbf{R}^{N_f \times N_t},\ \forall\ s \in \{1, .., N_s\}$

$(\mathbf{W}, \mathbf{H}) \leftarrow$ PR-PLCA($\mathbf{X}_m$, $\mathbf{\Lambda}_s\ \forall\ s$, N_z, N_s, $[\mathbf{W}_1\ \mathbf{W}_2 \ldots \mathbf{W}_{N_{s_t}}]$)

for all $s \in N_s$ **do**

$\mathbf{F}_s \leftarrow \mathbf{W}_s\,\mathbf{H}_s\ /\ \mathbf{W}\,\mathbf{H}$ // compute filter

$\hat{\mathbf{X}}_s \leftarrow \mathbf{F}_s \odot \mathbf{X}$ // filter mixture

$\mathbf{x}_s \leftarrow$ ISTFT($\hat{\mathbf{X}}_s$, $\angle \mathbf{X}_m$, P)

end for

until satisfied

return: time-domain signals $\mathbf{x}_s,\ \forall s \in \{1, ..., N_s\}$

Chapter 5

Implementation and Software Design

5.1 Introduction

To fully embody our proposed source separation interaction paradigm and algorithm, we developed a new software system called ISSE - An Interactive Source Separation Editor [104]. ISSE is a free, open-source audio editing tool that allows a user to separate a single recording of a mixture of sounds into two sources using drawing and painting tools. In this chapter, we discuss implementation issues in Section 5.2, including third-party software libraries and computation speed, and present several screenshots of the current version of our software in Section 5.3. To download the software (application and/or code), please see `http://ccrma.stanford.edu/~njb/thesis`.

5.2 Implementation Details

The proposed system is written in the C++ programing language for efficiency purposes. It is built for OSX, Windows, and Linux operating systems (32 and 64-bit), and licensed under the GNU General Public License Version 3 [105]. Features include audio playback/transport control, spectrogram viewing with zoom controls, paintbrush tools, undo/redo, file saving/loading, mute/solo/volume control, and a fully multithreaded user interface and processing architecture. Current painting tools include time select, frequency select, box select, training select (used to select isolated training data if available), and a spray-paint

brush.

We also allow users to change the spectrogram visualization rendering settings such as the color map and clip limit, to change the painting colors and adjust the audio input/output routing and latency settings. In addition, a majority of the main functionality of the software has keyboard shortcuts to enable fast usage.

5.2.1 Third Party Libraries

The software is heavily dependent on several third-party, open-source libraries, including JUCE [106], Eigen [107], and FFTW [108]. JUCE, or Jules Utility Class Extensions, is a large, open-source, cross-platform C++ class library and is used for all user-interface and audio-playback functionality. Eigen is a high-performance, open-source, cross-platform C++ template library for linear algebra, matrices, vectors, solvers, and related algorithms and is used for the core signal processing/machine learning-based separation algorithm. FFTW is an open-source, cross-platform C subroutine library for computing fast Fourier transforms and is used for all fast Fourier transform operations needed for both the spectrogram displays and the separation algorithm.

5.2.2 Computation Speed

Finally, because of our emphasis on interactivity, we briefly comment on the computational speed of our proposed approach. Fig. 5.1 depicts the time it takes our system to react to a user annotation, re-estimate the separation results, and present the results back to the user. Results are shown as a function of the input file length using high-quality (44.1kHz sampling rate) and low-quality (16kHz sampling rate) settings of the algorithm. We also plot the file lengths and computation time of the five tasks of our user study (to be discussed).

As shown, our algorithm is unfortunately linearly dependent on the input file length. Even though our algorithm is faster than real-time, this limits the degree of interactivity for files longer than 20-30 seconds. Fortunately, however, the recent work of Battenberg and Wessel [109] shows that graphics processing unit (GPU) implementations of similar separation algorithms can be increased over 30x, potentially allowing for a much higher-degree of feedback in future implementations.

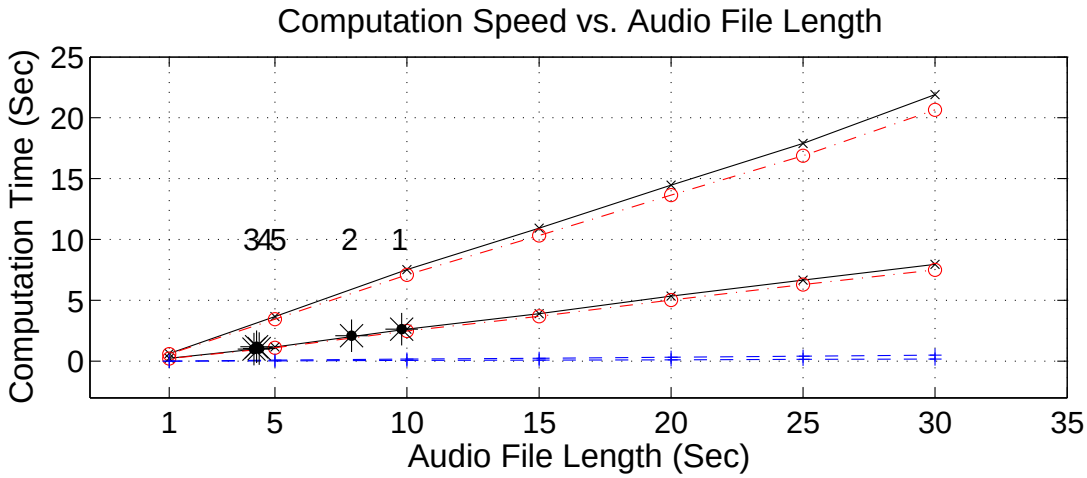


Figure 5.1: Computation Speed. Overall time (black, x, solid line), PLCA time (red, circle, dash-dot line), STFT time (blue, plus, dashed line), and user studies tasks (black, star, numbers) are shown for high-quality (44.1kHz, slower lines) and low-quality (16kHz, faster lines) audio sample rates.

5.3 Screenshots

In this section, we provide several screenshots of our current user-interface. The screenshots include the Multi Paint View in Fig. 5.2, the Single Paint View in Fig. 5.3 and Fig. 5.4, and the Settings View in Fig. 5.5.

In the Multi Paint View, a user can listen, view, and paint on the input mixture recording (top track) and the separated outputs (middle and bottom track). In the Single Paint View, a user can view a single, selected track from the Multi Paint View on a larger display (either the input or one of the two outputs). These two views encompass the central functionality of ISSE, in addition to the Settings View, which allows a user to dynamically control properties of the separation algorithm, such as the short-time Fourier transform or other parameters, and the number of basis vectors.

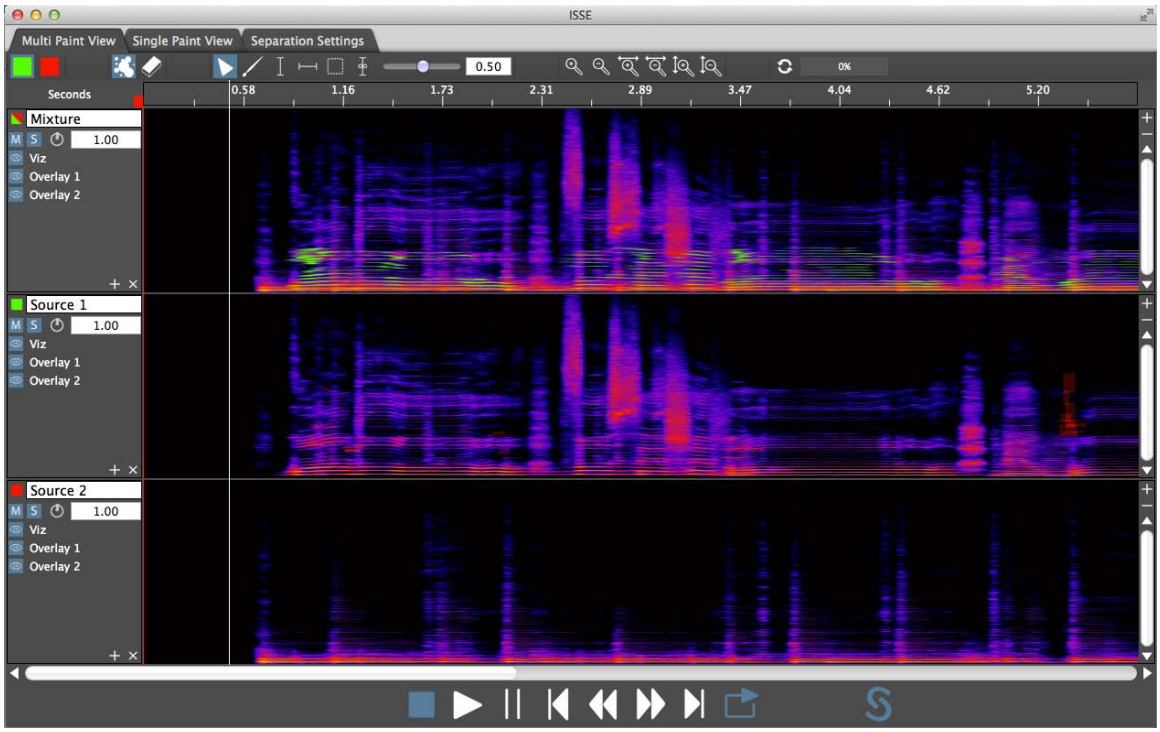


Figure 5.2: The Multi Paint View. In the Multi Paint View, a user can listen and annotate both the input and outputs of the separation process.

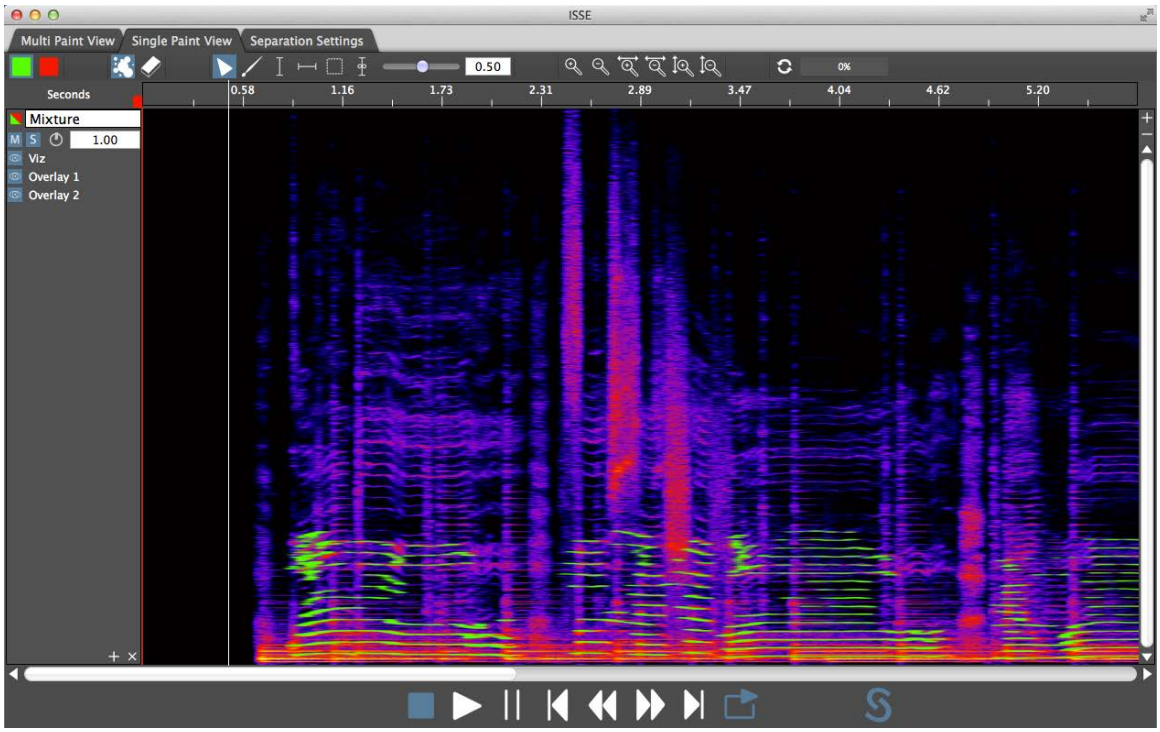


Figure 5.3: The Single Paint View. In the Single Paint View, a user can view and listen to a single, selected track on a larger display.

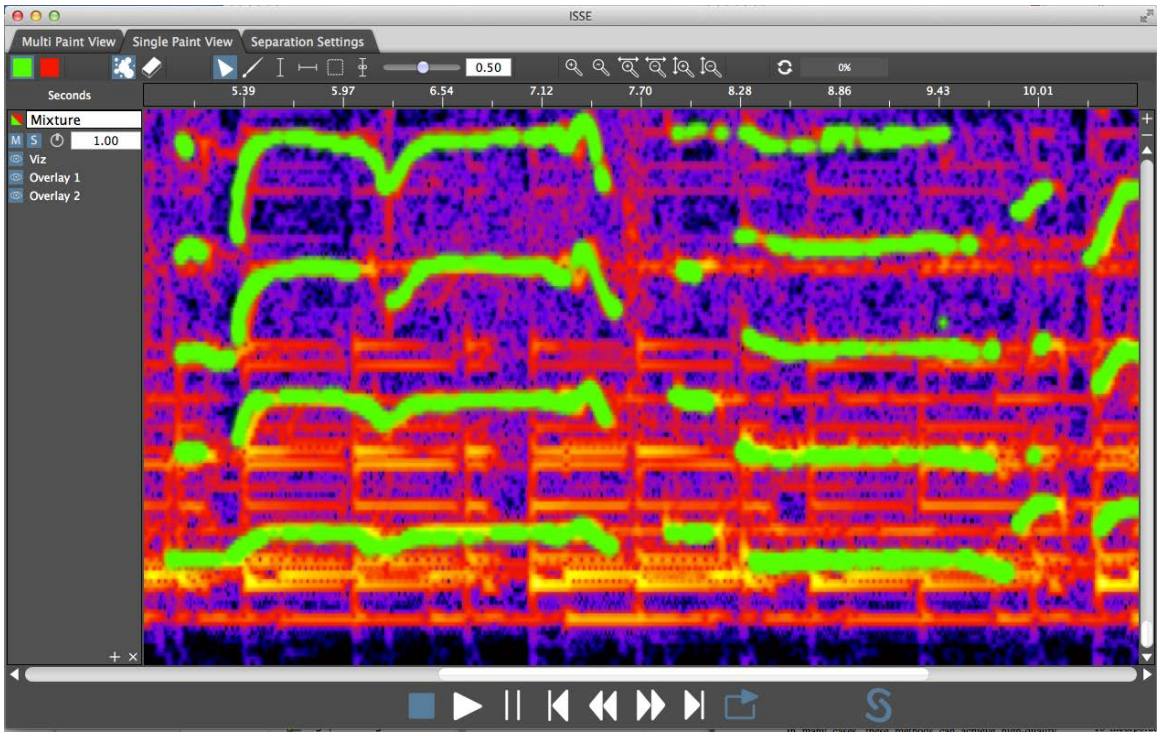


Figure 5.4: The Single Paint View zoomed in. In this example, the harmonics of a voice, which are annotated in green, have a great deal of time-frequency overlap with the harmonics of a guitar in the background.

