AUDIO COMPUTING IN THE WILD:
FRAMEWORKS FOR BIG DATA AND SMALL COMPUTERS

BY

MINJE KIM

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Professor Paris Smaragdis, Chair
Professor Rob A. Rutenbar
Professor Mark A. Hasegawa-Johnson
Dr. Gautham J. Mysore, Adobe Research

Abstract

This dissertation presents some machine learning algorithms that are designed to process as much data as needed while spending the least possible amount of resources, such as time, energy, and memory. Examples of those applications, but not limited to, can be a large-scale multimedia information retrieval system where both queries and the items in the database are noisy signals; collaborative audio enhancement from hundreds of user-created clips of a music concert; an event detection system running in a small device that has to process various sensor signals in real time; a lightweight custom chipset for speech enhancement on hand-held devices; instant music analysis engine running on smartphone apps. In all those applications, efficient machine learning algorithms are supposed to achieve not only a good performance, but also a great resource-efficiency.

We start from some efficient dictionary-based single-channel source separation algorithms. We can train this kind of source-specific dictionaries by using some matrix factorization or topic modeling, whose elements form a representative set of spectra for the particular source. During the test time, the system estimates the contribution of the participating dictionary items for an unknown mixture spectrum. In this way we can estimate the activation of each source separately, and then recover the source of interest by using that particular source's reconstruction. There are some efficiency issues during this procedure. First off, searching for the optimal dictionary size is time consuming. Although for some very common types of sources, e.g. English speech, we know the optimal rank of the model by trial and error, it is hard to know in advance as to what is the optimal number of dictionary elements for the unknown sources, which are usually modeled during the test time in the semi-supervised separation scenarios. On top of that, when it comes to the non-stationary unknown sources, we had better maintain a dictionary that adapts its size and contents to the change of the source's nature. In this online semi-supervised separation scenario, a mechanism that can efficiently learn the optimal rank is helpful. To this end, a deflation method is proposed for modeling this unknown source with a nonnegative dictionary whose size is optimal. Since it has to be done during the test time, the deflation method that incrementally adds up new dictionary items shows better efficiency than a corresponding naïve approach where we simply try a bunch of different models.

We have another efficiency issue when we are to use a large dictionary for better separation. It has been known that considering the manifold of the training data can help enhance the performance for the separation. This is because of the symptom that the usual manifold-ignorant convex combination models, such as from low-rank matrix decomposition or topic modeling, tend to result in ambiguous regions in the source-specific subspace defined by the dictionary items as the bases. For example, in those ambiguous regions, the original data samples cannot reside. Although some source separation techniques that respect data manifold could increase the performance, they call for more memory and computational resources due to the fact that the models call for larger dictionaries and involve sparse coding during the test time. This limitation led the development of hashing-based encoding of the audio spectra, so that some computationally heavy routines, such as nearest neighbor searches for sparse coding, can be performed in a cheaper bit-wise fashion.

Matching audio signals can be challenging as well, especially if the signals are noisy and the matching task involves a big amount of signals. If it is an information retrieval application, for example, the bigger size of the data leads to a longer response time. On top of that, if the signals are defective, we have to perform the enhancement or separation job in the first place before matching, or we might need a matching mechanism that is robust to all those different kinds of artifacts. Likewise, the noisy nature of signals can add an additional complexity to the system. In this dissertation we will also see some compact integer (and eventually binary) representations for those matching systems. One of the possible compact representations would be a hashing-based matching method, where we can employ a particular kind of hash functions to preserve the similarity among original signals in the hash code domain. We will see that a variant of Winner Take All hashing can provide Hamming distance from noise-robust binary features, and that matching using the hash codes works well for some keyword spotting tasks. From the fact that some landmark hashes (e.g. local maxima from non-maximum suppression on the magnitudes of a mel-scaled spectrogram) can also robustly represent the time-frequency domain signal efficiently, a matrix decomposition algorithm is also proposed to take those irregular sparse matrices as input. Based on the assumption that the number of landmarks is a lot smaller than the number of all the time-frequency coefficients, we can think of this matching algorithm efficient if it operates entirely on the landmark representation. On the contrary to the usual landmark matching schemes, where matching is defined rigorously, we see the audio matching problem as soft matching where we find a similar constellation of landmarks to the query. In order to perform this soft matching job, the landmark positions are smoothed by a fixed-width Gaussian caps, with which the matching job is reduced down to calculating the amount of overlaps in-between those Gaussians. The Gaussian-based density approximation is also useful when we perform decomposi-

tion on this landmark representation, because otherwise the landmarks are usually too sparse to perform an ordinary matrix factorization algorithm, which are originally for a dense input matrix. We also expand this concept to the matrix deconvolution problem as well, where we see the input landmark representation of a source as a two-dimensional convolution between a source pattern and its corresponding sparse activations. If there are more than one source, as a noisy signal, we can think of this problem as factor deconvolution where the mixture is the combination of all the source-specific convolutions.

The dissertation also covers Collaborative Audio Enhancement (CAE) algorithms that aim to recover the dominant source at a sound scene (e.g. music signals of a concert rather than the noise from the crowd) from multiple low-quality recordings (e.g. Youtube video clips uploaded by the audience). CAE can be seen as crowdsourcing a recording job, which needs a substantial amount of denoising effort afterward, because the user-created recordings might have been contaminated with various artifacts. In the sense that the recordings are from not-synchronized heterogenous sensors, we can also think of CAE as big ad-hoc sensor array processing. In CAE, each recording is assumed to be uniquely corrupted by a specific frequency response of the microphone, an aggressive audio coding algorithm, interference, band-pass filtering, clipping, etc. To consolidate all these recordings and come up with an enhanced audio,Probabilistic Latent Component Sharing (PLCS) has been proposed as a method of simultaneous probabilistic topic modeling on synchronized input signals. In PLCS, some of the parameters are fixed to be same during and after the learning process to capture common audio content, while the rest of the parameters are for the unwanted recording-specific interference and artifacts. We can speed up PLCS by incorporating a hashing-based nearest neighbor search so that at every EM iteration PLCS can be applied only to a small number of recordings that are closest to the current source estimation. Experiments on a small simulated CAE setup shows that the proposed PLCS can improve the sound quality from variously contaminated recordings. The nearest neighbor search technique during PLCS provides sensible speed-up at larger scaled experiments (up to 1000 recordings).

Finally, to describe an extremely optimized deep learning deployment system, Bitwise Neural Networks (BNN) will be also discussed. In the proposed BNN, all the input, hidden, and output nodes are binaries (+1 and -1), and so are all the weights and bias. Consequently, the operations on them during the test time are defined with Boolean algebra, too. BNNs are spatially and computationally efficient in implementations, since (a) we represent a real-valued sample or parameter with a bit (b) the multiplication and addition correspond to bitwise XNOR and bit-counting, respectively. Therefore, BNNs can be used to implement a deep learning system in a resource-constrained environment, so that we can deploy a deep learning system on small devices without using up the power, memory, CPU clocks, etc. The training procedure

for BNNs is based on a straightforward extension of backpropagation, which is characterized by the use of the quantization noise injection scheme, and the initialization strategy that learns a weight-compressed real-valued network only for the initialization purpose. Some preliminary results on the MNIST dataset and speech denoising demonstrate that a straightforward extension of backpropagation can successfully train BNNs whose performance is comparable while necessitating vastly fewer computational resources.

To my wife, my parents,

and Piglet, in loving memory.

Acknowledgments

The past five academic years for my PhD study has been the most fun and fruitful period of my life. As I finish up my study by now, I can easily admit that all my achievements greatly rely on the kind helps and warm regards from my advisors, mentors, teachers, colleagues, friends, and family.

First of all, I would like to wholeheartedly thank my advisor, Prof. Paris Smaragdis. His insight and open-minded philosophy in academic guiding led me to independently exploring all the exciting research projects, while eventually I could focus more on the influential ones that he "nudged" me to work on. I know he does not like this kind of flattering words, but I literally felt like "jamming with a rock star" when I worked with him for the past five years. I could not have made it without his guidance.

I also thank all my dissertation committee members. First, I thank Prof. Rob Rutenbar for all his support which started as soon as I first entered the program. From the research projects he led, the course he taught, and all the experienced guidance for my job searches, I learned a lot from him. Especially, the big theme of this dissertation, came from the inspiration I got from his career. Second, I also appreciate Dr. Gautham Mysore for his kind mentoring throughout my PhD study and my internships in Adobe Research. His detailed comments always strengthened my internship projects, which have been basically a very exciting invention procedure. I could fully focus on those fun parts of the project because he let me do that. Finally, I also got a lot of thankful helps from Prof. Mark Hasegawa-Johnson, who ignited my deep learning research through his insightful teaching. I also finally mastered some signal processing topics from his course, too. They all were great references during my job search as well.

I would also like to thank my lab mates, Johannes Traa, Cem Sübakan, Nasser Mohammadiha, Ramin Anushiravani, Shrikant Venkataramani, and Kang Kang. I was lucky to have all those highly motivated and hard-working office mates around me. The four-time internships at the Creative Technologies Lab in Adobe Research gave me the chances to meet great mentors, such as Matthew Hoffman and Peter Merrill, and so many wonderful fellow interns, too.

Here in Illinois, I met some good colleagues. The co-work with Po-Sen Huang has opened a big research direction for me, and was very fun. The co-work and discussion with Glenn Ko, Wooil Kim, and Jungwook

Choi helped me understand the real challenges in the more hardware-friendly implementations. I also thank all my Korean friends in the Department of Computer Science. In addition to that, I appreciate every student who asked me questions. As their teaching assistant, I learned a lot by having to answer those questions. Finally, I also thank my undergraduate mentees, Igor Fedorov, Vinay Maddali, and Aswin Sivaraman, for their great projects that I enjoyed working on, too.

I relied on the cheers from my friends back in Korea. Being an international student is a lonely job basically, but I was not lonely thanks to all their support and "comments". It was lucky for me to meet those lifetime friends in T.H.i.S, IMLAB in POSTECH, the class of 2004, and ETRI. I especially thank Jonguk Kim and Seungkwon Beack for their outstandingly warm regards. On top of that, I would like to express special thanks to my previous advisors, Prof. Seungjin Choi and Prof. Kyubum Wee, for their prolonged advice.

I thank my family and in-laws in Korea. They formed such a nice team back there in our hometowns. Without their support, my adventure must have been rootless.

Finally, my deepest gratitude goes to my beloved wife, Kahyun Choi, for her tremendous support during the past five years of marriage and another eleven years of relationship before that. Her motivation, encouragement, and affection have been the sources of my everyday life. You know what? I have just followed her lead.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AIR Audio Information Retrieval

ASR Automatic Speech Recognition

AUC Area Under Curve

BNN Bitwise Neural Networks

BSS Blind Source Separation

CAE Collaborative Audio Enhancement

CCNMF Convolutive Common Nonnegative Matrix Factorization

DNN Deep Neural Networks

EM Expectation-Maximization

FFT Fast Fourier Transform

FLOP FLoating-point OPeration

GMM Gaussian Mixture Model

HMM Hidden Markov Model

IBM Ideal Binary Mask

ICA Independent Component Analysis

IR Information Retrieval

IRM Ideal Binary Mask

KNN *K*-Nearest Neighbor

LDA Latent Dirichlet Allocation

LSH Locality Sensitive Hashing

MAP Maximum A Posteriori

MFCC Mel-Frequency Cepstrum Coefficient

MIR Music Information Retrieval

MLD Mixture of Local Dictionaries

MPS Manifold Preserving Separation without hashing

MPS-WTA Manifold Preserving Separation with WTA hashing

NFD Nonnegative Factor Deconvolution

NMF Nonnegative Matrix Factorization

NMPCF Nonnegative Matrix Partial Co-Factorization

PLCA Probabilistic Latent Component Analysis

PLCS Probabilistic Latent Component Sharing

PLSI Probabilistic Latent Semantic Indexing

QbSH Query by Singing/Humming

ROC Receiver Operating Characteristic

SAR Signal-to-Artifact Ratio

SDR Signal-to-Distortion Ratio

SGD Stochastic Gradient Descent

SIR Signal-to-Interference Ratio

SNR Signal-to-Noise Ratio

STFT Short-Time Fourier Transform

SVD Singular Value Decomposition

SVM Support Vector Machine

USM Universal Speech Model

WTA Winner Take All

Chapter 1

Introduction:

Audio Computing in the Wild

1.1 Conventional Challenges in Audio Processing

There are certain aspects of real-world audio signals that can ruin the performance of an application that uses those signals, although it is also true that they sometimes even enrich people's listening experiences. We humans are surprisingly good at focusing on the desired aspect of the audio signal while suppressing the other parts, and that is why we can enjoy music recordings in a highly reverberant cathedral; a conversation with an intimate at a bar even with upbeat music in the background (perhaps not too loud). We do not recognize our ability to do this kind of job since what happens in our brain is flawless most of the time, but those unwanted parts of sound challenge a computer algorithm who tries to mimic human behaviors.

First, the additive nature of audio signals often makes audio processing tasks more difficult. When we observe a real-world audio signal, it is typically a mixture of multiple sources. On the contrary to the visual scene where an interfering object occludes the background object, in the mixed audio signal we hear all the sources at the same time. If we would like to develop an Automatic Speech Recognition (ASR) system, we want it to be robust to all the different kinds of artifacts added to the speech, including some interfering noise. For the ones who seek only the main melody of polyphonic music, all the other concurrent melodies will play as interferences, too. We can also think of a detection system that listens to sound events so that it can determine what is going on around it [1], where a few different events can occur at the same time. Likewise, the way audio sources are mixed puzzles the pattern recognition problems involving audio signals as input.

In this sense, source separation can serve as a basic functionality of most audio processing systems. For example, if we know more about how the mixture is composed of, we can do the scene recognition better [2]. ASR can obviously benefit from cleaner speech signals if speech denoising as preprocessing is done effectively [3] or the ASR model is aware of the noisiness of the mixture input in the first place [4]. Source separation is also useful for some Music Information Retrieval (MIR) tasks, for example, music transcription [5], singing voice separation for main melody extraction [6], drum source separation for beat tracking [7],

baseline and drum separation for mood classification [8], etc, where a task (e.g. main melody extraction) is more relevant to a particular source (e.g. singing voice) than the others (e.g. drums). Therefore, we can expect that a quality source separation system can be beneficial for its subsequent task.

On top of the unwanted sources mixed in the observation, there are other types of artifacts in an audio signal, too. For example, reverberation can be seen as a mixture of delayed versions of the same source. Depending on the amount of reverberation, e.g. how long the reverberation filter is, it can cause severe problems in recognition jobs, calling for a dereverberation procedure[9]. A recording process can also suffer from a lack of sampling rates, e.g. due to an aggressive compression, which can eventually cause band-limited audio signals. Bandwidth expansion is a demanding job for this kind of situation [10, 11, 12]. If the microphone is too close to the source or the source is too loud in general, we also observe a symptom called *clipping*, where some parts of waveforms above a certain value are flattened [13]. All these additional artifacts can harm a certain aspect of the audio quality, and eventually the performance of a subsequent task, too.

This dissertation does not attempt to resolve all the abovementioned issues. Instead, we are going to cover the source separation part in detail, but focusing more on the efficiency of the machine learning algorithms involved in the procedure. Especially, between the two different source separation scenarios, *over-determined* and *under-determined* cases, we are going to focus more on the latter situation, where the number of observations is smaller than that of sources. We are particularly interested in an extreme case when only a *single-channel* recording of multiple sources is available. Chapter 3 discusses these topics. We will touch upon the other artifacts in Chapter 5 as well.

Yet, some of those audio enhancement tasks such as dereverberation, bandwidth expansion, and declipping can share a similar algorithm with single-channel source separation. Therefore, some frameworks discussed here particularly for source separation have a potential use for the other tasks as well.

1.2 Why Does Efficiency Matter? – Some Audio Applications

Audio enhancement is generally a difficult task in terms of its performance. However, if the desired level of performance can only be achieved by putting a great deal of resources, then we need to start worrying about the trade-off between performance and cost. In this section, we cover a few scenarios that need those considerations about efficiency.

Note that in this dissertation we concentrate more on the efficiency during the test time rather than at training, although training efficiency also matters in some research involving big data. The reason behind

this is the facts that (a) we still need to use some unsupervised learning algorithms, such as topic modeling or matrix factorization, where an EM-like iterative updates are required during the test time (b) there are many real-time audio applications where runtime efficiency is a critical issue (c) for some resource-constrained devices even a well-trained supervised system can be a burden during the test time.

1.2.1 Big audio data processing: an Information Retrieval (IR) system

We can think of an IR system whose query and collection are all audio signals. If the collection is very large, then the matching procedure can be a demanding job even in a computing environment with enough resources, because the response time greatly influences the user experience. We consider this kind of IR system on a big collection of audio clips as an example of big audio data processing. Examples of those big IR systems can be found in music search applications, such as Shazam[1], SoundHound[2], Sound Search for Google Play[3], Gracenote's MusicID[4], and so on.

In those systems, a query can be either a recorded excerpt of the original sound clip or a recording of users' humming or singing, both are potentially captured in a noisy environment. As for the collection, it is obvious that it can contain easily over millions of songs. Due to its size a reasonably fast matching procedure is already a challenging task, which can be tackled by a hashing technique that converts the time-frequency representation of audio into bit strings [14]. In the hashing-based matching approach, the bit strings are carefully designed so that matching can be done robustly even with queries recorded in a noisy environment and under an aggressive coding procedure.

While the IR system using recorded audio queries shows an industry-strength search performance, there can be more difficult types of matching problems where matching is loosely defined. For example, in the Query by Singing/Humming (QbSH) is an example where the query is not a deformed version of the originally identical song in the collection, but a monophonic pitched signal that can be only similar to the dominant melody of the song at best [15]. Aside from the difficulty in matching off-pitched and off-beat queries by humming and singing to the ground truth melody, the system also need to suppress the other accompanying instrumental sound, too. In this kind of soft matching problem in general, i.e. Audio Information Retrieval (AIR), which can eventually cover a lot more diverse set of sound clips, a successful system should be able to retrieve as similar entities as possible based on its robust matching between a noisy query and the collection of mixed sound. Moreover, the retrieval must be fast and efficient enough.

Chapter 4 introduces a couple of efficient matching algorithms that are specially designed for audio

[1] http://www.shazam.com
[2] http://www.soundhound.com
[3] https://play.google.com/store/apps/details?id=com.google.android.ears
[4] http://www.gracenote.com/music/music-recognition/

search problems. To address the requirement about soft matching and the algorithmic efficiency, the proposed methods use either a hash function that is designed to be resistant to temporal stretches or a smoothing technique based on Parzen windowing [16] on a sparse representation. As for the hashing technique, we take care of the interferences by relying on the robustness of the hash function to the additive noise. On the other hand, the smoothing technique more directly handles the unmixing problem by harmonizing topic modeling in its sparse representation. See Section 4.1 and 4.2 for more detail, respectively.

1.2.2 Big ad-hoc sensor arrays

Nowadays, many people carry personal hand-held devices, e.g. smartphones, and record a scene using them as a camera or a microphone. We can see hundreds of clips in Youtube[5] that recorded the same scene from different angles, such as a famous musician's concert or an inauguration speech. It is an interesting crowdsourcing problem if we would like to consolidate these recordings in order to come up with an enhanced version of the audio scene, because we replace a potentially expensive professional recording job with the inexpensive workforce. As a crowdsourcing problem we need to figure out how to "denoise" the user-created recordings aside from the fact that we need to synchronize the signals in the first place. This set of recordings can be also seen as social audio data, because people use their own recording to share their particular feelings or impressions about the event.

Another view of this dataset is that we can regard the devices spread in the scene as a loosely connected microphone array, or an *ad-hoc* sensor array [17, 18]. An ad-hoc sensor array is trickier to process using traditional array processing algorithms, since the sensors are not synchronized and their characteristics are not known, while the prior knowledge is important to estimate the direction of arrival based on the delays of the sound wave observed at different sensor locations [19, 20].