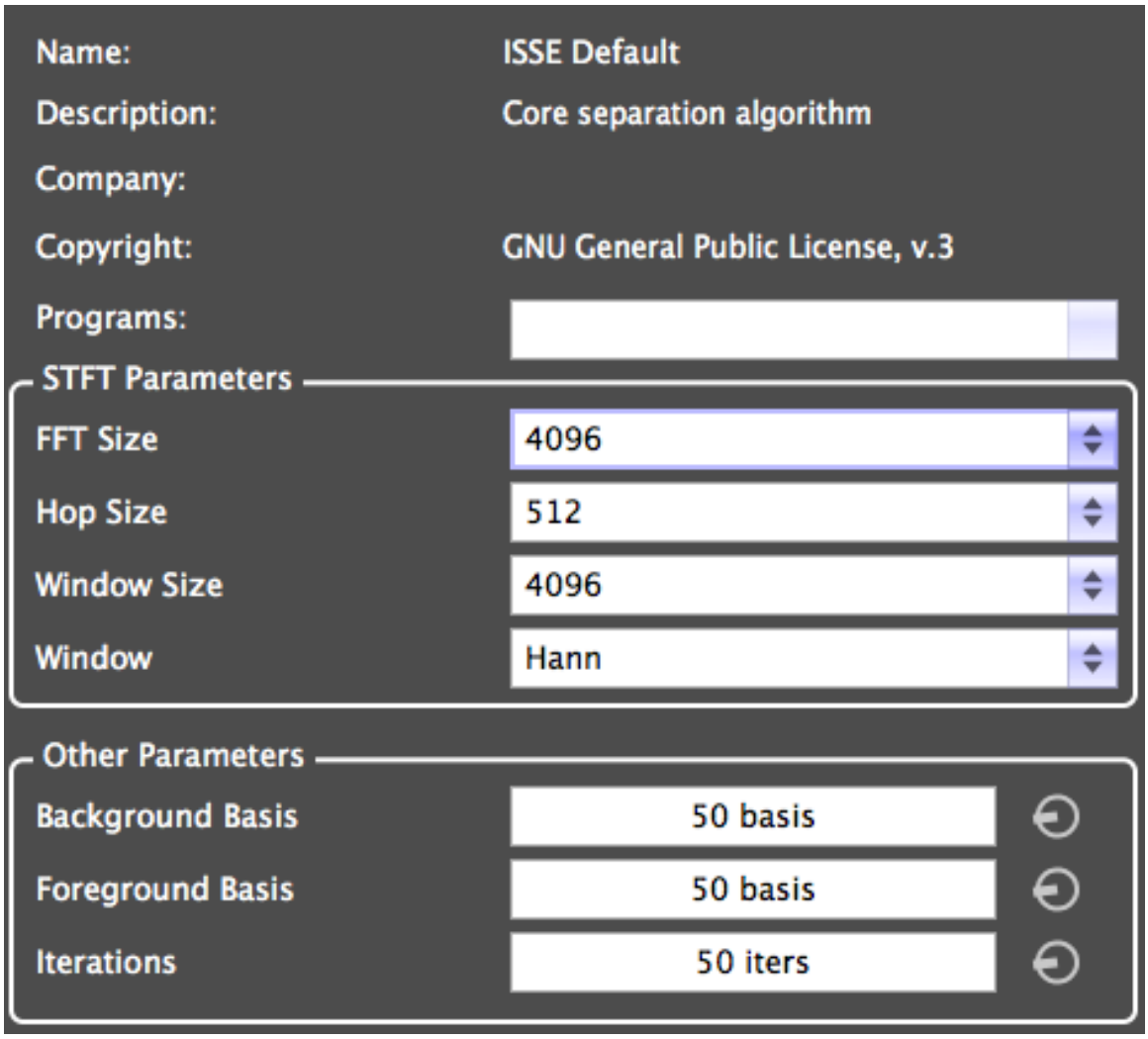


Figure 5.5: The Settings View. In this view, different parameters of the separation can be dynamically modified.

Chapter 6

Evaluation

6.1 Introduction

To evaluate our proposed methodology and developed software system, we performed an initial separation quality comparison, performed user studies to test novice users' ability to use our software, submitted separation results to an international separation evaluation campaign, and produced several additional musical sound examples. We discuss the results from each in Sections 6.3, 6.4, 6.5, and 6.6, respectively. Before each of those sections, however, we first outline our evaluation methodology in Section 6.2.

6.2 Evaluation Methodology

In order to effectively measure separation quality, we use a suite of objective evaluation metrics as well as a suite of pseudo-subjective evaluation scores. The first set of metrics is called the BSS-EVAL metrics for blind signal separation evaluation, developed by Vincent et al. [110], and the second set of metrics is called the PEASS scores, which stands for Perceptual Evaluation methods for Audio Source Separation and was developed by Emiya et al. [111]. We discuss each in Section 6.2.1 and Section 6.2.2, respectively.

Using these metrics, we compare our approach to past methods, a standard baseline algorithm, and standard ideal oracle algorithms. Comparing against past approaches allows us to see how well we perform relative to prior work. Comparing against a standard baseline

allows us to get an idea of how well we are performing relative to an algorithm that achieves no separation. Comparing against ideal oracle algorithms allow us to compare our results to the best possible results for the general class of algorithms we are using (i.e., time-frequency soft mask filtering). The latter point is of significant interest because the BSS-EVAL metrics and PEASS scores do not provide a normalized, scaled score (i.e., between zero and one), are dependent on the recording length, and are dependent on the original source material, making it difficult to judge what is a low and high score. We discuss the baseline algorithm in Section 6.2.3 and the ideal algorithm in Section 6.2.4.

6.2.1 BSS-EVAL Metrics

To measure the separation quality, we first use the standard BSS-EVAL suite of objective source separation evaluation metrics [110]. The suite includes several separate measurements including the Source-to-Interference Ratio (SIR), Source-to-Artifacts Ratio (SAR), and Source-to-Distortion Ratio (SDR). The SIR measures the level of suppression of the unwanted sources, the SAR measures the level of artifacts introduced by the separation process, and the SDR gives an average measure of separation quality that considers both the suppression of the unwanted sources and the level of artifacts introduced by the separation algorithm compared to ground truth. All three of the metrics are statistical measures and are computed by comparing the estimated separated source signals to the original unmixed recordings of a given mixture sound.

In more detail, the SDR, SAR, and SIR are computed by decomposing the estimated separated source signals $\hat{\mathbf{x}}_s$ via

$$\hat{\mathbf{x}}_s = \mathbf{x}_{target} + \mathbf{e}_{interf} + \mathbf{e}_{noise} + \mathbf{e}_{artif}, \tag{6.1}$$

where $\mathbf{x}_{target}$ is the target ground-truth source within a specified tolerance, $\mathbf{e}_{interf}$ is an interference error signal, $\mathbf{e}_{noise}$ is a noise error term, and $\mathbf{e}_{artif}$ is an artifact error term. To compute each of these terms, a signal projection-based scheme is used, with details found in the work of Vincent et al. [110]. The individual elements of the decomposition are then used in various forms to compute the SDR, SAR, and SIR metrics, which all have units of decibels (dB) and consider higher values to be better.

The Source-to-Artifacts Ratio (SAR) is defined as

$$\text{SAR} := 10\log_{10}\frac{||s_{target}+e_{interf}+e_{noise}||^2}{||e_{artif}||^2}, \tag{6.2}$$

the Source-to-Interference Ration (SIR) is defined as

$$\text{SIR} := 10\log_{10}\frac{||s_{target}||^2}{||e_{interf}||^2}, \tag{6.3}$$

and the Source-to-Distortion Ration (SDR) is defined as

$$\text{SDR} := 10\log_{10}\frac{||s_{target}||^2}{||e_{interf}+e_{noise}+e_{artif}||^2}. \tag{6.4}$$

6.2.2 PEASS Metrics

In contrast to the completely objective BSS-EVAL metrics, an alternative approach for evaluation is the PEASS evaluation scores developed by Emiya et al. [111]. The PEASS scores attempt to measure the perceived quality of the estimated source signal using four individual measures: the Overall Perceptual Score (OPS), the Interference-related Perceptual Score (IPS), the Artifact-related Perceptual Score (APS), and the Target-related Perceptual Score (TPS).

They do so by correlating objective statistical measures such as the SDR, SAR, and SIR metrics with perceptual listening test results in an effort to learn a perceptual scoring function. The learned scoring function is then applied to new, unforeseen separation results in a standard fashion. The OPS, IPS, and APS are analogous to the SDR, SIR, and SAR, respectively, and the TPS corresponds to something called the image-to-spatial distortion ratio (ISR), which measures spatialization accuracy. Because the technical development of these scores is somewhat involved, we do not address their operation further.

6.2.3 Baseline Separation Algorithm

To compute our baseline separation algorithm, we use a standard unsupervised NMF/PLVM-based separation algorithm, as discussed in Section 2.6.1. We can think of this baseline as

an empirical lower bound on separation quality for which we should always perform better (though it is possible to perform worse than the baseline). Also, using a standard unsupervised separation algorithm as a baseline makes it easy for us to compare our approach with and without user-interaction and directly see the benefit of interaction. Note, however, that the baseline algorithm would never be used in practice.

When we use an unsupervised NMF/PLVM-based separation, we must decide the divergence function, the STFT parameters, and the number of basis vectors we allocate to each source within a mixture. In this work, we always use the KL divergence and set the STFT parameters and number of basis vectors to be the same as whatever separation algorithm we are comparing against. If the opposing separation algorithm does not have an option to choose the number of basis vectors and/or STFT parameters, we set the number of basis vectors to a large number (e.g., 50 basis vectors per source) and the STFT parameters according to common conventions.

6.2.4 Ideal Oracle Separation Algorithm

To compute our ideal oracle separation algorithm, we use the ideal soft mask algorithm. This algorithm replaces our NMF/PLVM separation algorithm and simply uses the ground truth unmixed source signals that compose a given mixture to "perform separation." More specifically, the ground truth recordings are used to compute the source filter $\mathbf{F}_s$ as described in Section 2.7.2, using the following procedure:

1. Compute the STFT magnitude $\mathbf{V}_m$ of a given mixture recording.
2. Compute the STFT magnitude of each ground truth source recording $\mathbf{V}_s$ that was used to create the mixture.
3. Create the source filter $\mathbf{F}_s$ via
$$\mathbf{F}_s = \frac{\mathbf{V}_s}{\mathbf{V}_m}. \tag{6.5}$$

This filter is called an ideal soft mask filter and is applied to the original mixture STFT magnitude and phase in standard fashion. Alternatively, the filter $\mathbf{F}_s$ can be quantized so all elements of are either zero or one. This quantized form, called an ideal binary mask, is

common in the computational auditory scene analysis and speech denoising communities, and is used for comparison in the signal separation evaluation campaign.

When the soft mask filter is used, notice that all we are doing is computing the STFT of each source signal, replacing the phase of an individual source with the mixture phase, and then inverting back to the time-domain to create the estimated separation estimates. This simulates a perfect estimation of the source signal STFT magnitudes and an imperfect phase estimate. Because we simply use the mixture phase in our proposed approach, the ideal soft mask separation algorithm gives us an empirical upper bound for which we can strive to achieve (note, however, it is possible to perform better than the oracle).

6.3 Initial Results

To test the proposed method using these evaluation metrics, we used our developed software system to perform separation on two test sets of sound examples. For the first comparison, denoted "example suite," we created five real-world mixture sounds, each with two sound sources, and then performed unsupervised, supervised, and semi-supervised separation with and without user-interaction. This allows us to compare how each method fares when compared to one another and the ideal soft mask algorithm. For the second comparison, we tested four example rock/pop songs from the Signal Separation Evaluation Campaign (SiSEC) 2011 database [112] with the challenge of removing vocals from background music over the course of thirty minutes. For a full description of each test, please see Section 6.3.1 and Section 6.3.2, respectively.

6.3.1 Example Suite

For our example suite, we created five mixture sounds, including: ambulance siren + speech (S), cell phone ring + speech (C), drum + bass loop (D), orchestra + coughing (O), and piano chords + incorrect piano note (P). To do so, the original ground truth sources for each example were normalized to have a maximum of 0 dB gain and summed together to create the mixture sound. Then, we used our developed software system to separate the mixture

recordings using unsupervised, supervised, and semi-supervised separation with and without user-interaction and a fixed number of basis vectors $N_z = 100 + 100$. For supervised separation, the original unmixed tracks were used for training. For semi-supervised separation, we used the regions of the mixture data in which only one source was present for training data.