On top of the difficulty due to the ad-hoc setting, the potential size of the problem is noteworthy, because it can grow into a big data problem as we are interested in a huge amount of recordings from hundreds or thousands of people. Once we have to deal with this amount of data, source localization is a very difficult job, even with the special consideration about the ad-hoc setting, because each of those user-created recordings is contaminated with a unique set of artifacts, such as someone else's sining-along right next to me which is not audible from the other sensors, different audio coding techniques, a microphone specific frequency response, etc. The larger size increases the probability of having not only quality recordings in the set, but also bad ones. Synchronizing all those uniquely deformed signals also calls for a lot of efficiency and robustness to the variety.

[5] https://www.youtube.com

In this dissertation we call the big ad-hoc sensor array problem Collaborative Audio Enhancement (CAE) to distinguish it from the ordinary definition of ad-hoc sensor arrays, which does not actually take the complication caused by the size and variety of the social recordings into account. Chapter 5 introduces a probabilistic latent topic model that learns a set of shared topics from the recordings to represent a common and dominant audio source, while setting aside a few individual recording-specific topics that capture the local artifacts that are not common. In order to process a large amount of dataset, we speed up this procedure by using a nearest neighborhood search on the recordings, which are then used as the reduced input to the shared topic model.

1.2.3 Audio computing with restricted resources

Even if the amount of data is ordinary, some systems with only limited available resources still need to seek efficiency in the implementations. We can think of many resource-sensitive applications where the comprehensive machine learning techniques can be burdensome. For example, we can think of some always-on spoken keyword spotting systems that necessitate implementations that minimize resource usage, something that is imperative with personal assistant services, e.g. Google Now, S-voice, and Siri. For example, one can use the keyword to initiate those services, such as "Hey, Siri", "Hi, Galaxy", "OK, Google". However, they basically assume an always-on one-word speech recognition system that is running in the background, which can potentially drain the battery. Therefore, the implementation must be mindful of resource usage. Moreover, in general contextual information that can be induced from analyzing the signals captured by always-on sensors is also limited for the same reason.

From this, we can easily think of some context-aware computing scenarios that are based on the analysis of various sensor signals. Context-aware computing is challenging not only due to the needs for a high level of artificial intelligence, but its requirement for power-efficiency. The latter requirement comes from the fact that context intelligence involves pattern recognition processes that are always running on mobile devices with limited resources, e.g. proximal discovery, geofencing, device position classification, motion information recognition, and the Gimbal platform on the service providers' side[6]. They have to analyze multiple types of sensor signals to perform a various recognition tasks, while minimizing their footprint on memory and power resources. This is especially the case in wearable devices which came with even lesser resources.

In those real-time systems the main issue is the trade-off between the application goals, e.g. a desired speed and performance, and the use of restricted resources. For example, if a learning algorithm involved in

[6]http://gimbal.com

the application tends to produce better results up to a certain number of iterations, nothing stops a system from conducting the necessary amount of computations, except when doing so can take up too much time in a real-time systems. One can try to speed up the procedure by allocating more resources, e.g. through parallelizing, but if this speed-up drains too much resource, for example batteries in embedded systems, the faster implementation is not a welcome improvement.

This dissertation introduces a few speed-up techniques that blend well with existing machine learning models for audio computing. At the same time, we save the time not by using more resources, but by relying on bitwise computations on the transformed binary audio features and by narrowing the solution space down to a small number of candidates. Eventually, the proposed efficient machine learning algorithms vastly reduce the use of resources to achieve the speed. These advantages come at a cost – the lightweight models tend to perform worse than their corresponding comprehensive models. In other words, our goal is to get the efficiency while sacrificing as little performance as possible.

1.3 Outline

The dissertation consists of a few chapters that are devoted to some efficient machine learning frameworks and their applications to audio computing. Before we get into the details of the proposed models, in Section 2.1 introduces NMF as one of the basic source separation model on magnitude spectra. Section 2.2 shows some basics about single-channel source separation will be covered in the context of topic modeling or matrix factorization. Section 2.3 will also introduce some background material about a particular hashing technique, namely WTA hashing, which we will heavily use for our efficient computing.

We will start to go over some efficient source separation algorithms using variants of topic modeling in Chapter 3. Section 3.1 introduces a new concept where the dictionary-based model can preserve the data manifold by grouping the dictionary into some local dictionaries. Section 3.2 proposes a more direct way to preserve the data manifold during source separation, where the manifold preserving topic models reduce the complexity of a sparse coding procedure by having some hierarchy in the topic model. We speed up this hierarchical topic model by using WTA hashing in 3.3 while not sacrificing the performance.

Some efficient audio pattern matching frameworks are proposed in Chapter 4. In Section 4.1 we will see that a carefully designed hashing technique can be used as the main pattern recognition engine for a keyword spotting problem, especially in the presence of additive noise. Section 4.2 follows to investigate another efficient sparse representation of audio signals, and a more direct source separation functionality in the model. These efficient frameworks have potential applications in audio search tasks in general.

CAE is a relatively new audio application we studies in Chapter 5. In Section 5.1 a baseline topic model is proposed to capture the common source while suppressing the recording-specific artifacts, along with an introduction to the basic concepts and assumptions of the problem. Section 5.2 is for some suggestions about making this procedure more efficient by incorporating with a nearest neighbor search over the entire dataset given a tentative source reconstruction. It enlarges the experiments so that the system can cover up to 1,000 recordings in a reasonable runtime.

Finally, a new neural network framework is proposed in Chapter 6, where all the participating values and the operations on them are defined in a bitwise fashion. We call this network Bitwise Neural Networks (BNN). We will see that we can easily extend the usual backpropagation using Stochastic Gradient Descent (SGD) to train this highly quantized network with an acceptable performance loss. BNNs have a lot of potential given their extremely simple and hardware-friendly forwardpropagation procedure, especially for embedded devices.

Chapter 2

Background

2.1 Nonnegative Matrix Factorization (NMF) for Source Separation

Nonnegative Matrix Factorization (NMF) [21, 22] has been widely used in audio research, e.g. automatic music transcription [5], musical source separation [23], speech enhancement [24], etc. Once the signal is transformed into a nonnegative matrix, e.g. by using Short-Time Fourier Transform (STFT) and taking its magnitudes, the flexible approximation of the NMF model can successfully provide intuitive analysis results.

NMF takes a nonnegative matrix $V \in \mathbb{R}_+^{M\times N}$, where $\mathbb{R}_+$ stands for real values bigger than or equal to zero. Then, it seeks nonnegative factor matrices $W \in \mathbb{R}_+^{M\times K}$ and $H \in \mathbb{R}_+^{K\times N}$, whose product minimizes an error function, $\mathcal{D}(V||WH)$, between the input and the reconstruction. It is common to use β-divergence as a generalized error function of NMF problems,

$$\mathcal{D}_\beta(x||y) = \begin{cases} \frac{x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}}{\beta(\beta-1)}, & \beta \in \mathbb{R}\backslash\{0,1\} \\ x(\log x - \log y) + (y-x), & \beta = 1 \\ \frac{x}{y} - \log\frac{x}{y} - 1, & \beta = 0, \end{cases} \tag{2.1}$$

as it covers Frobenius norm, unnormalized KL-divergence, and Itakura-Saito divergence when $\beta = 2$, 1, and 0 as special cases [25]. Thus, we can define the β-divergence NMF problem as follows:

$$\arg\min_{W,H} \mathcal{J}_{\text{NMF}} = \mathcal{D}_\beta(V||WH), \text{s.t. } W \geq 0, H \geq 0. \tag{2.2}$$

The standard NMF algorithm solves this constrained optimization problem by representing the gradient descent method via a set of multiplicative update rules. The multiplicative update rules can be equivalently derived by considering the negative and positive terms of the partial derivatives as numerator and

denominator of the factor. Therefore, the update rules are:

$$W \leftarrow W \odot \frac{[\frac{\partial \mathcal{J}_{\mathrm{NMF}}}{\partial W}]^-}{[\frac{\partial \mathcal{J}_{\mathrm{NMF}}}{\partial W}]^+} = W \odot \frac{\{(WH)^{\cdot(\beta-2)} \odot V\}H^\top}{(WH)^{\cdot(\beta-1)}H^\top}, \tag{2.3}$$

$$H \leftarrow H \odot \frac{[\frac{\partial \mathcal{J}_{\mathrm{NMF}}}{\partial H}]^-}{[\frac{\partial \mathcal{J}_{\mathrm{NMF}}}{\partial H}]^+} = H \odot \frac{W^\top\{(WH)^{\cdot(\beta-2)} \odot V\}}{W^\top(WH)^{\cdot(\beta-1)}}, \tag{2.4}$$

where $\odot$ represent the Hadamard product and division and exponentiation are carried in the element-wise manner, too. $[\cdot]^-$ and $[\cdot]^+$ are negative and positive terms of the expression inside $[\cdot]$, respectively. Once initialized with nonnegative values, the sign of the parameters W and H remains positive during the process as it only involves multiplications by nonnegative factors, thereby ensuring the desired parameter nonnegativity.

2.1.1 Dictionary-based source separation using NMF

This section reviews a common source separation procedure that uses NMF basis vectors as a dictionary.

Dictionary learning

For each source c, either $c = S$ for speech or $c = N$ for noise, we first perform the STFT and take the magnitude to build a source specific nonnegative training matrix $V^c_{dic} \in \mathbb{R}_+^{M \times N_c}$. NMF then finds a pair of factor matrices $W^c_{dic} \in \mathbb{R}_+^{M \times R_c}$ and $H^c_{dic} \in \mathbb{R}_+^{R_c \times N_c}$ that define a convex cone to approximate the input: $V^c_{dic} \approx W^c_{dic}H^c_{dic}$ [21, 22]. Among all the possible choices of β-divergences to measure the approximation error as proposed in [25], we focus on the case $\beta = 1$, or a generalized KL-divergence as follows:

$$\mathcal{D}(x|y) = x(\log x - \log y) + (y - x). \tag{2.5}$$

The parameters W^c_{dic} and H^c_{dic} that minimize the error $\mathcal{D}(V^c_{dic}|W^c_{dic}H^c_{dic})$ are estimated by changing the step size of the gradient descent optimization so that they are updated in a multiplicative way:

$$\begin{aligned} W^c_{dic} &\leftarrow W^c_{dic} \odot \left\{ \left(\frac{V^c_{dic}}{W^c_{dic}H^c_{dic}} \right) {H^c_{dic}}^\top \right\} \Big/ \left\{ \mathbf{1}{H^c_{dic}}^\top \right\}, \\ H^c_{dic} &\leftarrow H^c_{dic} \odot \left\{ {W^c_{dic}}^\top \left(\frac{V^c_{dic}}{W^c_{dic}H^c_{dic}} \right) \right\} \Big/ \left\{ {W^c_{dic}}^\top \mathbf{1} \right\}, \end{aligned} \tag{2.6}$$

where the Hadamard product $\odot$ and division are carried out in the element-wise fashion. Once the parameters are initialized with nonnegative random numbers, their sign stays the same after the updates. The

learned basis vectors W_{dic}^{c} per each source represent the source as a dictionary.

Source separation

Some concepts should be clarified before we introduce the usual single-channel source separation procedure using NMF. First, the definition of sources can be vague depending on what we want to do in the applications. If we want to transcribe music signals, each musical note can be seen as a source even if they are all from the same polyphonic instrument, e.g. piano [5]. On the other hand, some can consider each instrument as a source, such as drums [26] and singing voice [27]. In single-channel speech enhancement tasks [28], we often assume that there are two sources: speech and noise, although it is still vague whether to consider an additional interfering voice as one of the speech sources or noise. Although this kind of examples cannot cover the entire source separation scenarios, the techniques that I will cover can be extended to the other cases without loss of generality.

The speech enhancement (or source separation in general) procedure on the unseen noisy signals is done by learning the activations of the corresponding dictionaries learned from the procedure in the previous clause. For a noisy spectrogram $V_{test} \in \mathbb{R}_{+}^{M \times N_{test}}$, the activation per each source c is estimated as follows:

$$H_{test}^{c} \leftarrow H_{test}^{c} \odot \left\{ {W_{dic}^{c}}^{\top} \left(\frac{V_{test}}{W_{dic} H_{test}} \right) \right\} \Big/ \left\{ {W_{dic}^{c}}^{\top} \mathbf{1} \right\}, \tag{2.7}$$

where $W_{dic} = [W_{dic}^{S}, W_{dic}^{N}]$ and $H_{test} = \begin{bmatrix} H_{test}^{S} \\ H_{test}^{N} \end{bmatrix}$. Note that we call this case the *supervised* separation since both the speech and noise dictionaries are known. We do not usually update the learned dictionaries during the supervised separation. Finally, the speech part of the test spectrogram is recovered by masking the mixture matrix by the proportion of the speech estimate in the total reconstruction:

$$V_{test}^{S} \approx V_{test} \odot (W_{dic}^{S} H_{test}^{S}) \Big/ (W_{dic} H_{test}). \tag{2.8}$$

In the semi-supervised separation either the speech or noise training set is not available [29]. If the noise dictionary W_{dic}^{N} is unknown, it has to be learned from the mixture signal, calling for an update of the dictionary in addition to (2.7):

$$W_{dic}^{N} \leftarrow W_{dic}^{N} \odot \left\{ \left(\frac{V_{test}}{W_{dic} H_{test}} \right) {H_{test}^{N}}^{\top} \right\} \Big/ \left\{ \mathbf{1} {H_{test}^{N}}^{\top} \right\}. \tag{2.9}$$

2.2 Probabilistic Latent Semantic Indexing (PLSI) for Source Separation

Probabilistic topic models have been widely used for various applications, such as text analysis [30, 31, 32], recommendation systems [33], visual scene analysis [34], and music transcription [35, 25]. A common intuition behind such models is that they seek a convex hull that wraps the input M-dimensional data points in the $M-1$ dimensional simplex. The hull is defined by the positions of its corners, also known as basis vectors, whose linear combinations reconstruct the inputs inside the hull.

Although these linear decomposition models provide compact representations of the input by using the learned convex hull, an ambiguity exists: the hull loses the data manifold structure as it redundantly includes areas where no training data exist. This is problematic especially when the input is a mixture of distinctive data sets with heterogeneous manifolds. In this case, the desirable outcome of this analysis is not only to approximate the input, but to separate it into its constituent parts, which we will refer to as sources. In text these could be sets of topics, in signal processing they could be independent source signals, etc.

Without knowing the nature of each source, the separation task is ill-defined. Hence, it is advantageous to start with learned sets of basis vectors. Each set approximates the training data of a particular source. Figure 2.1 depicts a separation result using Probabilistic Latent Semantic Indexing (PLSI) [30, 31]. In this example two data sets are modeled using their four-cornered convex hulls (red and blue dashed polytopes)

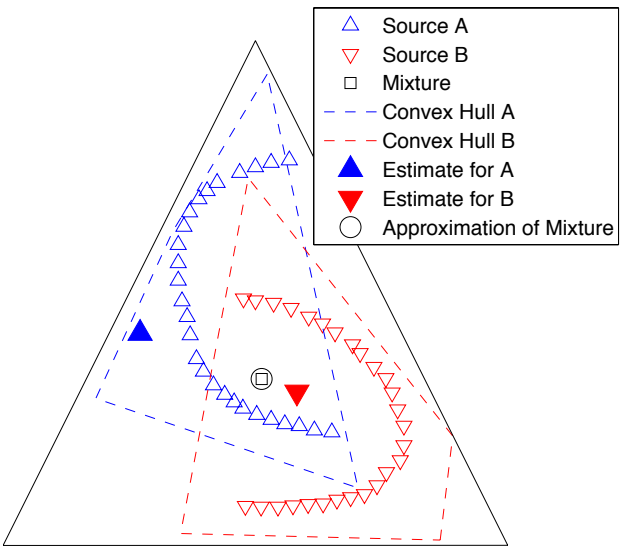


Figure 2.1: Separation using convex hulls of sources that are learned from PLSI.

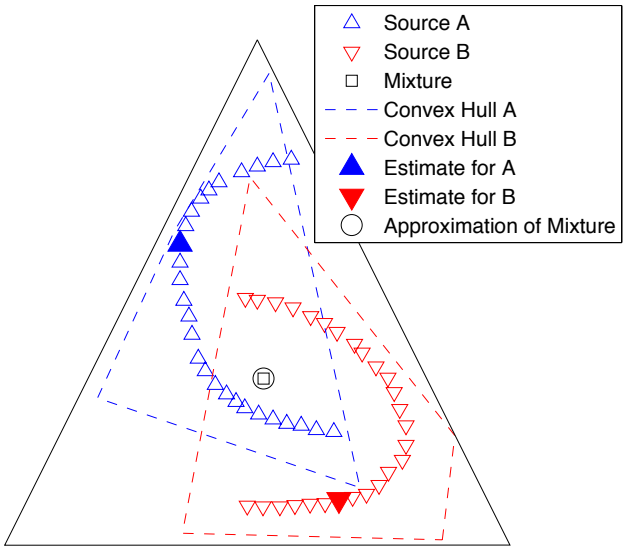


Figure 2.2: Separation by sparse coding on all the training data points.

as computed by PLSI, respectively. Once confronted with a new data point that is a linear mixture of these two classes (black square) we can decompose it using the already-known models. As seen in the simulation in the figure, the combination of the learned convex hulls can jointly approximate that mixture point very well (black circle), but estimates for the two source points that constitute the mixture (blue and red filled triangles) lie outside of the original two manifolds, thereby providing poor separation of sources. For instance, in the speech separation scenario the separated speeches do not reflect the characteristics of the sources while their mixture sounds a lot like the mixed signal.

If we use all the overcomplete training samples as candidate topics and force them to be activated in a very sparse way, it can be an alternative to the convex hull representation [36]. By doing so we can prevent reconstructions from being placed in areas away from the data manifold. For instance, in Figure 2.2, only one training sample from each class participated as a topic in estimating the constituent sources. Thus, the sparsity constraint can confine the source-specific reconstructions to lie on the data manifolds.

2.2.1 Topic models and separation with sparse coding

Probabilistic Latent Semantic Indexing (PLSI)

Ordinary topic models, such as PLSI, take a matrix as input, whose column vectors can be seen as observations with multiple entries, e.g. news articles with finite set of words, sound spectra with frequency bin energies, vectorized images with pixel positions, etc. The goal of the analysis is to find out topics $P(f|z)$

and their mixing weights $P_t(z)$ that best describe the observations $X_{f,t}$ as follows:

$$X_{f,t} \sim \sum_z P(f|z)P_t(z), \tag{2.10}$$

where t, f, and z are indices for observation vectors, elements of a topic, and the latent variables, respectively. The Expectation-Maximization (EM) algorithm is common to estimate the model parameters, and in this case this works by minimizing the sum of cross entropy between $X_{f,t}$ and $\sum_z P(f|z)P_t(z)$ for all t:

E-step:

$$P_t(z|f) = \frac{P(f|z)P_t(z)}{\sum_z P(f|z)P_t(z)}$$

M-step:

$$P(f|z) = \frac{\sum_t X_{f,t}P_t(z|f)}{\sum_{f,t} X_{f,t}P_t(z|f)}, \quad P_t(z) = \frac{\sum_f X_{f,t}P_t(z|f)}{\sum_{f,z} X_{f,t}P_t(z|f)}.$$

For example, in Figure 2.1, we can construct the convex hull of source A by taking source A's training data as input $X^A_{f,t}$ and getting $P_A(f|z)$ as four corners of the hull, which are designated by z. $P_t(z)$ is the mixing weight of z-th corner to reconstruct t-th input.

Sparse Probabilistic Latent Semantic Indexing (PLSI)

The t-th data point of the mixture input $X^M_{f,t}$ is an observation drawn from a multinomial distribution, which is a convex sum of multiple sources s:

$$X^M_{f,t} \sim \sum_s P_t(f|s)P_t(s), \tag{2.11}$$

where t-th source multinomial $P_t(f|s)$, which corresponds to the filled triangles in Figure 2.1 and 2.2, can be further decomposed into combination of topics (corners) as in (2.10) by seeing $P_t(f|s)$ as input:

$$P_t(f|s) \sim \sum_z P_s(f|z)P_t(z|s). \tag{2.12}$$

As discussed in the previous section, it is convenient to pre-learn the source-specific topics, $P_s(f|z)$. For instance, if we learned several political topics as $P_{s_1}(f|z)$ and medical topics as $P_{s_2}(f|z)$, respectively, we can reconstruct a news article about a medical bill in the council. For a spectrum representing a mixture of speech signals of two different people, we can reconstruct it as a weighted sum of speaker-wise estimates

by using each individual's sets of "topic" spectra. In other words, a mixture input $X_{f,t}^M$ can require more than one set of similar topics, as opposed to the traditional use of PLSI where the input is not a mixture of multiple sources.