We display the complete SDR, SAR, and SIR evaluation results in Table 6.1, 6.3, and 6.2. We also illustrate an example set of input and output spectrograms in Fig. 6.1 for the case of the cell phone example using semi-supervised separation. Notice how the proposed method significantly outperforms standard PLCA with minimal user interaction (only a single harmonic of a single ring is annotated).

In all cases, the user-interaction increases the SDR, SAR, and SIR for supervised, semi-supervised, and unsupervised separation. The improvement in performance is greatest for unsupervised separation, then semi-supervised separation, and then supervised separation, as expected. In certain cases, our method even approached the quality of the ideal mask. When we analyze which of the learning procedures performs best, we notice that typically supervised separation with user-interaction performs best. Surprisingly, supervised separation with interaction, however, is not always the winning algorithm. Occasionally, unsupervised separation with user-interaction can actually perform best.

EXAMPLE	IDEAL	SUPERVISED	SEMI-SUPERVISED	UNSUPERVISED
CELL	30.7	29.2 / 27.6	28.4 / 06.5	28.8 / -0.6
DRUM	14.8	09.7 / 08.5	07.7 / 03.9	10.0 / 00.2
COUGH	15.8	14.0 / 12.5	12.0 / 10.5	13.8 / -2.1
PIANO	26.1	26.0 / 21.6	14.9 / 08.4	23.1 / 01.1
SIREN	27.8	23.8 / 18.9	21.0 / 19.9	24.2 / -4.2

Table 6.1: Initial SDR (dB) results with (left) and without interaction (right).

EXAMPLE	IDEAL	SUPERVISED	SEMI-SUPERVISED	UNSUPERVISED
CELL	39.6	39.8 / 33.3	39.1 / 19.1	39.6 / 1.1
DRUM	20.5	17.9 / 12.1	16.5 / 8.8	19.6 / 1.4
COUGH	22.6	20.8 / 16.2	18.6 / 17.3	21.9 / 1.8
PIANO	29.9	29.8 / 24.5	21.7 / 9.7	29.2 / 1.6
SIREN	36.2	33.7 / 23.8	29.1 / 26.0	36.0 / 2.5

Table 6.2: Initial SIR (dB) results with (left) and without interaction (right).

EXAMPLE	IDEAL	SUPERVISED	SEMI-SUPERVISED	UNSUPERVISED
CELL	31.3	29.7 / 29.2	28.9 / 15.8	29.2 / 11.4
DRUM	16.3	10.7 / 13.9	8.5 / 12.1	10.6 / 9.9
COUGH	17.4	15.2 / 15.6	14.4 / 11.6	15.3 / 7.3
PIANO	28.9	28.9 / 25.0	15.9 / 14.6	25.3 / 14.4
SIREN	28.7	25.3 / 20.7	22.1 / 21.6	25.2 / 6.2

Table 6.3: Initial SAR (dB) results with (left) and without interaction (right).

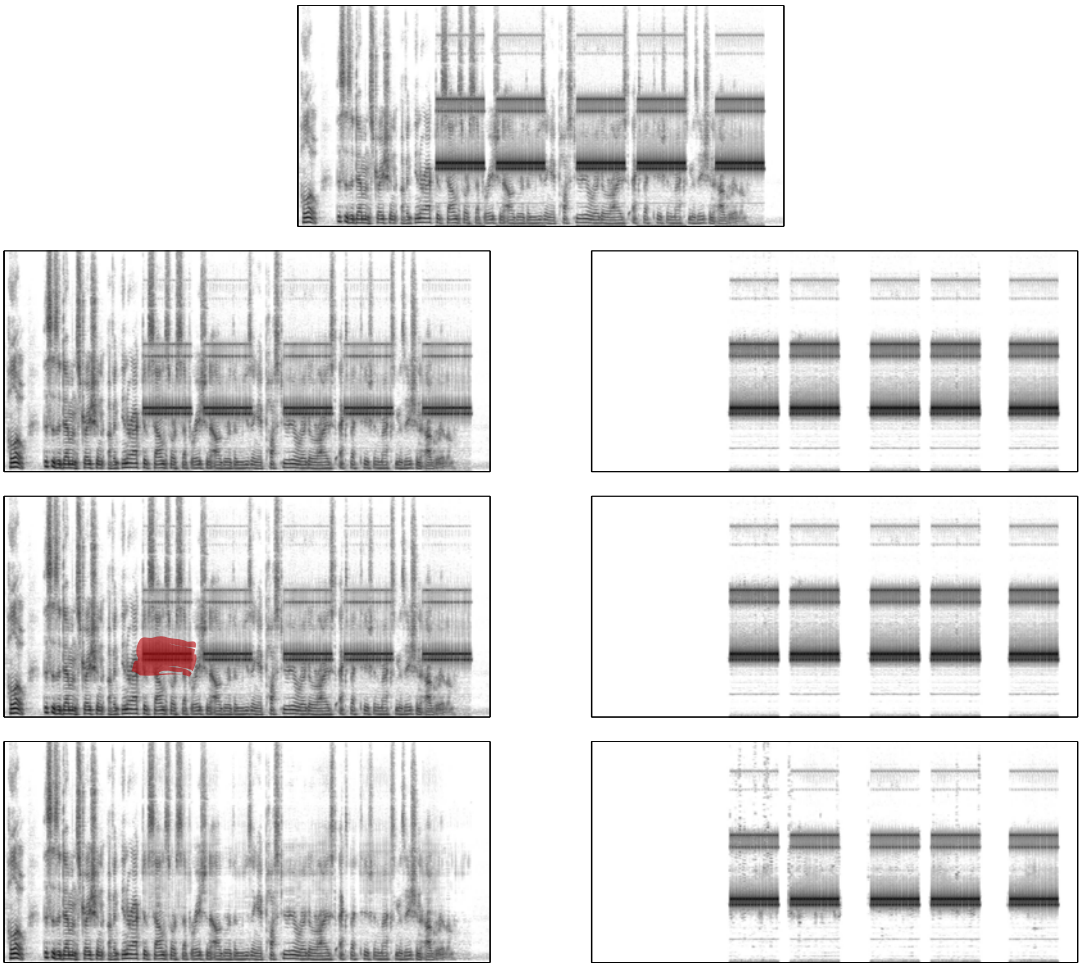


Figure 6.1: (First Row) A mixture spectrogram of speech + cell phone. (Second Row) Initial separated speech (left) and cell phone (right) after semi-supervised PLCA. (Third Row) Painting annotation overlaid on the incorrectly separated regions. Note: only a single harmonic of a single ring is annotated. (Bottom) Refined separated speech and cell phone.

6.3.2 Vocal Separation

For a second test, we tested our proposed separation algorithm on the task of separating vocals from background music. We used four example rock/pop songs (S1, S2, S3, S4) from the training set of the Signal Separation Evaluation Campaign (SiSEC) 2011 database [112]. We then used our software system over the course of thirty minutes to perform separation using fixed number of basis vectors $N_z = 100 + 100$, and then compared our method

against competitive user-guided separation methods, including the method of Lefèvre et al. [32] and Durrieu et al. [31].

The complete the SDR, SIR, and SAR results are shown in Tables 6.4, 6.5, and 6.6, respectively. As shown, our proposed method outperforms the baseline, the method of Lefèvre, and the method of Durrieu in all metrics for all examples. Note, the method of Durrieu previously ranked best SDR on average for the 2011 SiSEC evaluation campaign for removing vocals. In addition, in certain cases, the proposed method is only one or two decibels away form the ideal soft mask, which is a promising result.

EXAMPLE	ORACLE	BASELINE	LEFÈVRE	DURRIEU	PROPOSED
SONG 1	13.2	-0.8	7.0	9.0	9.2
SONG 2	13.4	0.2	5.0	7.8	11.1
SONG 3	11.5	-0.2	3.8	6.4	7.8
SONG 4	12.5	1.4	5.0	5.9	7.9

Table 6.4: SDR (dB) results for vocal extraction of four SiSEC rock/pop songs.

EXAMPLE	ORACLE	BASELINE	LEFÈVRE	DURRIEU	PROPOSED
SONG 1	17.8	0.5	13.0	16.4	17.4
SONG 2	18.0	1.6	14.1	16.8	20.1
SONG 3	17.5	0.9	8.8	13.0	14.8
SONG 4	19.5	3.1	11.5	12.6	13.8

Table 6.5: SIR (dB) results for vocal extraction of four SiSEC rock/pop songs.

EXAMPLE	ORACLE	BASELINE	LEFÈVRE	DURRIEU	PROPOSED
SONG 1	15.4	8.9	8.9	10.5	10.7
SONG 2	15.4	8.5	7.3	9.0	12.0
SONG 3	13.1	8.8	6.1	8.0	9.0
SONG 4	13.6	10.0	6.5	8.3	9.5

Table 6.6: SAR (dB) results for vocal extraction of four SiSEC rock/pop songs.

6.3.3 Model Selection

Finally, to show how the proposed method behaves when varying the number of basis vectors per source, we performed separation on the example suite, then with the annotations fixed, varied the number of basis vectors and recomputed the results. Fig. 6.2 displays the SDR for the experiment, which shows that the method is relatively insensitive to changes in N_z, as long as the size is sufficiently large. This is notable in that, unlike standard methods, the proposed method does not require the use of model selection to decide the number of basis vectors to use for a given separation task.

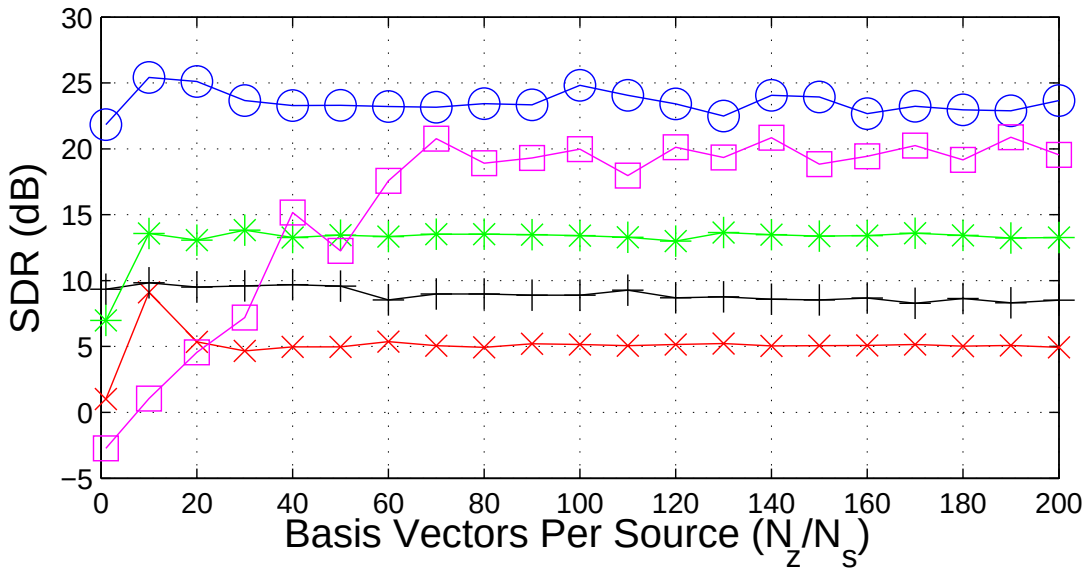


Figure 6.2: Comparison of SDR (in dB) to the number of basis vectors per source. Examples include Phone (blue, circle), Drum (red, x-mark), Orchestra (black, plus), Piano (green, star), and Siren (magenta, square).

6.4 User Studies

To further test the proposed system, we designed a user study with the following questions in mind: Can inexperienced users with a music and audio background use the proposed system to achieve a reasonable level of separation quality? How does this separation quality compare to the quality achieved by experienced, expert users of the system? And what is

the maximum achievable separation quality of the system?

6.4.1 Methodology

To answer these questions, we first studied how inexperienced users (with music and audio backgrounds) performed on a variety of separation tasks using the proposed method. The specifics of the study are described below and were approved through standard institutional review board procedure.

Participants.

We recruited 10 participants from the local-users@ccrma.stanford.edu email listserv at the Center for Computer Research in Music and Acoustics (CCRMA) at Stanford University with the incentive of a $\$25$ gift card. The participants had significant, but varying levels of musical training, audio editing skills, and knowledge of spectrogram displays of sound, but no past experience with our system. The level of formal musical training varied between 0-30 years and averaged approximately 12 years. The level of experience in music production varied between 1-15 years and averaged approximately 4 years. And finally, the level of experience with spectrogram displays ranged between $\frac{1}{2}$-10 years and averaged approximately 4 years. In addition to the 10 participants, an expert user of our system (the author of this thesis) was also tested for comparison. The expert user had 10 years of music production and editing background and hundreds of hours of experience with/in working and designing the system.

Training.

To train each participant, we presented him or her with a short, five-minute, introductory video. The video outlined the general functionality of the system and provided three demonstrative examples. Following the video, a standardized description of what would be required of the user was read aloud. After, we held a short, five-minute question-and-answer session.

Tasks.

Once the training was complete, five real-world separation tasks were given to each user. Each of the five tasks required the user to separate a mixture recording of two sounds into its respective sources over the course of ten minutes. At the end of each task, the separation results were saved and stored for later analysis. At any time, a user was allowed to ask questions on the functionality of the system (mechanics of buttons, sliders, etc.). The mixture sounds used for each task were arranged in order of difficulty and included: (Task 1) a cell phone + speech, (Task 2) ambulance siren + speech, (Task 3) bass guitar + drums, (Task 4) cough + orchestra, and (Task 5) vocals + guitar.

Debrief.

Once the five tasks were completed, each user was given a short background questionnaire and debriefing survey. The debriefing survey was used to gauge the difficulty and satisfaction level of each task on a scale of one to five and record users' overall experience using the system. Questions included: 1) Did you feel like your ability to separate sounds improved over time? 2) What was the most difficult aspect of the system? 3) What was the most fun aspect of the system? 4) Additional comments?