With the learned and fixed topics per each source $P_s(f|z)$, the rest of the separation analysis consists of inferring global source weights $P_t(s)$ and source-wise reconstruction weights $P_t(z|s)$ using EM:

E-step:

$$P_t(s,z|f) = \frac{P_t(s)P_t(z|s)P_s(f|z)}{\sum_s P_t(s) \sum_{z \in \mathbf{z}^{(s)}} P_t(z|s)P_s(f|z)},$$

M-step:

$$P_t(z|s) = \frac{\sum_f X_{f,t}^M P_t(s,z|f)}{\sum_{f,z} X_{f,t}^M P_t(s,z|f)}, \quad P_t(s) = \frac{\sum_f X_{f,t}^M \sum_{z \in \mathbf{z}^{(s)}} P_t(s,z|f)}{\sum_f X_{f,t}^M \sum_s \sum_{z \in \mathbf{z}^{(s)}} P_t(s,z|f)}, \tag{2.13}$$

where $\mathbf{z}^{(s)}$ is a set of topic indices for source s.

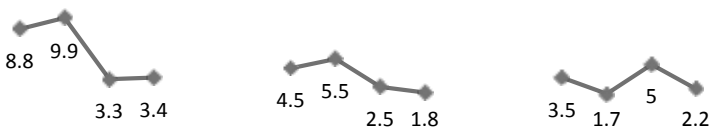
The sparse PLSI model additionally assumes that the weights $P_t(z|s)$ and $P_t(s)$ are sparse, so that the mixture and source estimation in (2.11) and (2.12) try to use less number of sources $P_t(f|s)$ and topics $P_s(f|z)$, respectively. Furthermore, instead of using the corners of the learned convex hull as topics, the sparse PLSI requires the topics to be the source specific training data itself. Consequently, $P_t(z|s)$ has weights on only a very small portion of the training points as active topics. These two properties result in a manifold-preserving source estimate during the separation procedure. Obviously this is a demanding operation as the training data can be a large data set resulting in an unusually high number of topics.

We will discuss about the way of employing sparsity constraints in the EM algorithm more specifically in Section 3.2.

2.3 Winner Take All (WTA) Hashing

The recent application of Winner Take All (WTA) hashing [37] to a big image searching task provided accurate and fast detection results [38]. As a kind of locality sensitive hashing [39], it has several unique properties: (a) similar data points tend to collide more (b) Hamming distance of hash codes approximately reflects the original distance of data. Therefore, it can be seen as a distribution on a family of hash functions $\mathcal{F}$ that takes a collection of objects, such that for two objects x and y, $\Pr_{h \in \mathcal{F}}[h(x) = h(y)] = \text{sim}(x,y)$. $\text{sim}(x,y)$ is some similarity function defined on the collection of objects [40].

WTA hashing is based on the rank correlation measure that encodes relative ordering of elements. Al-



$x_1 = [8.8, 9.9, 3.3, 3.4]$ $x_2 = [4.5, 5.5, 2.5, 1.8]$ $x_3 = [3.5, 1.7, 5.0, 2.2]$

(a) Three input vectors

8.8	9.9	**2**
8.8	3.3	**1**
8.8	3.4	**1**
9.9	3.3	**1**
9.9	3.4	**1**
3.3	3.4	**2**

$c_1 = [2, 1, 1, 1, 1, 2]$

4.5	5.5	**2**
4.5	2.5	**1**
4.5	1.8	**1**
5.5	2.5	**1**
5.5	1.8	**1**
2.5	1.8	**1**

$c_2 = [2, 1, 1, 1, 1, 1]$

3.5	1.7	**1**
3.5	5.0	**2**
3.5	2.2	**1**
1.7	5.0	**2**
1.7	2.2	**2**
5.0	2.2	**1**

$c_3 = [1, 2, 1, 2, 2, 1]$

(b) All possible pairwise orders (ideal codes)

1	2
1	3
4	2

$\mathcal{P} \in \mathbb{R}^{3\times 2}$

(c) A permutation table

8.8	9.9	**2**
8.8	3.3	**1**
3.4	9.9	**2**

$c_1 = [2, 1, 2]$

4.5	5.5	**2**
4.5	2.5	**1**
1.8	4.5	**2**

$c_2 = [2, 1, 2]$

3.5	1.7	**1**
3.5	5.0	**2**
2.2	1.7	**1**

$c_3 = [1, 2, 1]$

(d) Hash codes generated according to $\mathcal{P}$

Figure 2.3: A WTA hashing example. (a) x_1 and x_2 are similar in their shape while x_3 looks different. (b) Exhaustive pairwise ordering can give ideal hash codes c, but the feature space (hash code) is high dimensional. (c) A permutation table provides a fixed set of random pairwise selections. (d) $\mathcal{P}$ generates succinct hash codes that approximate the original similarity of inputs.

though the relative rank order can work as a stable discriminative feature, it non-linearly maps data to an intractably high dimensional space. For example, the number of orders in M-combinations out of an F-dimensional vector is (# combinations) $\times$ (# orders in each combination) $= \frac{F!}{M!(F-M)!} \times M$. Instead, WTA hashing produces hash codes that compactly approximate the relationships.

WTA hashing first defines a permutation table $\mathcal{P} \in \mathbb{R}^{L\times M}$ that has L different random index sets, each of which chooses M elements. For the l-th set the position of the maximal element among M elements is encoded instead of the full ordering information. Therefore, the length of hash codes is ML-bits since each permutation results in M bits, where only one bit is on to indicate the position of the maximum, e.g. $3 = 0100$, and there are L such permutations. Whenever we do this encoding for an additional permutation, at most $M - 1$ new pairwise orders (maximum versus the others) are embedded in the hash code. The permutation table is fixed and shared so that the hashing results are consistent. Figure 2.3 shows the hashing procedure on simple data. Note that the Euclidean distance between x_1 and x_2 are larger than that

of x_2 and x_3 as opposed to the similarity in their shapes. WTA hashing results in hash codes that respect this shape similarity. Note also that WTA hashing is robust to the difference between x_1 and x_2, which can be explained as some additive noise.

Even though WTA hashing provides stable hash codes that can potentially replace the original features, the approximated rank orders cannot always provide the same distance measure with the original ones, e.g. cross entropy. Therefore, in this dissertation, we only use this hashing technique to reduce the size of the solution space based on Hamming distance, and then do the nearest neighbor search in this reduced candidate neighbor set rather than dealing with the entire data samples.

WTA hashing can be easily extended to multi-dimensional inputs. For instance, a 2D image representation of audio can be vectorized as an input to the hash function. However, as for speech signals, sometimes we have to take the spectral and temporal invariances into account. In Section 4.1, a variant of WTA hashing is discussed to handle this issue.

Chapter 3

Efficient Source Separation

In this chapter, we are going to discuss some NMF or PLSI-based single-channel source separation techniques, where we put a stress on the efficiency during the test time. Since both NMF and PLSI algorithms for separation can be seen as dictionary-based methods, where we learn a dictionary per source in advance during the training phase, and then estimate the activation of the dictionary items during the test time.

In those dictionary-based separation models, the estimation of activations is done by using EM updates, which introduce some complexity during the test time. This can be burdensome for some systems where a large dictionary is involved for better separation performance. In Section 3.1 we first introduce a manifold preserving NMF model that can harmonize multiple local dictionaries to build up a large dictionary that performs better than a usual universal, but small dictionary in terms of the separation quality. In Section 3.2 we see that this manifold preservation concept can be extended to the PLSI case while we still need an efficient algorithm to achieve sparse coding. Section 3.3 introduces a hashing technique that converts this dictionary encoding procedure into a nearest neighbor search problem on hash MLD codes, which can speed-up the procedure without losing the performance.

3.1 Mixture of Local Dictionaries (MLD)

A key strategy for applying NMF towards single channel speech enhancement is to model a source's training set (usually magnitude spectra) with a dictionary that consists of a small number of basis vectors. These basis vectors should be able to approximate all of the source's produced spectra. Since both bases and weights are nonnegative, the dictionary learned from NMF eventually models the input magnitude spectra with a convex cone (more precisely, a simplicial cone [42]). Then, the main goal of the single channel speech enhancement becomes that of representing the noisy input spectrum as a weighted sum of speech bases and estimated noise bases. The source estimates are forced to lie inside their respective convex cones. Therefore, the source-specific convex cone concept is critical for the denoising performance since the less

Part of Section 3.1 has been published in [41].

the convex cones overlap each other, the more discriminative they are.

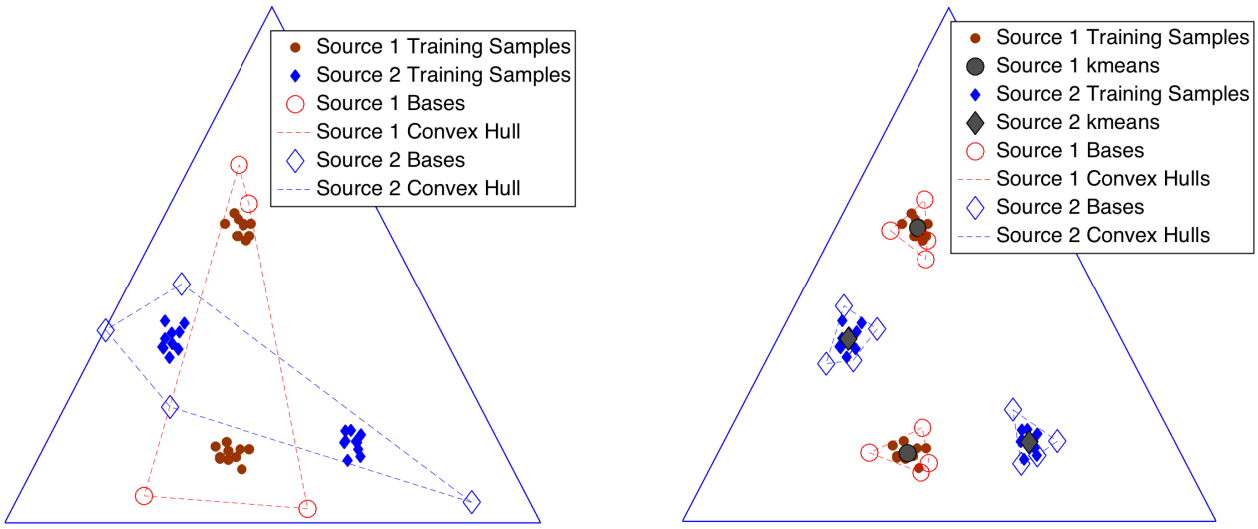
However, this linear decomposition model can be limited when it comes to modeling a complex source manifold. Although NMF has been found to be a suitable model for analyzing audio spectra because of its flexible additive nature, this flexibility sometimes hinders the ability to discriminate between different sources. If all the data points on a complex manifold structure should belong to a convex set defined by the basis vectors, it is inevitable that this convex cone will include some unnecessary regions where source spectra cannot reside.