Scoring Success

To measure the separation quality achieved by our participants and compare the result to those of an expert user, we use the standard BSS-EVAL metrics discussed above. To make it easier to compare separation quality across different tasks, we then normalize these metrics for a given task by computing the separation quality from our oracle algorithm and subtracting it off from the individual results. We do this because the SDR, SAR, and SIR metrics do not natively provide an upper bound score for separation quality. In addition to this empirical pseudo upper bound, we also test our baseline lower bound algorithm. Both benchmarks help us get a better idea of how well our participants performed.

6.4.2 Results

The results of the user studies are presented in two forms: using standard source separation objective metrics, as described above, and via user responses from the debriefing surveys.

Objective Separation Quality

The computed SDR, SAR, and SIR results for each participant/task are shown in Fig. 6.3, Fig. 6.4, and Fig. 6.5 respectively. The results for each participant, the participant average, and the participant standard deviation, are reported alongside the results for the expert user, ground truth method, and no-interaction baseline method.

Out of these three figures, the most notable is Fig. 6.3, which gives us an overall measure of separation quality. From this figure, we can view several interesting observations. Firstly, as expected, the expert user outperformed nearly all inexperienced users in all tasks. What is unexpected, however, is that in more than one instance, a select few inexperienced users actually outperformed the expert. Given our additional evaluation regarding expert user results discussed below, this is an exciting result.

Secondly, there are four tasks (1, 2, 3, and 4) in which one or more participants achieved separation results within 5dB of the ideal result. While this benchmark is somewhat difficult to translate into perceptual quality, this type of performance is similar to, or only slightly worse than, state-of-the-art separation quality reported in the community-based signal separation evaluation campaign discussed below, albeit for easier tasks. And thirdly, in four out of five tasks, the average inexperienced user outperformed the no-interaction baseline by over 5dB and in two out of five cases, outperformed the baseline by nearly 15dB or more.

Similar phenomena are found in Fig. 6.4 and Fig. 6.5, and together give us an initial indication that inexperienced users can achieve good separation quality with minimal instruction. It should also be noted that in Fig. 6.5, the SAR with no interaction is occasionally better than user scores. This is because the results from the no-interaction are almost identical to the input (no separation). If no separation happens, there will be minimal artifacts introduced by the algorithm. Of course, this means that, without interaction, the system is not useful. The corresponding SIR scores reflect this fact and are proportionally

low in those cases.

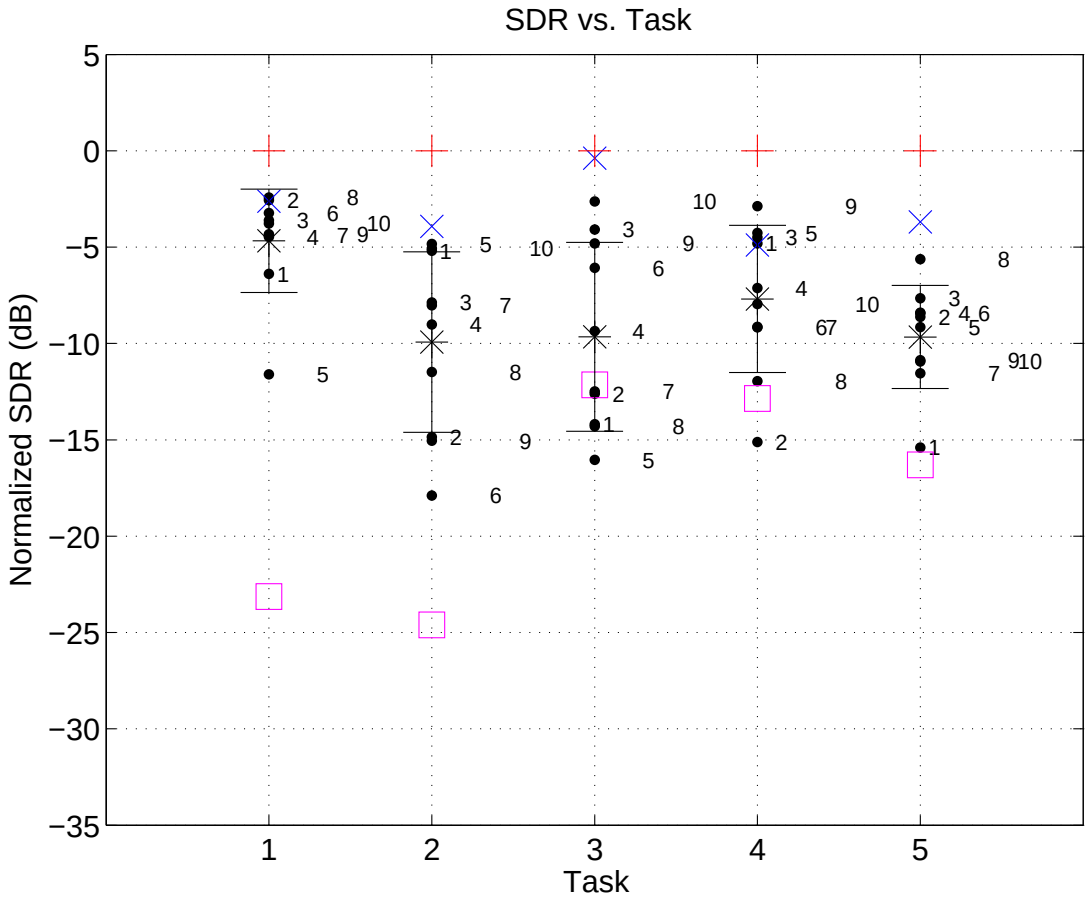


Figure 6.3: Normalized SDR results. Inexperienced-user scores (black, dots/numbers), inexperienced-user average scores (black, star), inexperienced-user one standard deviation (black, line), expert-user scores (blue, x), ideal scores (red, plus), and no-interaction scores (magenta, square) are shown for each task.

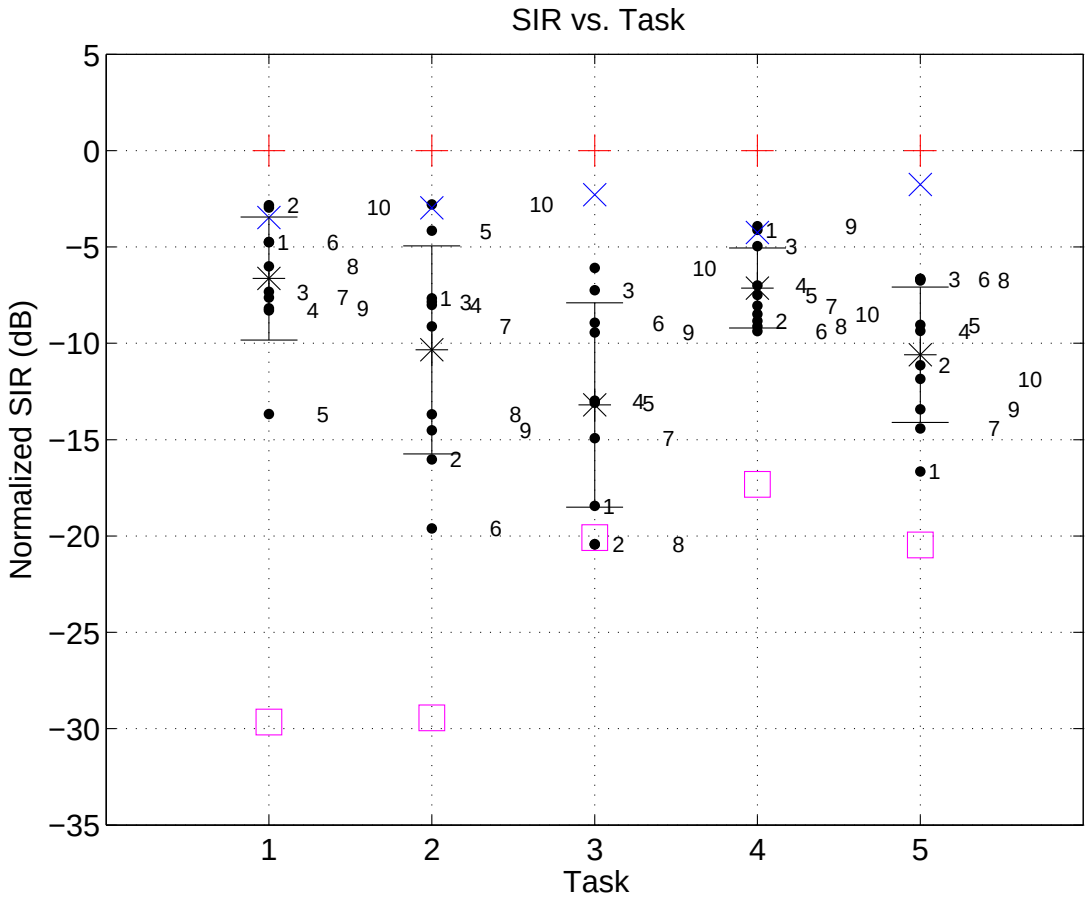


Figure 6.4: Normalized SIR results. Inexperienced-user scores (black, dots/numbers), inexperienced-user average scores (black, star), inexperienced-user one standard deviation (black, line), expert-user scores (blue, x), ideal scores (red, plus), and no-interaction scores (magenta, square) are shown for each task.

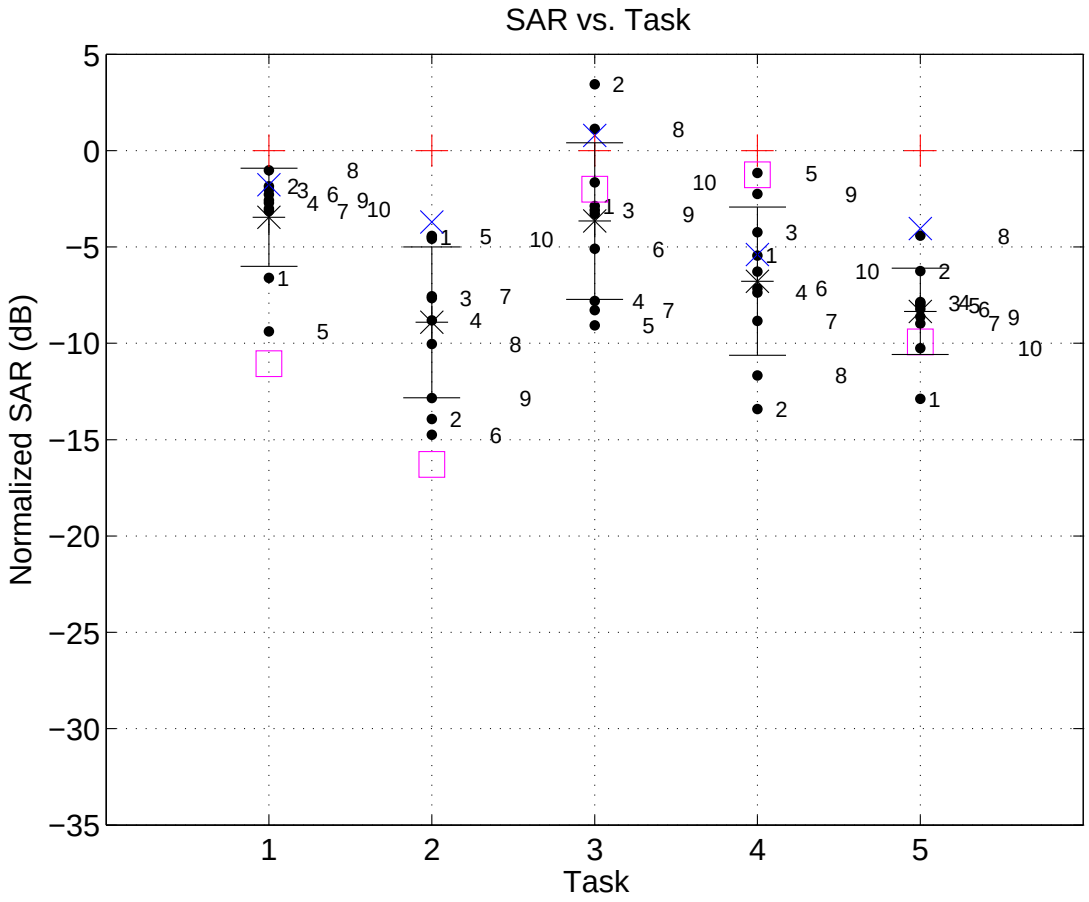


Figure 6.5: Normalized SAR results. Inexperienced-user scores (black, dots/numbers), inexperienced-user average scores (black, star), inexperienced-user one standard deviation (black, line), expert-user scores (blue, x), ideal scores (red, plus), and no-interaction scores (magenta, square) are shown for each task.

User Responses

In addition to the objective separation evaluation metrics discussed above, it is also interesting to look at the results from the debriefing surveys. We first discuss the participant's rating of difficulty and satisfaction of each task in Fig. 6.6 and Fig. 6.7, and then address the remaining follow-up questions.

In Fig. 6.6, we can see a steep increase in the perceived difficulty of each task over the course of the entire experiment, as intended by design. When we correlate this to the separation quality discussed above, it is interesting to see that while the perceived difficulty of each task increases, the average SDR stays more or less the same. This indicates that the average user was able improve his or her ability to separate sounds over the course of the study, and coincides with the fact that all users self-reported that their ability to separate sounds improved. Both observations further suggest that inexperienced users can learn to use the system in a relatively short amount of time.

In Fig. 6.7, we can see the reported satisfaction regarding the separation quality of each task. For the first two easier tasks, most users gave a satisfaction rating of 4/5. For the more difficult tasks, user satisfaction decreased to a 3/5 level. This loosely suggests that the user-reported difficulty rating is correlated to user satisfaction.

Regarding the most difficult aspects of the system, nearly all users commented on the task of associating a sound to its visualization, as expected. Regarding the most fun aspects of the system, participants commented "When it worked!," "Interactively being able to control the separation," "This was not possible with the tools of the past!," and "The real-timeness of the software made everything fun and engaging."

Regarding additional comments, participants stated "This would be a useful teaching tool!," "real great work, some insight on what is actually happening may be useful for optimizing user activity," "Include progress of rendering in each window," and "This is awesome." In general, we found the users' responses to be positive, encouraging further study.

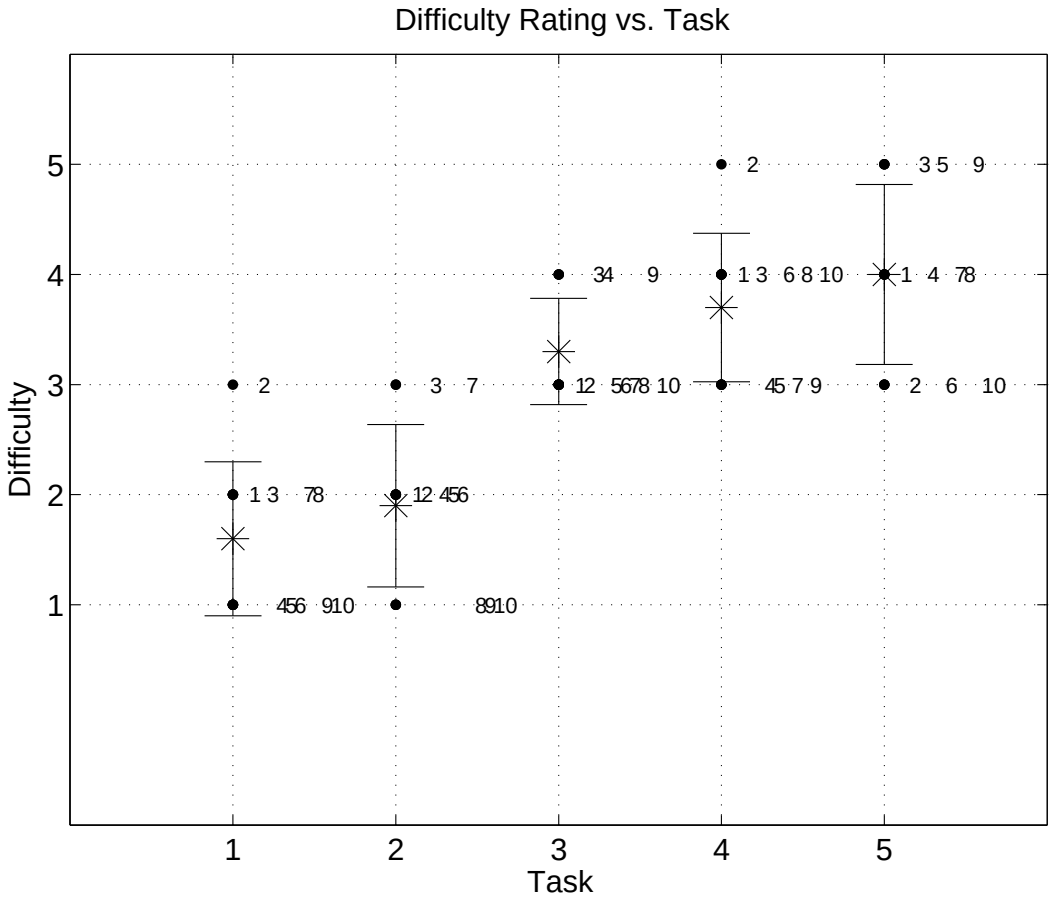


Figure 6.6: Reported user difficulty (black, dots/number) for each of the five separation tasks, alongside the average (black, star) and standard deviation rated difficulty (black, line). Notice how the average difficulty rating increases for each task.

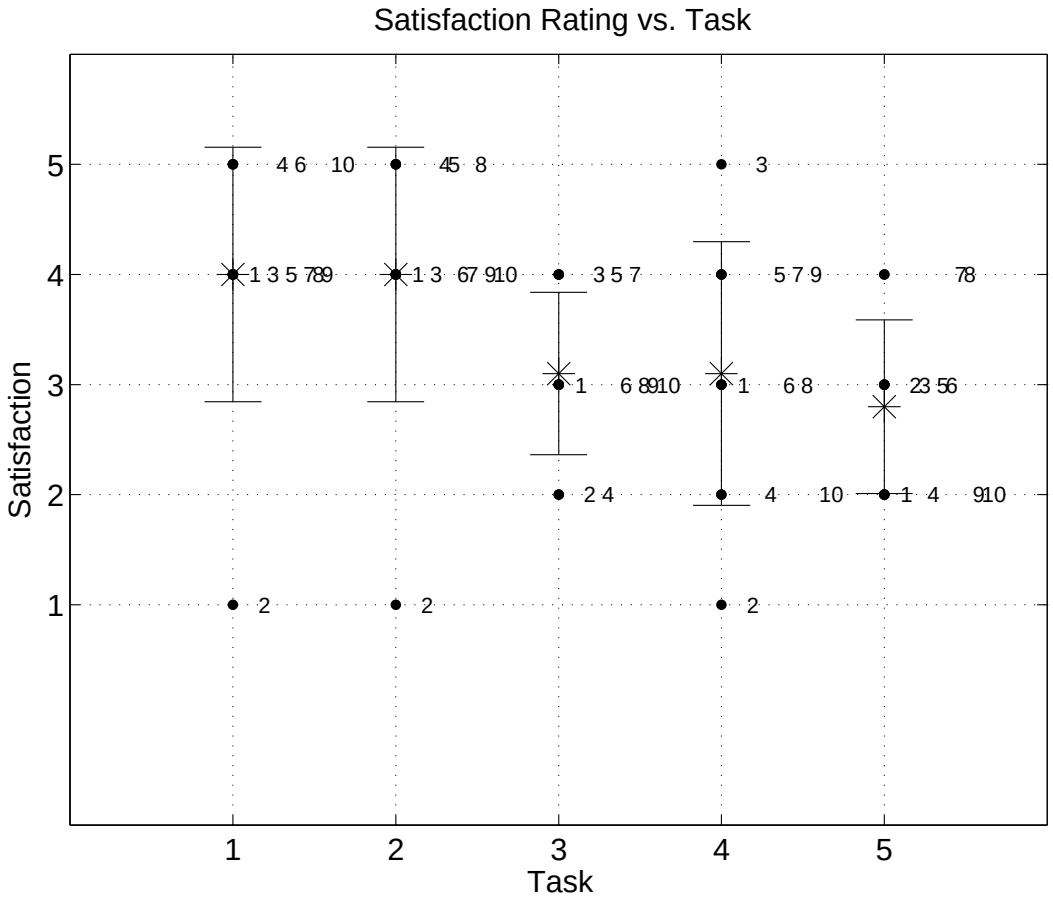


Figure 6.7: Reported user satisfaction (black, dots/number) for each of the five separation tasks, alongside the average (black, star) and standard deviation rated satisfaction (black, line).

6.5 Signal Separation Evaluation Campaign 2013 Results

For a more complete and objective evaluation, we submitted separation results to two separate tasks of the fourth signal separation evaluation campaign (SiSEC) 2013. The tasks include: professionally produced music recordings and two-channel mixtures of speech and real-world background noise. In both cases, we used our algorithm without any training

data (only user-interaction) and 50 basis vectors per source.

The task of separating professionally produced music recordings involved separating a test set of nine pop/rock music recordings. Seven out of nine recordings are short 20-30 second snippets of music. Two out of seven recordings are full duration versions, which are all simply longer versions of two short recordings (i.e., there is redundancy). Each recording typically consists of two, three, or four sound sources including vocals, guitar, bass, piano, and other. It was not required to submit results for all recordings or even all sources within a recording, making algorithm comparison difficult in some cases.

In total, fifteen algorithms were submitted, along with one oracle algorithm (ideal binary mask). The algorithms included: Algorithm 1 [113], Algorithm 2 [114], Algorithm 3 [115], Algorithm 4 [116], Algorithm 5 [116] , Algorithm 6 [117], Algorithm 7 [118], Algorithm 8 [33], Algorithm 9 [119], Algorithm 10 [120], Algorithm 11 [121, 122, 38], Algorithm 12 [121, 122, 38], Algorithm 13 [58, 57], Algorithm 14 [123], Algorithm 15 [123], and Algorithm 16 (reference/ideal binary mask).

Because the full-length recordings were simply longer versions of the short duration clips, we submitted separation results only for all sources of all shorter duration clips. This results in 24 total subtasks for which we can compare our proposed approach to alternative methods. When we do so, we find the following results.

- Overall results for songs submitted (excludes the two full-length clips).
  - Best SDRi 16/24 times. Next closest algorithm got best SDRi 4/24 times.
  - Highest average overall SDRi (2.6 dB above second best average)
  - Best ISRi 11/24 times. Next closest algorithm got best ISRi 6/24 times.
  - Highest average overall ISRi (2.7 dB above the second best average).
  - Best SIRi 17/24 times. Next closest algorithm got best SIRi 2/24 times.
  - Highest average overall SIRi (5.7 dB above the second best average).
  - Best SARi 12/24 times. Next closest algorithm got best SIRi 8/24 times.
  - Highest average overall SARi (2.3 dB above the second best average).
  - Best OPSi 9/24 times. Next closest algorithm got best OPSi 5/24 times.

  - Highest average overall OPSi (4.0 above second best average).

- Vocal results for songs submitted (excludes the two full-length clips).
  - Best Vocal SDRi 6/7 times.
  - Best Vocal OPSi 6/7 times.
  - Best average Vocal SDRi (1.8 dB above second best average).
  - Best average Vocal SARi (1.8 dB above second best average).
  - Best average Vocal SIRi (1.6 dB above second best average).
  - Best average Vocal ISRi (1.9 dB above second best average).
  - Best average Vocal OPSi (10.0 above second best average).
- Drum results for songs submitted (excludes the two full-length clips).
  - Best Drum SDRi 2/5 times.
  - Best Drum OPSi 1/5 times.
  - Best average Drum SDRi (1.1 dB above second best average).
  - Best average Drum ISRi (2.1 dB above second best average).
  - Best average Drum SIRi (1.9 dB above second best average).
  - Best average Drum SARi (2.2 dB above second best average).
- Bass results for songs submitted (excludes the two full-length clips).
  - Best Bass SDR 2/5 times. Next closest algorithm got best Bass SDR 2/5 times.
  - Best average Bass SDRi (1.0 dB above second best average).
  - Best average Bass ISRi (4.1 dB above second best average).
  - Best average Bass SIRi (2.6 dB above second best average).
  - Best average Bass SARi (2.7 dB above second best average).
  - Best average Bass APSi (7.2 above second best average)
- Piano results for songs submitted (excludes the two full-length clips).

  - Best Piano SDRi 1/2 times.
  - Best Piano OPSi 1/2 times.
  - Best average Piano SIRi (.3 dB above second best average).
  - Best average Piano OPSi (.3 above second best average).

- Guitar results for songs submitted (excludes the two full-length clips).
  - Best Guitar SDRi 1/1 times.
  - Best average Guitar SDRi (.8 dB above the second best average).
  - Best average Guitar SIRi (5.5 dB above the second best average).

- Other results for songs submitted (excludes the two full-length clips).
  - Best Other SDRi 4/4 times.
  - Best average Other SDRi (2.6 dB above second best average).
  - Best average Other SIRi (5.1 dB above second best average).

Because of these results, it is our belief that our proposed approach can claim state-of-the-art separation results for the task of professional produced music recordings. Note, however, that because the SiSEC is an evaluation campaign, no winners are announced.

For the task of separating two-channel mixtures of speech and real-world background noise, the final SiSEC results across algorithms were unfortunately difficult to compare. This is because there were several different subtests and most algorithms only submitted results to a small portion of the subtasks (ourselves included). As a result, we cannot adequately give a thorough analysis of the results as we did for the professionally produced music recordings tasks. To see all published results for the SiSEC 2013, please see [60].

6.6 Audio and Video Demonstrations

Finally, to demonstrate our proposed software system in action, we made an introductory demonstration video and several sound examples. Sound examples include separation results for vocals + guitar, drums + bass, Neil Armstrong's speech + background noise, piano

chords + wrong note, orchestra + cough, phone ring + speech, and vocals + drums + bass + other.

We also produced several short "mini-remixes" and "mini-mashups" to show how the proposed system can be useful for music production and remixing applications. To make the remixes and mashups, we used our proposed method to perform vocal extraction on several popular songs:

- "Last Kiss" by J. Frank Wilson and The Cavaliers
- "Paper Bag" by Fiona Apple
- "When I'm Gone" by Anna Kendrick
- "Gamble Everything For Love" by Ben Lee
- "Crazy Kids" by Ke$ha
- "I Want You Back" by The Jackson 5
- "Don't Stop Believing" by Journey
- "Last Friday Night" by Katy Perry
- "Wildflowers" by Tom Petty
- "Respect" by Aretha Franklin.

We then remixed the separated vocals with different instrumental background music.

Nearly all the remixes/mashups were created by separating approximately eight measures of vocals from previously existing pop songs. The Jackson 5 sound example was remixed with previously produced music (i.e., 'Want U Back" by Cher Lloyd). For all other examples, we created new original background music using Apple's Logic Pro software package.

The main reason for choosing the specific vocal examples was personal interest. Also, most of the chosen vocal clips have prominent vocals with standard pop song instrumentation. Having prominent vocals makes the extraction easier and typically results in higher-quality separation. Overall, these results demonstrate high-quality, real-world use-cases of the proposed method that were previously not possible with alternative methods.

Chapter 7

Conclusions

In this section, we give an overview of the benefits of our proposed interactive approach to source separation in Section 7.1, talk about some of the problems with our approach in Section 7.2, give ideas for future work in Section 7.3, and conclude with final remarks in Section 7.4.

7.1 The Benefits of an Interactive Approach

When we analyze our proposed interactive approach to source separation, we see there are several benefits. Firstly, when we compare our proposed approach to completely manual approaches to separation (e.g., time-frequency selection and filtering), we note that we do not require complete annotations to perform separation, thus reducing the effort required by an end-user and improving performance.