We propose a model for learning Mixture of Local Dictionaries (MLD) along the lines of recent attempts to preserve the manifold of the audio spectra. The basic idea is to learn dictionaries using the sparse coding concept to better approximate the input [43]. Particularly, in [36] a non-parametric overcomplete dictionary model was proposed that fully makes use of the entire training spectra instead of discarding them after learning their convex model. This method encourages the source estimates to lie on the manifold by approximating them with only a very small number of training samples. A succeeding model proposed a more direct way by encompassing only the nearest neighbors for the source estimation [44].

Another relevant work is the Universal Speech Model (USM) [45]. USM tackles the case where the identity of the speaker is unknown and clean speech signals from anonymous speakers are available for training instead. Since the anonymous training spectra can have too much variance, a naïve NMF approach that learns a single convex cone from the entire set of training signals can produce less discriminative results than the hypothetical ones learned from the ideal speaker. In order to address that, USM first uses regular NMF on each speaker to learn a speaker-specific convex cone, and then during the separation stage it only activates a very small number of speakers' dictionaries at a time, the ones that best fit the observed data. To this end, USM involves a block sparsity constraint that was also used in [46, 47], so that irrelevant speakers' basis sets are turned off in the group-wise manner.

The proposed MLD model intends to preserve the manifold of the source data in a more controlled way. The benefit of using MLD comes from the following points:

- During training MLD discovers several convex cones per a source, each of which covers a chunk of similar spectra across all speakers rather than one per a speaker.
- MLD penalizes the difference between each local dictionary and its *a priori*, such as in the Maximum A Posteriori (MAP) estimation, to make the learned bases to be more concentrated on the prior. As a result, each convex cone covers a smaller area than without Maximum A Posteriori (MAP).
- During denoising MLD activates only a small number of dictionaries for a given noisy input spec-



(a) Results from NMF

(b) The proposed MLD model

Figure 3.1: A comparison of convex hulls learned from a toy dataset.

trum. Because MLD makes this decision in the frame-by-frame way, the model dynamically finds an optimal fit while USM approach does this in a global sense over time.

3.1.1 The MLD model

In this section we first address some downsides of the conventional NMF model with respect to source manifold preservation, and introduce the proposed MLD approach.

Locality in data manifolds

Fig. 3.1 (a) depicts the behavior of NMF dictionaries. For illustrational convenience we project the input vectors onto the simplex as if they are normalized. In this toy example, there are three spectral features that correspond to the three corners of the simplex. In Fig. 3.1 there are two different kinds of sources represented with small red dots and blue diamonds respectively. If we learn a set of four basis vectors that describe each source, they define the corners of a convex hull[1] (empty diamonds and circles), which surrounds the data points.

Each source has a manifold structure that consists of two distinct clusters, but NMF does not take it into account and results in a convex dictionary that wraps both intrinsic clusters. Hence, each convex hull can reconstruct not only the training data, but also spurious cases in the areas where the data is very unlikely to exist. On top of that, since NMF does not guarantee that the convex hull will surround the data tightly,

[1]Since the data points are normalized for the simplex representation, the convex cone learned by an NMF run reduces to a hull.

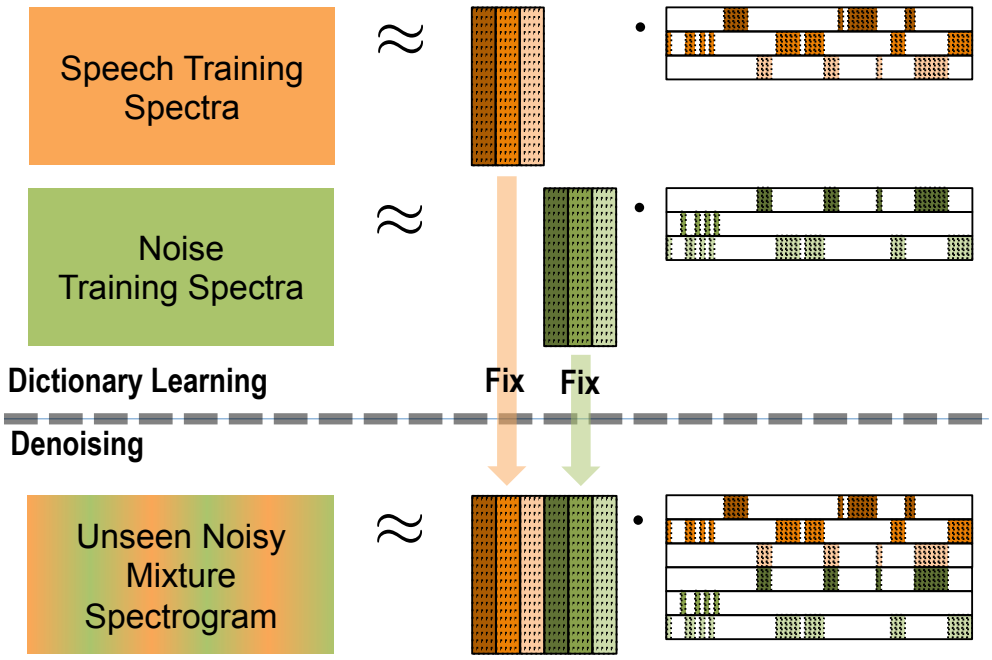


Figure 3.2: A block diagram for the full speech enhancement procedure including source specific dictionary learning and its (block) sparse coding with the learned dictionaries.

the dictionary can include even more unnecessary regions. Meanwhile, when we unmix them, we use only the learned bases after discarding the training samples. Since the bases from the sources do not precisely represent the individual source's structure, we end up with overlapping convex hulls. This is problematic, because when an estimated source spectrum falls in the overlapped region of the red and blue convex hulls, it is impossible to identify which source it belongs to.

On the other hand, MLD learns a set of small convex dictionaries per each source, similarly to the way that a mixture model approximates an arbitrary distribution. The underlying latent modality of a mixture model corresponds to each convex cone in MLD. In Fig. 3.1 (b) we can see that MLD successfully tracks the data manifold with two disjoint convex hulls. Moreover, the hull wraps the data points tightly enough to include as small empty space as possible. Hence, we can expect a better preserved source manifolds in the learned dictionaries.

Mixture of Local Dictionaries (MLD): algorithms

Fig. 3.2 describes the entire speech enhancement procedure using MLD. We first have to learn basis vectors from each source (orange and green) as in the dictionary learning procedure introduced in 2.1. In MLD however, the basis vectors for a source are grouped into a pre-defined number of blocks, e.g. in the figure five bases per a block and three blocks per a source, each of which corresponds to a convex cone, or a local

dictionary. At the same time the activation matrix H is learned to be block-wise sparse, so that a training sample belongs to only one or a very small number of local dictionaries. We use these learned dictionaries as they are when we perform the denoising job on some unseen noisy signals while we newly learn a block-wise sparse encoding matrix, H.

The objective function $\mathcal{J}$ is defined as follows:

$$\mathcal{J} = \mathcal{D}(V|WH) + \lambda \sum_t \Omega(h_t) + \eta \sum_g \mathcal{D}(\mu^{(g)}|W^{(g)}), \tag{3.1}$$

where $W = [W^{(1)}, \cdots, W^{(G)}]$, $H = [h_1, \cdots, h_N]$ and $h_t = [{h_t^{(1)}}^\top, \cdots, {h_t^{(G)}}^\top]^\top$. t and g are for the frame and group indices. The first term stands for KL-divergence in (2.5) from the original NMF algorithm. The function Ω on g-th block of each frame t is to give penalty to the solutions that are not sparse. In particular, we use $\log / l_1$ penalty, $\Omega(h_t) = \sum_g \log(\epsilon + ||h_t^{(g)}||_1)$, which was also used in [47, 45], for its monotonicity and induced multiplicative updates. The third term governs basis vectors in each block so that they are similar to each other and the resulting convex cones are compact. λ and η control the amount of the regularization.

The main difference between USM and the proposed MLD model comes from the fact that the former sets block sparsity on speakers. It selects some relevant speakers in a global fashion, so the chosen ones are always active regardless of the time index t whereas the proposed method selects the participating blocks dynamically. Therefore, USM does not have the index t in the second term. Since it is not guaranteed that each speaker is assigned to a cluster, the data-driven way we choose is more suitable for modeling the general human speech with the limited number of clusters. After all, we expect that MLD can sort out the similar sound components into the same block regardless of who speaks them, and then eventually approximate the data more precisely.

Another thing that makes MLD unique is the newly introduced third term. For each block g we can have *a priori* knowledge about the bases, which can be learned beforehand by using any clustering techniques, e.g. K-means clustering. When the algorithm tries to reduce the error in the third term, the bases are more likely to be similar to the *a priori* information. This will also result in more concentrated solutions. Note that the regularization works as a conjugate prior in the corresponding probabilistic models, such as Dirichlet priors in PLSI [30].