Secondly, when we compare our method to completely automatic separation approaches, we note that in general, interactive approaches will always equal or outperform their automatic counterpart. This is because, for an interactive approach, a user can simply disable the interaction if the performance is worse.

Thirdly, when we compare our method to past NMF/PLVM-based methods, we see that our proposed approach does not required isolated training data to perform separation, which is a common requirement. This makes our approach useful to a much wider array of separation tasks compared to past approaches.

Fourthly, we gain the ability to iterate and refine, which removes the burden placed on the separation algorithm to be perfect the first time–a very strong assumption that is rarely achieved by automatic methods. This places a portion of the responsibility to achieve high-quality results back on the user. For the professional, who is the level of user we are interested in, this is acceptable and in many cases advantageous.

Fifthly, when we compare our method to past NMF/PLVM-based methods and other model-based methods, we see that our approach is more general and can work on a wider array of mixture sounds. This is because we allow a user to define the concept of what a source is (e.g., source = drums or source = drums + bass) simply by annotating the correct regions of the mixture spectrogram, and do not have to pre-define individual models, which is common for speech denoising and pitched-based separation.

Sixthly, our interactive feedback-loop helps users to both learn how to interpret spectrogram displays and understand how the separation algorithm reacts to their painting annotations. Given that sound is a perceptual domain and our method of interaction is indirect (i.e., painting on visualizations of sound), this is essential. In a sense, human users must also learn and adjust their behavior alongside the separation algorithm to accomplish something neither could do independently.

Lastly, we are able to achieve state-of-the-art separation results as a result of fast, interactive user-feedback. One reason for this is that within our separation algorithm we allow a user to constantly evaluate the separation quality. The user's approval or disapproval is then used to update our optimization objective, essentially indirectly incorporating a perceptual model into our separation algorithm.

7.2 The Problems of an Interactive Approach

Given all the benefits of our proposed approach, there are also several significant drawbacks. The single biggest drawback of our proposed approach is that it ultimately requires a user. Without a user, we not only achieve zero separation, but we also have no idea what we are trying to separate because there are no built-in assumptions of sound source.

Secondly, in addition to requiring a user, there is also a significant learning curve to our software system. While our user study did show hope that novice users can use our method

with minimal training, there is nonetheless a learning curve.

Thirdly, we note that even with expert user guidance, there are still no guarantees of high-quality separation results. Mixture sounds such as densely orchestrated music or noisy and/or compressed synthesizer music are just more difficult to separate than others.

Lastly, the overall computation time to perform a complete separation can be slow compared to past works, even though a single complete iteration of our parameter estimation algorithm is fast. This is because we allow users to iterate until satisfied. As a result, for complicated mixture sounds, we found it was common to perform separation with our algorithm hundreds or even thousands of times (in an interactive fashion) to achieve a final, high-quality result.

7.3 Future Work

In this section, we outline several directions for future research, including extensions to the proposed interaction paradigm, separation algorithm, and software system. Interesting extensions include: smart selection tools, multi-channel user interactions, a faster algorithm, strategies on separating long duration recordings, data-driven annotations, alternative linear algebra-based or probabilistic models, and ideas on how to improve the evaluation of separation algorithms.

7.3.1 Smart Selection Tools

One of the most requested and suggested extensions to the proposed approach is to add *smart* selection capabilities to complement our NMF/PLVM learning algorithm. The most useful and immediately obvious smart selection tools include a sinusoid selection tool, harmonic selection tool, and a transient selection tool. For a sinusoidal and harmonic tool, for example, we could allow a user to easily click on a small portion of a sinusoidal peak track and automatically select the entire peak track segment, along with any harmonics.

Such an extension could easily be incorporated into our approach in a very decoupled manner by performing a separate sinusoidal modeling analysis and then rendering any

selected peak track segments into our user-guided annotations matrices. For a transient selection tool, a similar sinusoidal analysis could be used to detect noise and transient regions within a recording. In both cases, this would greatly increase the speed and efficiency of the user-interaction and could significantly improve separation quality by minimizing user fatigue.

7.3.2 Faster Algorithm

The second most obvious and useful extension would be to make our separation algorithm faster. This could be done both at an implementation level and an algorithm level. At an implementation level, graphics processing units (GPUs) could be used to drastically speed up our current algorithm. The work of Battenberg and Wessel, for example, shows that custom GPU implementations of related NMF algorithms can achieve speedups of over 30x.

At an algorithm level, it could be possible to develop a faster EM algorithm using an alternative iterative update algorithm that only uses salient data points for computing partial updates [124]. Also, entirely new parameter estimation algorithms could be explored, such as the method of moments for latent variable models developed by Hsu, Anandkumar, and many others [125, 126, 127]. In this case, we would replace our EM algorithm with a method of moments algorithm.

7.3.3 Separation of Long Duration Recordings

The third most obvious and desirable extension would be to reduce the long annotation and computation time required for long duration recordings. This type of extension could include user-interface modifications and/or algorithmic modifications. In terms of user-interface modifications, an additional display screen could be used to allow a user to select a short duration audio clip from within a longer recording. The user could then switch to the normal Multi Paint View display, perform the interaction as needed, and separate only that short section. Once completed with separating the subregion, the user could adjust which subregion to separate, and proceed to separate the entire long duration recording.

In terms of algorithmic modifications, multiple NMF/PLVMs dictionaries (collection

of basis vectors) could be used to model each short subregion of a long duration recording. The individual models could then be combined in an intelligent way that jointly models the full length recording. This type of modification could be done in conjunction with the above-mentioned user-interface modification.

7.3.4 Data-Driven and Automatic Annotations

Another interesting extension is to try to eliminate or at least reduce user-interaction by automatically computing the required annotations from data via a side-chain analysis algorithm. The annotations could be computed from a query audio signal (e.g., itself provided by a user or otherwise), by spatial information encoded in a multichannel recording, by encoding auditory grouping principals, or some other type of information. We could even attempt to train a learning algorithm from past user-annotations to predict future annotations akin to the work of Lefèvre et al. [32].

In this way, we could keep our exact separation algorithm, but just replace the user. Alternatively, we could use a separate analysis algorithm to take a first pass on the annotations, and then let a user refine the outputs as before. This idea is similar to the smart selection idea discussed above.

7.3.5 Alternative Models

We could replace our relatively simple PLVM with a more intricate and detailed probabilistic model. Interesting alternative models include convolutive models [128], models with temporal dynamics [21, 93, 129], models that incorporate spatial information, and models that incorporate further assumptions such as harmonic-based separation algorithms. As long as there is a way to map the user-annotations to some form of regularization parameter or other type of constraint, user interaction should be able to help improve separation quality.

7.3.6 Multi-Channel User-Interactions

A completely unexplored area of future research is that of multi-channel user-interaction paradigms for source separation. In this case, a user could be employed to annotate some form of spatial cues or activations, in conjunction with our time-frequency annotations or not, and then all annotations could collectively be incorporated into a single separation algorithm. This would be especially useful if an alternative model was used, which used spatial information to perform separation jointly from multiple-channel recordings. If this was the case, even having a user simply annotate a static source panning position could greatly improve results.

7.3.7 Separation Evaluation by Interactive Ranking

Finally, one exciting extension of this work is to leverage it for evaluation purposes. As known, it is incredibly difficult to evaluate the quality of separation algorithms without user tests, given the perceptual nature of the problem domain. As a result, most people use automatic object evaluation measures or pseudo-subjective scores, such as the BSS-EVAL metrics or the PEASS scores. These metrics, however, are not perfect by any means and motivate alternative evaluation methods.

One approach to performing evaluation by leveraging this work could be as follows: use the developed software to perform separation on a large collection of mixture sounds. For each individual mixture separation, save the "path" of intermediate separation results. Then assume that the separation quality approximately improves between each intermediate output (i.e., the user verifies the results are getting better), so that you have a collection of separation results in order of low-quality to high-quality for each example mixture sound. Use these ranked recordings to learn an evaluation metric that correctly predicts the rank of the intermediate separation results for a large collection of sounds.

7.4 Final Remarks

We have presented a new interaction paradigm and separation algorithm for single-channel source separation. In addition, we presented an open-source, freely available, cross-platform

audio editing tool that embodies our method as well as evaluation showing the method can achieve state-of-the-art separation quality. The proposed method works by allowing a user to provide feedback into the separation process by painting on spectrogram displays of both the input and output sounds of the separation process. The painting annotations are then used to inform an NMF/PLVM-based separation system and iteratively improve separation quality. Evaluation and demonstrations were presented for a wide variety of sounds, showing promise that the proposed approach can be useful for a wide variety of real-world audio and music editing scenarios. Collectively, we hope that this demonstrates the usefulness of incorporating HCI ideas into machine learning and signal processing problems such as source separation, where the use of user-feedback can be extremely beneficial. To download the application, code, and audio/video demonstrations, please see `http://ccrma.stanford.edu/~njb/thesis`.

Appendices

Appendix A

Notation and Terminology

A.1 Variables, Symbols, and Operations

$\approx$	approximately equal to
$:=$	defined to be
$\equiv$	equivalent to
$\propto$	proportional to
∞	infinity
$\geq$	element-wise greater than or equal to for scalars, vectors, and matrices
$\leq$	element-wise less than or equal to for scalars, vectors, and matrices
$>$	element-wise greater than for scalars, vectors, and matrices
$<$	element-wise less than for scalars, vectors, and matrices
$\odot$	element-wise matrix or vector multiplication
$x \leftarrow y$	in an algorithm: assign variable x to the new value y
$\arg\max_x f(x)$	the value of x that leads to the maximum value of $f(x)$
$\arg\min_x f(x)$	the value of x that leads to the minimum value of $f(x)$
$\ln(x)$	natural logarithm, base e
$\log_{10}(x)$	logarithm, base 10
$f(x;\theta)$	a function of x that is parametrized by θ
$\sum_{i=1}^{n} a_i$	the sum of values indexed from $i = 1$ to n
$\prod_{i=1}^{n} a_i$	the product of values indexed from $i = 1$ to n
$f^+(x)$	the positive part of a function $f^+(x) = \max(f(x), 0)$
$f^-(x)$	the positive part of a function $f^-(x) = -\min(f(x), 0)$

A.2 Vectors and Matrices

$\mathrm{R}^{N\times 1}$	N-dimensional Euclidean space
$\mathrm{R}_{+}^{N\times 1}$	all non-negative reals in N-dimensional Euclidean space
Δ^n	an n-simplex or $\{(t_0,\ldots,t_n)\in \mathrm{R}^{n+1} \mid \sum_{i=0}^{n} t_i = 1, t_i \geq 0 \forall i\}$
$\mathbf{x}, \mathbf{Y}, \ldots$	boldface is used for (column) vectors and matrices
$\mathbf{I}$	square identity matrix, 1's on the diagonal, 0's elsewhere
$\mathbf{1}$	column vector of all ones
$\mathbf{0}$	column vector of all zeros
$\mathbf{x}^{\mathrm{T}}$	matrix transpose of the vector $\mathbf{x}$
$\nabla = \begin{pmatrix} \frac{d}{dx_1} \\ \vdots \\ \frac{d}{dx_N} \end{pmatrix}$	gradient operator in $\mathrm{R}^{N\times 1}$
$x \in \mathrm{R}$	real-valued scalar
$\mathbf{x} \in \mathrm{R}^{N\times 1}$	real-valued vectors with N elements. By convention all vectors are columns vectors. To denote a row vector, we write $\mathbf{x}^{\mathrm{T}}$
$\mathbf{X} \in \mathrm{R}^{M\times N}$	a real-valued matrices with M rows and N columns
x_i	ith element of a vector $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$
$\mathbf{x}_i$	ith column vector of a matrix $\mathbf{X} = \begin{bmatrix} \vert & & \vert & & \vert \\ \mathbf{x}_1 & \ldots & \mathbf{x}_N \\ \vert & & \vert & & \vert \end{bmatrix}$
$\mathbf{x}_j^{\mathrm{T}}$	jth row vector of a matrix $\mathbf{X} = \begin{bmatrix} — & \mathbf{x}_1^{\mathrm{T}} & — \\ & \vdots & \\ — & \mathbf{x}_M^{\mathrm{T}} & — \end{bmatrix}$
X_{ij}	the element at ith row and jth column of a matrix $\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \ldots & X_{1N} \\ X_{21} & X_{22} & \ldots & X_{2N} \\ \vdots & \vdots & \cdots & \vdots \\ X_{M1} & X_{M2} & \ldots & X_{MN} \end{bmatrix}$

A.3 Sets

$\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots, \mathcal{D}$	calligraphic fonts denote sets or lists
$\mathbf{x} \in \mathcal{D}$	the vector $\mathbf{x}$ is an element of the set $\mathcal{D}$
$\mathcal{A} \cup \mathcal{B}$	union of two sets $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{C} \cap \mathcal{D}$	intersection of two sets $\mathcal{C}$ and $\mathcal{D}$
$\lvert\mathcal{A}\rvert$	cardinality or number of elements in the set $\mathcal{A}$

A.4 Probability and Random Variables

Ω	sample space or the set of all outcomes of a random experiment
$\mathcal{F}$	set of events, event space, or the set whose elements are subsets of Ω
$X(w)$ or X	a random variable or a function $X : \Omega \rightarrow \mathrm{R}$
x	an outcome value of the random variable X
$\sum_x$	sum over all possible values that X can take
$\mathbf{X}$	a set of random variables
$\mathbf{X}_i$ or X_i	ith random variable in the set of random variables $\mathbf{X}$
$\mathbf{x}$	an outcome value for a set of random variables $\mathbf{X}$
$p(\cdot)$	probability distribution or function $P : \mathcal{F} \rightarrow \mathrm{R}$ that assigns probabilities to events and follows the axioms of probability
$p(X = x)$	the probability of the set of outcomes for which the random variable X takes on the value x or $P(\{w : X(w) = x\})$
$p(x)$	shorthand for $P(X = x)$ or equivalently $P(\{w : X(w) = x\})$
$p_X(x)$ or $p(x)$	probability mass function (PMF), where $p_X(x) : \Omega \rightarrow \mathrm{R}$
$p(x, y)$	joint probability between the random variable outcomes x and y
$p(x\lvert\theta)$	conditional probability of the outcome value x given θ
$Val(X)$	set of values that the random variable X can take
$\lvert Val(X)\rvert$	number of elements in $Val(X)$
x^i	ith outcome value of the random variable X

A.5 Probability Distributions

The $Multinomial(N, \boldsymbol{\pi})$ distribution is defined by the probability mass function

$$p(x_1, x_2, \ldots, x_k; n, \boldsymbol{\pi}) = \begin{cases} \frac{n!}{x_1! x_2! \ldots x_k!} \pi_1^{x_1} \pi_2^{x_2} \ldots \pi_k^{x_k} & \sum_{i=1}^{k} x_i = n \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

where n is the total number of trials of a random experiment with k possible outcomes, $x_1, x_2, \ldots, x_k$ are non-negative integers that represent the number of outcomes for each of the k categories, and $\boldsymbol{\pi} = (\pi_1, \pi_2, \cdots, \pi_k)$ denotes the probability of each of the k outcomes, where $\pi_1, \pi_2, \cdots, \pi_k \geq 0$ and $\sum_{i=1}^{k} \pi_i = 1$. A random variable X that is distributed according to a multinomial distribution is denoted $X \sim Multinomial(n, \boldsymbol{\pi})$.

The $Poisson(\mu)$ distribution is defined by the probability mass function

$$p(k; \mu) = \frac{\mu^k e^{-\mu}}{k!} \tag{A.2}$$

where e is the base of the natural logarithm, $k = 1, 2, 3, \cdots$ is a non-negative integer representing the number of occurrences of a random variable, and $\mu > 0$ is the mean of the distribution. A random variable X that is distributed according to a Poisson distribution is denoted $X \sim Poisson(\mu)$.

A.6 Acronyms

APS	Artifact-related Perceptual Score
COLA	Constant Overlap-add
DTFT	Discrete-time Fourier Transform
DFT	Discrete Fourier Transform
EM	Expectation-Maximization
FFT	Fast Fourier Transform
GPU	Graphics Processing Unit
HCI	Human-Computer Interaction
IID	Independently Identically Distributed
IML	Interactive Machine Learning
IPS	Interference-related Perceptual Score
ISR	Image-to-Spatial Distortion Ratio
ISSE	Interactive Source Separation Editor
IS	Itakura-Saito
ISTFT	Inverse Short-Time Fourier transform
JUCE	Jules' Utility Class Extensions
KL	Kullback-Leibler
ML	Maximum Likelihood
MM	Majorize-Minimization or Minorize-Maimization
NLP	Natural Language Processing
NMF	Non-Negative Matrix Factorization
OLA	Overlapp-Add
OPS	Overall Perceptual Score
PLCA	Probabilistic Latent Component Analysis
PLSA	Probabilistic Latent Semantic Analysis
PLSI	Probabilistic Latent Semantic Indexing
PLVM	Probabilistic Latent Variable Model
PR	Posterior Regularization

SAR	Source-to-Artifact Ratio
SDR	Source-to-Distortion Ratio
SiSEC	Signal Separation Evaluation Campaign
SIR	Source-to-Interference Ratio
STFT	Short-Time Fourier transform
TPS	Target-related Perceptual Score
WOLA	Weighted Overlapp-Add

Appendix B

Relationship Between Poisson and Multinomial Distributions

In this appendix, we follow the proof of Steel [95] and show that

$$p(X_1, X_2, \ldots, X_k | Y = n) = Multinomial(n, \boldsymbol{\pi}) \tag{B.1}$$

where $X_1, X_2, \ldots X_k$ independently identically distributed Poisson random variables

$$\begin{aligned} X_1 &\sim Poisson(\mu_1) \\ X_2 &\sim Poisson(\mu_2) \\ &\vdots \\ X_k &\sim Poisson(\mu_k) \end{aligned} \tag{B.2}$$

with means $\mu_1, \mu_2, \ldots, \mu_k$, $\boldsymbol{\pi} = (\frac{\mu_1}{\mu}, \frac{\mu_2}{\mu}, \ldots \frac{\mu_k}{\mu})$, and $\mu = \mu_1 + \mu_2 + \ldots \mu_k$.

Proof

Consider the joint distribution of the k independent Poisson random variables

$$\begin{aligned} p(X_1 = x_1, X_2 = x_1, \ldots, X_k = x_k) &= \frac{\mu_1^{x_1} e^{-\mu_1}}{x_1!} \cdot \frac{\mu_2^{x_2} e^{-\mu_2}}{x_2!} \cdot \ldots \cdot \frac{\mu_1^{x_k} e^{-\mu_k}}{x_k!} \\ &= \prod_{i=1}^{k} \frac{e^{-\mu_i} \mu_i^{x_i}}{x_i!} \\ &= e^{-\sum_{i=1}^{k} \mu_i} \prod_{i=1}^{k} \frac{\mu_i^{x_i}}{x_i!} \end{aligned} \tag{B.3}$$

Then consider a new random variable $Y = \sum_i^k X_i$. The probability that Y takes on the value n is,

$$\begin{aligned} p(Y = n) &= p(\sum_{i}^{k} X_i = n) \\ &= \sum_{x_1 + x_2 + \ldots + x_k = n} p(X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k) \\ &= \sum_{x_1 + x_2 + \ldots + x_k = n} p(X_1 = x_1) p(X_2 = x_2) \ldots p(X_k = x_k) \\ &= \sum_{x_1 + x_2 + \ldots + x_k = n} \frac{e^{-\mu_1} \mu_1^{x_1}}{x_1!} \cdot \frac{e^{-\mu_2} \mu_2^{x_2}}{x_2!} \cdot \ldots \cdot \frac{e^{-\mu_k} \mu_k^{x_k}}{x_k!} \\ &= e^{-\sum_{i=1}^{k} \mu_i} \sum_{x_1 + x_2 + \ldots + x_k = n} \frac{\prod_{j=1}^{k} \mu_j^{x_j}}{\prod_{i=1}^{k} x_i!} \\ &= e^{-\sum_{i=1}^{k} \mu_i} \frac{(\sum_{i=1}^{k} \mu_i)^n}{n!} \\ &= e^{-\mu} \frac{\mu^n}{n!}, \end{aligned} \tag{B.4}$$

where $\mu = \sum_i^k \mu_i$ and the second to last step leverages the multinomial theorem

$$(\mu_1 + \mu_2 + \ldots + \mu_k)^n = \sum_{x_1 + x_2 + \ldots + x_k = n} \frac{n! \prod_{j=1}^{k} \mu_i^{x_j}}{\prod_{i=1}^{k} x_i!}. \tag{B.5}$$

This implies $Y \sim Poisson(\sum_{i=1}^{k} \mu_i)$ or the sum of independent Poisson random variables is itself a Poisson distributed random variable.

Knowing this, we can then compute the conditional distribution of $X_1, X_2, \ldots X_k$ given $Y = n$ knowing $\sum_i x_i = n$

$$
\begin{aligned}
p(X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k | Y = n) = & \\
&= \frac{p(Y = n | X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k) p(X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k)}{p(Y = n)} \\
&= \frac{1 \cdot p(X_1 = x_1, X_2 = x_2, \ldots, X_k = x_k)}{p(Y = n)} \\
&= (\prod_{i=1}^{k} \frac{e^{-\mu_i} \mu_i^{x_i}}{x_i!}) / (e^{-\mu} \frac{\mu^n}{n!}) \\
&= \frac{n!}{\prod_{i=1}^{k} x_i!} \mu^n \prod_{i=1}^{k} \mu_i^{x_i} \\
&= \frac{n!}{\prod_{i=1}^{k} x_i!} \mu^{\sum x_i} \prod_{i=1}^{k} \mu_i^{x_i} \\
&= \frac{n!}{\prod_{i=1}^{k} x_i!} \prod_{i=1}^{k} (\frac{\mu_i}{\mu})^{x_i},
\end{aligned}
$$

The final result is in the form of a multinomial distribution with parameters n and $\boldsymbol{\pi} = (\frac{\mu_1}{\mu}, \frac{\mu_2}{\mu}, \ldots \frac{\mu_k}{\mu})$, where $\mu = \mu_1 + \mu_2 + \ldots \mu_k$ as desired.

Bibliography

[1] P. Griffiths, *Modern Music: A Concise History*. New York, NY, USA: Thames and Hudson Inc., 1994.

[2] S. Sonvilla-Weiss, *Mashup Cultures*. New York, NY, USA: Springer-Verlag Inc., 2010.

[3] J. Benesty, J. Chen, and Y. Huang, *Microphone Array Signal Processing*. Springer, 2008.

[4] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1985.

[5] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks*, vol. 13, no. 4-5, pp. 411–430, May 2000.

[6] D. Wang and G. J. Brown, *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. Wiley-IEEE Press, 2006.

[7] A. L. Wang, "Instantaneous and frequency-warped signal processing techniques for auditory source separation," Ph.D. dissertation, Stanford University, Stanford, CA, USA, 1994.

[8] J. O. Smith III, *Spectral Audio Signal Processing*. http://ccrma.stanford.edu/-˜jos/sasp, accessed 2013, online book.

[9] S. F. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Transactions on Acoustics, Speech and Signal Processing (TASSP)*, vol. 27, no. 2, pp. 113–120, 1979.

[10] M. Berouti, R. Schwartz, and J. Makhoul, "Enhancement of speech corrupted by acoustic noise," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, 1979, pp. 208–211.

[11] J. S. Lim and A. V. Oppenheim, "Enhancement and bandwidth compression of noisy speech," *Proceedings of the IEEE*, vol. 67, no. 12, pp. 1586–1604, 1979.

[12] Y. Ephraim and D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," *IEEE Transactions on Acoustics, Speech and Signal Processing (TASSP)*, vol. 32, no. 6, pp. 1109–1121, 1984.

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

[14] 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 (WASPAA)*, Oct. 2003, pp. 177 – 180.

[15] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 15, no. 3, pp. 1066–1074, Mar. 2007.

[16] C. Févotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis," *Neural Computation*, vol. 21, no. 3, pp. 793–830, Mar. 2009.

[17] B. Raj and P. Smaragdis, "Latent variable decomposition of spectrograms for single channel speaker separation," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct. 2005, pp. 17 – 20.

[18] P. Smaragdis, B. Raj, and M. Shashanka, "A probabilistic latent variable model for acoustic modeling," in *Advances in Neural Information Processing Systems (NIPS), Workshop on Advances in Modeling for Acoustic Processing*, 2006.

[19] ——, "Supervised and semi-supervised separation of sounds from single-channel mixtures," in *International Conference on Independent Component Analysis and Signal Separation (ICASS)*. Springer-Verlag, 2007, pp. 414–421.

[20] A. Ozerov and C. Févotte, "Multichannel nonnegative matrix factorization in convolutive mixtures. with application to blind audio source separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2009, pp. 3137–3140.

[21] G. J. Mysore, P. Smaragdis, and B. Raj, "Non-negative hidden markov modeling of audio with application to source separation," in *International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, 09/2010 2010. [Online]. Available: https://ccrma.stanford.edu/~gautham/Site/NFHMM.html

[22] C. Févotte, J. L. Roux, and J. R. Hershey, "Non-negative dynamical system with application to speech and audio," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2013.

[23] J. F. Woodruff, B. Pardo, and R. B. Dannenberg, "Remixing stereo music with score-informed source separation." in *International Society for Music Information Retrieval (ISMIR)*, 2006, pp. 314–319.

[24] J. Ganseman, G. J. Mysore, J. S. Abel, and P. Scheunders, "Source separation by score synthesis," *International Computer Music Conference (ICMC)*, pp. 462–465, 2010.

[25] R. Hennequin, B. David, and R. Badeau, "Score informed audio source separation using a parametric model of non-negative spectrogram," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 45–48.

[26] Z. Duan and B. Pardo, "Soundprism: An online system for score-informed source separation of music audio," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1205–1215, 2011.

[27] S. Ewert and M. Muller, "Using score-informed constraints for nmf-based source separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 129–132.

[28] P. Smaragdis, "User guided audio selection from complex sound mixtures," in *Symposium on User Interface Software and Technology (UIST)*. ACM, 2009, pp. 89–92.

[29] P. Smaragdis and G. J. Mysore, "Separation by "humming": User-guided sound extraction from monophonic mixtures," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2009, pp. 69–72.

[30] A. Ozerov, C. Févotte, R. Blouet, and J.-L. Durrieu, "Multichannel nonnegative tensor factorization with structured constraints for user-guided audio source separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 257 –260.

[31] J.-L. Durrieu and J.-P. Thiran, "Musical audio source separation based on user-selected f0 track," in *International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, 2012, pp. 438–445.

[32] A. Lefèvre, F. Bach, and C. Févotte, "Semi-supervised nmf with time-frequency annotations for single-channel source separation," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2012.

[33] A. Ozerov, N. Q. Duong, and L. Chevallier, "Weighted nonnegative tensor factorization with application to user-guided audio source separation," in *Submitted to IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2013)*, 2013.

[34] J. Graça, K. Ganchev, and B. Taskar, "Expectation maximization and posterior constraints," in *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[35] K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar, "Posterior regularization for structured latent variable models," *Journal of Machine Learning Research (JMLR)*, vol. 11, pp. 2001–2049, Aug. 2010.

[36] D. Fitzgerald, "User assisted separation using tensor factorisations," in *European Signal Processing Conference (EUSIPCO)*, 2012, pp. 2412–2416.

[37] B. Wang and M. D. Plumbley, "Investigating single-channel audio source separation methods based on non-negative matrix factorization," in *ICA Research Network International Workshop*, 2006.

[38] A. Ozerov, E. Vincent, and F. Bimbot, "A general flexible framework for the handling of prior information in audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 20, no. 4, pp. 1118–1133, 2012.