After majorizing the second term [45] we can derive some multiplicative update rules similarly to the

Algorithm 1 The dictionary learning algorithm using MLD

Input: $V \in \mathbb{R}_+^{M \times N}, G, R$
Output: W
Find G cluster means, $\mu^{(g)}$, by using K-means
Initialize $W^{(g)} \in \mathbb{R}_+^{M \times R}$ with $\mu^{(g)}$ and $H \in \mathbb{R}_+^{GR \times N}$ with random numbers
repeat
    Update W and H using (3.2), (3.3), and (3.4)
until Convergence

Algorithm 2 The speech enhancement algorithm using MLD

Input: $V \in \mathbb{R}_+^{M \times N}$, $\left\{ W_{dic}^{S\ (g)} \in \mathbb{R}_+^{M \times R_S} \middle| 1 \leq g \leq G_S \right\}$, and $\left\{ W_{dic}^{N\ (g)} \in \mathbb{R}_+^{M \times R_N} \middle| 1 \leq g \leq G_N \right\}$ (optional, semi-supervised) or R_N (optional, unsupervised)
Output: H^S, H^N
Initialize H^S and H^N with random numbers
repeat
    Update H using (3.3) and $h_t^{S(g)}$ using (3.4)
    if Unsupervised **then**
        Update W_{dic}^N using (2.9)
    else
        Update $h_t^{N(g)}$ using (3.4)
    end if
until Convergence

NMF case for all $g \in \{1, \cdots, G\}$ and $t \in \{1, \cdots, N\}$:

$$W^{(g)} \leftarrow W^{(g)} \odot \frac{\left(\frac{V}{WH}\right) H^{(g)\top} + \eta \frac{\mu^{(g)}}{W^{(g)}}}{\mathbf{1} H^{(g)\top} + \eta \mathbf{1}}, \tag{3.2}$$

$$H \leftarrow H \odot \left\{ W^\top \left(\frac{V}{WH} \right) \right\} \Big/ \left\{ W^\top \mathbf{1} \right\}, \tag{3.3}$$

$$h_t^{(g)} \leftarrow h_t^{(g)} \Big/ \left\{ 1 + \lambda \big/ \left(\epsilon + ||h_t^{(g)}||_1 \right) \right\}. \tag{3.4}$$

The MLD update rules are used to learn dictionaries from the source-specific training signals and to denoise an unseen noisy signal as in Section 2.1. Algorithm 1 shows how we learn the dictionaries. First, we prepare a big set of speech training spectra V_{dic}^S recorded by anonymous speakers, and define the number of blocks G and the number of bases R in each block. If a noise training set is available, we prepare V_{dic}^N as well. Then, we use each matrix as the input to Algorithm 1, respectively, to get W_{dic}^S and W_{dic}^N.

There are three test scenarios depending on the availability of dictionaries:

- Unsupervised: the case when neither the speaker identity nor the type of noise is known. Therefore, we learn a suboptimal dictionary from someone else' clean speech and apply the semi-supervised technique.

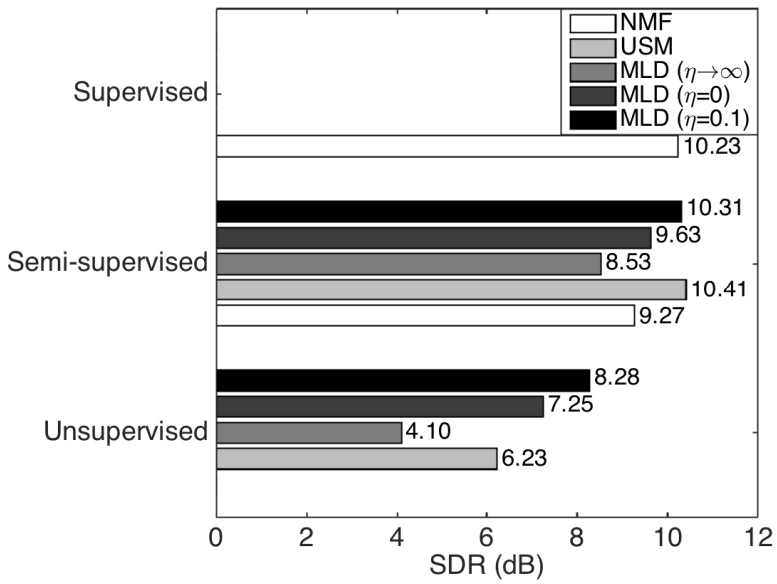


Figure 3.3: Average SDR results from three models and cases.

- Semi-supervised: in this case either speech or noise dictionary is missing. The missing dictionary is learned during the test phase.
- Supervised: both kinds of information are known and their training data are available.

MLD is mainly for the unsupervised and semi-supervised cases. In the unsupervised scenario, since we have no speaker information, Algorithm 1 learns the speech dictionary W^S_{dic} using third-party speech signals as an alternative training set. Then, the noise dictionary is learned from the mixture signal (the "Unsupervised" option in Algorithm 2) using the ordinary NMF update in (2.9). In the semi-supervised case where only the type of noise is known, we solve this as if it was a supervised case with the noise dictionary and the suboptimal speech dictionary.

3.1.2 Experimental results

We compare the results from the MLD algorithms with USM and NMF in the three denoising scenarios. All signals used are sampled at 16kHz. As for STFT, we use 1024 samples for Hann windowing and Fast Fourier Transform (FFT) with 256 pt hop size. A small value for $\epsilon = 10^{-5}$ worked fine. We randomly select 20 speakers from the training set of TIMIT corpus as our anonymous speakers, and mix 5 test speakers' speech with 10 different non-stationary noise signals that were proposed in [29] to build 50 test sequences as in USM experiments. Instead of learning 10 NMF bases from each of the 20 speakers, we learn a set of $G_S = 20$ local dictionaries, each of which holds $R_S = 10$ bases. This number of speech bases is eventually

same with the original USM setting while holding different information about the source. All USM and NMF results in this section are from the experiments in [45].

In the unsupervised case, the number of noise bases is fixed with the optimal ones investigated in [29], i.e. one of {20, 10, 200, 20, 20, 10, 20, 10, 10, 10}, depending on the noise type used. $\lambda = 64$ is big enough to yield sparse solutions. MLD outperforms USM by 2dB in terms of the average Signal-to-Distortion Ratio (SDR) [48] with an adequate $\eta = 0.1$ (the bottom bars in Fig. 3.3). Note that when $\eta \to \infty$ we get worse results, representing the case where each local dictionary is completely concentrated on its mean. Another case of interest is when $\eta = 0$, which boils down to a variation of USM where the speaker selection is done at every frame, but without the third regularization term in (3.1). We can say that another 1dB improvement is explained by the third term.

Next, we learn noise dictionaries as well with the same parameters ($G_N = G_S, R_N = R_S$), and then run Algorithm 2 without the unsupervised option. $\lambda = 2$ gives less sparse noise coding, but provides best separation results. Compared to the other semi-supervised algorithms (middle bars in the figure), the result is significantly better than NMF (1dB improvement) and comparable to USM. In this case, learning 200 basis vectors for each individual noise type is a difficult problem, and eventually does not outperform USM.

In Fig. 3.3 the result from the fully supervised NMF algorithm is provided for a further comparison. Once we are allowed to learn dictionaries for noise (semi-supervised), both USM and MLD models produce comparable results to the fully supervised NMF case, but without knowing the exact speaker.

3.1.3 Summary

In this section we proposed the Mixture of Local Dictionaries (MLD) model that preserves the underlying manifold of the data. With extensions to the Universal Speech Model (USM), such as temporally relaxed block sparsity and concentration of basis vectors, the near-disjoint combination of local dictionaries provided substantial improvement over NMF and USM in the unsupervised single-channel speech enhancement tasks with as little *a priori* information about the sources as possible. Since this assumption about the source is general enough, i.e. an English speech, while being robust to many possible variations, such as dialects, speaker identities, and genders, we believe that MLD could be a practical solution to the single-channel unsupervised speech enhancement problem.

3.2 Manifold Preserving Hierarchical Topic Models

In this section, we propose two hierarchical topic models. First, a quantization method is used to reduce the size of the overcomplete training data. Doing so is important since the overcomplete representation can often necessitate additional memory and computational resources to process larger number of parameters. Quantizing the data helps minimize redundancy in the data while retaining their expressive power. To this end, we introduce an additional latent variable that selects overcomplete candidate topics and use them to replace the entire overcomplete bases.

Second, the sparse topic model cannot always produce good source estimates especially when the training data is not dense enough, or some important part of the data is lost during the sampling procedure. To handle this issue, we propose another middle-layer latent variable, which is also dedicated to activate only selected data points: groups of neighboring data points of the current estimation of sources. That will result in more manifold-preserving reconstructions.

3.2.1 The proposed hierarchical topic models

Although the proposed extensions of PLSI have different applications, both the manifold preserving quantization and interpolation share some structural similarity: an additional latent variable that weeds out unneeded topics during the analysis.

Suppose that we have some observations $X_{f,t}$. We might need to learn both $P(f|z)$ and $P_t(z)$ for training, or can fix the provided $P_s(f|z)$ and learn the encodings only. In the proposed hierarchical models, we first seek a more compact representation of $P(f|z)$ by additionally decomposing them with a new latent variable y as follows:

$$X_{f,t} \sim \sum_y \sum_z P(f|z)P(z|y)P_t(y). \tag{3.5}$$

Hence, we can say that the linear transformation of topics, $\sum_z P(f|z)P(z|y)$, is a selection process once the selection parameters $P(z|y)$ meet certain criteria.

Manifold preserving quantization

The goal of manifold preserving quantization is to represent the input data with smaller number of samples, each of which can play as a representative topic that well respects locality of the data. Usually, this

Part of Section 3.2 has been published in [44].

quantization is to replace the overcomplete training data or their convex hull with smaller number of representatives on the manifold.

First of all, we use the input observation vectors $X_{f,t}$ as our topic multinomials $P(f|z)$ in (3.5) as they are as our topic multinomials as they are:

$$X_{f,t} \sim \sum_{y} \sum_{t'} X_{f,t'} P(t'|y) P_t(y), \tag{3.6}$$

where the new index t' is surely for column vectors of X as well, but introduced to distinguish from the observation indexing as they have a new usage, i.e. fixed topics. And then, we assume the selection parameter $P(z|y)$ has a smaller number of values of y than that of z, so that the selection procedure $\sum_{t'} X_{f,t'} P(t'|y)$ produces less samples than the inputs[2]. Furthermore, assumptions about sparsity of $P(t'|y)$ and $P_t(y)$ along t' and y axes, respectively, can let the learning results respect the manifold structure. For the sparsest case, assume that only t'-th element of $P(t'|y)$ is one while the others are zero. The only activation chooses an input vector as y-th representative sample. After getting the reduced number of topics $P(f|y)$ like this another sparsity constraint on y also forces each topic to represent as many surrounding inputs as possible by itself.