[39] H. Kirchhoff, S. Dixon, and A. Klapuri, "Missing template estimation for user-assisted music transcription," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 26–30.

[40] N. Bogaards, A. Röbel, and X. Rodet, "Sound analysis and processing with audiosculpt 2," in *International Computer Music Conference (ICMC)*, 2004.

[41] N. Bogaards, "Analysis-assisted sound processing with audiosculpt," in *Digital Audio Effects Conference (DAFX)*, 9 2005.

[42] C. G. v. d. Boogaart and R. Lienhart, "Audio brush: a tool for computer-assisted smart audio editing," in *Workshop on Audio and Music Computing Multimedia*, ser. AMCMM '06. ACM, 2006, pp. 115–124.

[43] M. Klingbeil, "Software for spectral analysis, editing, and synthesis," in *International Computer Music Conference (ICMC)*, 2005, pp. 107–110.

[44] Celemony, "Melodyne Editor," http://www.celemony.com/cms/index.php, 2013.

[45] Sony, "SpectralLayers," http://www.sonycreativesoftware.com/spectralayerspro, 2013.

[46] Adobe, "Adobe Audition," http://www.adobe.com/products/audition.html, 2013.

[47] Izotope, "Rx," http://www.izotope.com/products/audio/rx/, 2013.

[48] J. A. Fails and D. R. Olsen, Jr., "Interactive machine learning," in *International Conference on Intelligent User Interfaces (IUI)*. New York, NY, USA: ACM, 2003, pp. 39–45.

[49] D. Cohn, R. Caruana, and A. Mccallum, "Semi-supervised clustering with user feedback," in *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, S. Basu, I. Davidson, and K. Wagstaff, Eds. Chapman & Hall/CRC, 2008.

[50] S. Stumpf, V. Rajaram, L. Li, M. Burnett, T. Dietterich, E. Sullivan, R. Drummond, and J. Herlocker, "Toward harnessing user feedback for machine learning," in *International Conference on Intelligent User Interfaces (IUI)*. ACM, 2007, pp. 82–91.

[51] J. Talbot, B. Lee, A. Kapoor, and D. Tan, "Ensemblematrix: Interactive visualization to support machine learning with multiple classifiers," in *Human Factors in Computing Systems (CHI)*, 2009.

[52] J. Fogarty, D. Tan, A. Kapoor, and S. Winder, "Cueflik: interactive concept learning in image search," in *Human Factors in Computing Systems (CHI)*. ACM, 2008, pp. 29–38.

[53] R. Fiebrink, "Real-time human interaction with supervised learning algorithms for music composition and performance," Ph.D. dissertation, Princeton University, Princeton, NJ, USA, January 2011.

[54] B. Settles, "Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances," in *Empirical Methods in Natural Language Processing Conference (EMNLP)*. Association for Computational Linguistics, July 2011, pp. 1467–1478.

[55] S. Amershi, B. Lee, A. Kapoor, R. Mahajan, and B. Christian, "CueT: Human-guided fast and accurate network alarm triage," in *Human Factors in Computing Systems (CHI)*. ACM, 2011, pp. 157–166.

[56] N. J. Bryan and G. J. Mysore, "Interactive user-feedback for sound source separation," in *International Conference on Intelligent User Interfaces (IUI), Workshop on Interactive Machine Learning*, 2013, p. 2.

[57] ——, "An efficient posterior regularized latent variable model for interactive sound source separation," in *International Conference on Machine Learning (ICML)*, June 2013.

[58] ——, "Interactive refinement of supervised and semi-supervised sound source separation estimates," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2013.

[59] N. J. Bryan, G. J. Mysore, and G. Wang, "Source separation of polyphonic music with interactive user-feedback on a piano roll display," in *The International Society for Music Information Retrieval (ISMIR)*, 2013, p. 6.

[60] N. J. Bryan and G. J. Mysore, "Professionally produced music recordings (isse submission)," in *Signal Separation Evaluation Campaign (SiSEC)*, 2013. [Online]. Available: http://sisec.wiki.irisa.fr/

[61] N. J. Bryan, G. J. Mysore, and G. Wang, "ISSE: An interactive source separation editor (a demonstration)," in *Neural Information Processing (NIPS), Workshop on Machine Learning Open Source Software*, 2013. [Online]. Available: http://mloss.org/workshop/nips13/

[62] N. J. Bryan, G. J. Mysore, and G. Wang, "ISSE: An interactive source separation editor," in *Human Factors in Computing Systems (CHI)*, April 2014.

[63] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[64] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, 2010.

[65] I. Jolliffe, *Principal Component Analysis*. Wiley Online Library, 2005.

[66] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[67] M. A. Casey and A. Westner, "Separation of mixed audio sources by independent subspace analysis," in *International Computer Music Conference (ICMC)*, 2000, pp. 154–161.

[68] C. Uhle, C. Dittmar, and T. Sporer, "Extraction of drum tracks from polyphonic music using independent subspace analysis," in *International Symposium on Independent Component Analysis and Blind Signal Separation*, 2003, pp. 843–847.

[69] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent Component Analysis and Applications*. Access Online via Elsevier, 2010.

[70] S. Makino, T.-W. Lee, and H. Sawada, *Blind Speech Separation*. Springer, 2007.

[71] W. Wang, *Machine Audition: Principles, Algorithms and Systems*. IGI Publishing, 2010.

[72] J. O. Smith III, *Mathematics of the Discrete Fourier Transform (DFT)*. http://www.w3k.org/books/: W3K Publishing, 2007.

[73] ——, *Introduction to Digital Filters with Audio Applications*. http://www.w3k.org/books/: W3K Publishing, 2007.

[74] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[75] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*. Springer, Jul. 2003.

[76] D. Kollar and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

[77] C. M. Bishop, *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[78] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[79] R. N. Bracewell and R. Bracewell, *The Fourier Transform and its Applications*. McGraw-Hill New York, 1986, vol. 31999.

[80] J. B. Allen and L. R. Rabiner, "A unified approach to short-time fourier analysis and synthesis," *Proceedings of the IEEE*, vol. 65, no. 11, pp. 1558–1564, 1977.

[81] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.

[82] D. R. Hunter and K. Lange, "A tutorial on MM algorithms," *The American Statistician*, vol. 58, no. 1, 2004.

[83] S. Sra and I. S. Dhillon, "Generalized nonnegative matrix approximations with bregman divergences," in *Advances in Neural Information Processing Systems (NIPS)*, 2005, pp. 283–290.

[84] A. Cichocki, R. Zdunek, and S. Amari, "Csiszárs divergences for non-negative matrix factorization: Family of new algorithms," in *Independent Component Analysis and Blind Signal Separation (ICA)*. Springer, 2006, pp. 32–39.

[85] C. Févotte and J. Idier, "Algorithms for nonnegative matrix factorization with the β-divergence," *Neural Computation*, vol. 23, no. 9, pp. 2421–2456, 2011.

[86] Z. Yang and E. Oja, "Unified development of multiplicative algorithms for linear and quadratic nonnegative matrix factorization," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1878 –1891, Dec. 2011.

[87] T. Virtanen, A. Cemgil, and S. Godsill, "Bayesian extensions to non-negative matrix factorisation for audio signal modelling," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 1825–1828.

[88] M. Hoffman, "Poisson-uniform nonnegative matrix factorization," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5361–5364.

[89] P. Smaragdis, "Probabilistic decompositions of spectra for sound separation," in *Blind Speech Separation*, ser. Signals and Communication Technology, S. Makino, H. Sawada, and T.-W. Lee, Eds. Springer Netherlands, 2007, pp. 365–386.

[90] P. Smaragdis and B. Raj, "Shift-invariant probabilistic latent component analysis," *MERL Tech Report*, 2007.

[91] M. Shashanka, B. Raj, and P. Smaragdis, "Probabilistic latent variable models as nonnegative factorizations," *Computational Intelligence and Neuroscience*, 2008.

[92] T. Hofmann, "Probabilistic latent semantic indexing," in *International Conference on Research and Development in Information Retrieval*. ACM, 1999, pp. 50–57.

[93] G. J. Mysore, "A non-negative framework for joint modeling of spectral structure and temporal dynamics in sound mixtures," Ph.D. dissertation, Stanford University, Stanford, CA, USA, 2010.

[94] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[95] R. G. D. Steel, "Biometrics unit technical reports: Relation between poisson and multinomial distributions," Cornell University, Ithaca, New York, Tech. Rep. BU-39-M, April 1953.

[96] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*. Prentice Hall New Jersey, 2000, vol. 1.

[97] L. Benaroya, L. M. Donagh, F. Bimbot, and R. Gribonval, "Non negative sparse representation for wiener based source separation with a single sensor," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 6, April 2003.

[98] D. Fitzgerald and R. Jaiswal, "On the use of masking filters in sound source separation," in *International Conference on Digital Audio Effects (DAFx)*, 2012.

[99] J. Le Roux, E. Vincent, Y. Mizuno, H. Kameoka, N. Ono, and S. Sagayama, "Consistent Wiener filtering: Generalized time-frequency masking respecting spectrogram consistency," in *International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, Sep. 2010, pp. 89–96.

[100] J. Le Roux and E. Vincent, "Consistent wiener filtering for audio source separation," *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 217–220, Mar. 2013.

[101] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 309–314, Aug. 2004.

[102] T. Fowler, "Giraffe (modified)," http://www.flickr.com/photos/j33pman/7701103436/sizes/l/, 2012, Attribution-NonCommercial-ShareAlike 2.0 Generic License.

[103] J. Graça, K. Ganchev, B. Taskar, and F. C. N. Pereira, "Posterior vs parameter sparsity in latent variable models," in *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 664–672.

[104] N. J. Bryan, G. J. Mysore, and G. Wang, "ISSE: An interactive source separation editor," http://isse.sourceforge.net, 2013.

[105] Free Software Foundation, "GPL Version 3," http://gplv3.fsf.org/, 2007.

[106] J. Storer, *JUCE (Jules' Utility Class Extensions)*. http://rawmaterialsoftware.com/juce.php, 2013, online.

[107] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[108] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005, special issue on "Program Generation, Optimization, and Platform Adaptation".

[109] E. Battenberg and D. Wessel, "Accelerating non-negative matrix factorization for audio source separation on multi-core and many-core architectures," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2009, pp. 501–506.

[110] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 14, no. 4, pp. 1462 –1469, July 2006.

[111] V. Emiya, E. Vincent, N. Harlander, and V. Hohmann, "Subjective and objective quality assessment of audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 19, no. 7, pp. 2046–2057, 2011.

[112] S. Committee, "Professionally produced music recordings," in *Signal Separation Evaluation Campaign (SiSEC)*, 2011, `http`:http://sisec.wiki.irisa.fr/tiki-index.php`//sisec.wiki.irisa.fr/tiki-index.php`.

[113] E. Estefanía Cano, C. Dittmar, and G. Schuller, "Efficient implementation of a system for solo and accompaniment separation in polyphonic music," in *European Signal Processing Conference (EUSIPCO)*, 2012, pp. 285–289.

[114] E. Cano, G. Schuller, and C. Dittmar, "Pitch-informed solo and accompaniment separation: Towards its use in music education applications," *(Submitted) Special Issue on Informed Acoustic Source Separation EURASIP Journal on Advances in Signal Processing*, pp. 285–289, 2012.

[115] S. Gorlow and S. Marchand, "Informed audio source separation using linearly constrained spatial filters," *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 21, no. 1, pp. 3–13, 2013.

[116] C. Cao, M. Li, J. Liu, and Y. Yan, "Singing melody extraction in polyphonic music by harmonic tracking," in *International Society for Music Information Retrieval Conference (ISMIR)*, September 23-27 2007, pp. 373–374.

[117] R. Marxer and J. Janer, "Low-latency bass separation using harmonic-percussion decomposition," in *Digital Audio Effects Conference (DAFX)*, 2013.

[118] J. Janer and R. Marxer, "Separation of unvoiced fricatives in singing voice mixtures with semi-supervised nmf," in *Digital Audio Effects Conference (DAFX)*, 2013.

[119] Z. Rafii and B. Pardo, "Repeating pattern extraction technique (repet): A simple method for music/voice separation," *IEEE Transactions on Audio, Speech & Language Processing (TASLP)*, vol. 21, no. 1, pp. 71–82, 2013.

[120] ——, "Music/voice separation using the similarity matrix," in *International Society for Music Information Retrieval Conference (ISMIR)*, October 8-12 2012.

[121] S. Arberet, R. Gribonval, and F. Bimbot, "A robust method to count and locate audio sources in a multichannel underdetermined mixture," *IEEE Transactions on Signal Processing (TSP)*, vol. 58, no. 1, pp. 121–133, 2010.

[122] J.-L. Durrieu, B. David, and G. Richard, "A musically motivated mid-level representation for pitch estimation and musical audio source separation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1180–1191, 2011.

[123] A. Liutkus, Z. Rafii, R. Badeau, B. Pardo, and G. Richard, "Adaptive filtering for music/voice separation exploiting the repeating musical structure," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 53–56.

[124] R. M. Neal and G. E. Hinton, "A view of the em algorithm that justifies incremental, sparse, and other variants," in *Learning in graphical models*. Springer, 1998, pp. 355–368.

[125] D. Hsu, S. M. Kakade, and T. Zhang, "A spectral algorithm for learning hidden markov models," *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1460–1480, 2012.

[126] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *arXiv preprint arXiv:1210.7559*, 2012.

[127] A. Anandkumar, D. Hsu, and S. M. Kakade, "A method of moments for mixture models and hidden markov models," *arXiv preprint arXiv:1203.0683*, 2012.

[128] P. Smaragdis, "Convolutive speech bases and their application to supervised speech separation," *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 15, no. 1, pp. 1–12, 2007.

[129] C. Févotte, J. Le Roux, and J. R. Hershey, "Non-negative dynamical system with application to speech and audio," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2013.