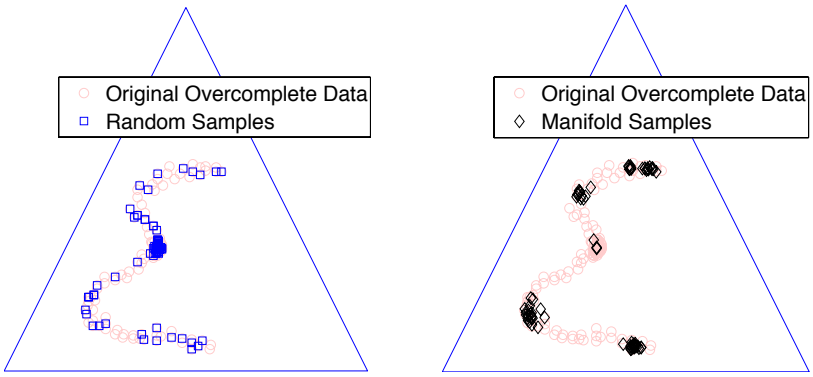
For the inference of the hierarchical latent variable model, we follow the conventional EM approach, but for each layer sequentially. However, for this particular quantization model, we can skip the first layer EM as the topical parameter $P(f|z)$ is substituted and fixed with the input vectors $X_{f,t'}$, and the other parameter $P_t(t')$ can be trivially reconstructed with the second layer parameters, $P_t(t') = \sum_y P(t'|y) P_t(y)$.

The second layer expected complete data log-likelihood $\langle \mathcal{L} \rangle$ for y is:

$$\begin{aligned}
\langle \mathcal{L} \rangle = & \sum_{f,t,t',y} X_{f,t} P_t(t', y|f) \Big\{ \ln P(t'|y) + \ln P_t(y) \Big\} \\
& + \gamma_1 \sum_{t'} \phi_{t'} \ln P(t'|y) + \gamma_2 \sum_{y} \theta_y \ln P_t(y) \\
& + \lambda_1 \{1 - \sum_{t'} P(t'|y)\} + \lambda_2 \{1 - \sum_{y} P_t(y)\} \\
& + \text{Constants},
\end{aligned} \tag{3.7}$$

where Lagrange multipliers λ_1 and λ_2 for ensuring parameters to sum to one are straightforward. The sparsity constraint terms $\gamma_1 \sum_{t'} \phi_{t'} \ln P(t'|y)$ and $\gamma_2 \sum_y \theta_y \ln P_t(y)$ are from Dirichlet priors with sparse parameters ϕ and θ, and γ_1 and γ_2 control the contribution of the sparsity terms in the objective function. One

[2]Note that we get a trivial solution when we set the same number of y as t', i.e. $P(t'|y)$ and $P_t(y)$ being identity matrices.



(a) (b)

Figure 3.4: The repeatedly (20 times) sampled 4 bases $P(f|y)$ on an ε shaped manifold with (a) random sampling (b) proposed sampling.

way to introduce sparsity to the Dirichlet hyper parameters is to substitute them with raised parameters,

$$\phi_{t'} = P(t'|y)^{\alpha}, \qquad \theta_y = P_t(y)^{\beta}$$

where α and β are some values bigger than 1, and parameters are the estimations from the previous EM iterations. This sparse priors can be intuitively understood, because, for instance, the L_2 norm (when $\alpha = \beta = 2$) of a p.d.f. is maximized when only one value of the variable has probability 1 while the others are 0. Generally, basing on the fact that the parameters have the same L_1 norm, α and β bigger than 1 can make the parameters sparser.

The second layer E-step for y is:

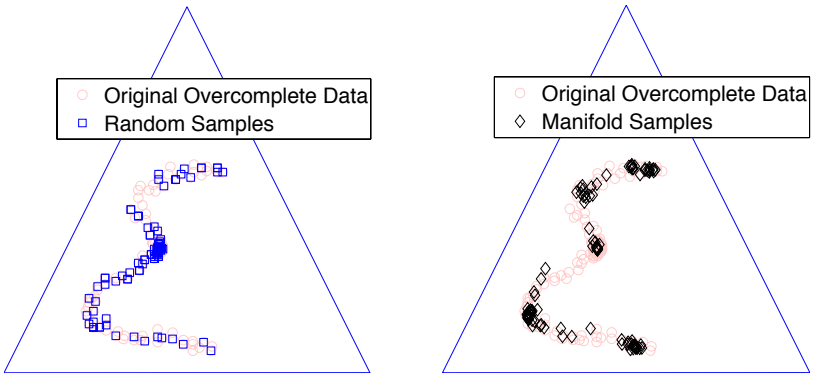
$$P_t(y, t'|f) = \frac{X_{f,t'} P(t'|y) P_t(y)}{\sum_{t'} X_{f,t'} \sum_y P(t'|y) P_t(y)}.$$

In the second layer M-step we find the solutions that make the partial derivatives of $\langle \mathcal{L} \rangle$ zero, which in turn become update rules as follow:

$$P(t'|y) = \frac{\sum_{f,t} X_{f,t} P_t(t', y|f) + \gamma_1 P(t'|y)^{\alpha}}{\sum_{f,t,t'} X_{f,t} P_t(t', y|f) + \gamma_1 P(t'|y)^{\alpha}}, \quad P_t(y) = \frac{\sum_{f,t'} X_{f,t} P_t(t', y|f) + \gamma_2 P_t(y)^{\beta}}{\sum_{f,t',y} X_{f,t} P_t(t', y|f) + \gamma_2 P_t(y)^{\beta}}. \tag{3.8}$$

Note that the hyper parameters are replaced at every iteration with the raised previous estimations.

Figure 3.4 shows the sampling results on ε-shaped manifold. The input exhibits a lower number of data points on the wings as opposed to a higher concentration in the middle. Figure 3.4 (a) shows the 80 random



(a) (b)

Figure 3.5: The repeatedly (20 times) sampled 5 bases $P(f|y)$ on a ε shaped manifold with (a) random sampling (b) proposed sampling.

samples, which consist of four samples that are repetitively drawn 20 times. From the random sampling result (blue squares), we observe that it is very possible to have the all four samples from the center, where the population is highest, but rarely from the wings.

On the other hand, in Figure 3.4 (b) with the proposed quantization, the four representatives (black diamonds) tend to lie on the two elbows and the two tips, which crucially explain the manifold. They can at least form a c-shape by ignoring the kink while naïve four samples all from the populous central kink give no shape information. Note that only three out of 80 are sampled from the center using the proposed quantization.

However, it is obvious that the fifth sample should be from the kink to complete the full ε-manifold. Figure 3.5 (b) gives the desired result where the fifth sample successfully represents the central kink, while in Figure 3.5 (a) the randomly sampled five do not provide such a well-structured quantization results.

Manifold preserving interpolation

Although the proposed manifold preserving quantization provides a compact representation, the original data points that were in-between those samples might not be modeled as accurately as with the overcomplete data plus sparse PLSI case. Moreover, it is also possible that the original training data might not be as dense in the first place.

Figure 3.6 describes this situation. Let us assume that we discarded all data points (pink circles) after quantizing them with only five samples (filled or empty diamonds). Given a mixture point (blue cross) and an already estimated source #1 for simplicity, the goal of the sparse PLSI is to select the best one out of the five samples, which eventually reconstructs the mixture with the reverse triangle. On the other hand,

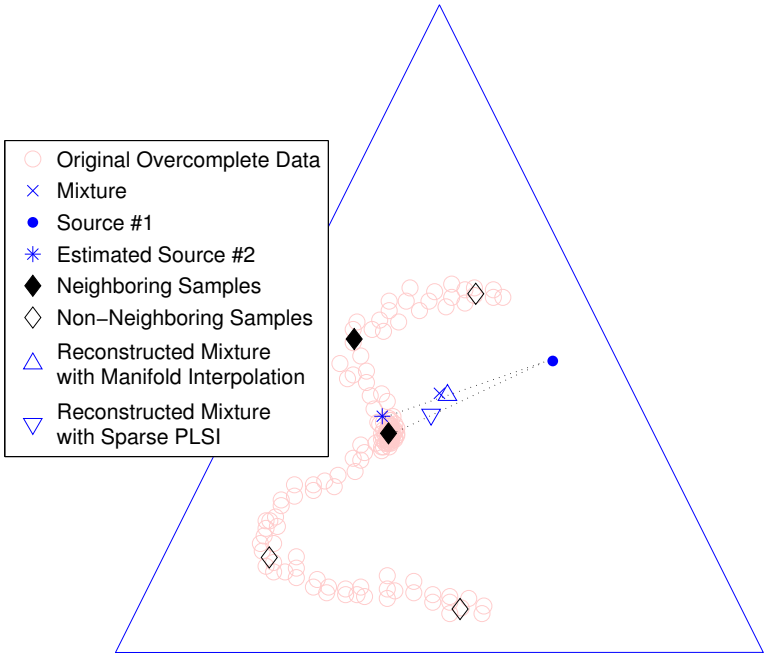


Figure 3.6: An illustration about the drawback of coupling manifold quantization and the sparse PLSI method. The proposed interpolation method resolves the issue by a local linear combination of samples.

the proposed manifold preserving interpolation seeks a linear combination of neighboring samples (filled diamonds), which provides an interpolation for the missing data between the samples (blue star). Note that the estimation of mixture with this approach (triangle) is closer to the input mixture than that from the quantization only.

We start from the same hierarchical topic model introduced in (3.5). For the t-th mixture vector $X_{f,t}$ the goal is to reconstruct it by combining a few neighbors:

$$X_{f,t} \sim \sum_{s} \sum_{z \in \mathcal{N}_t^s} P_s(f|z) P_t(z|s) P_t(s),$$

where s indicates the sources and $\mathcal{N}_t^s$ is the set of neighboring samples of s-th source estimation for t-th input (the filled diamonds in Figure 3.6). The selection parameter $P_t(z|s)$ now has the index t to provide weights for each set of neighbors per an input as in [49]. On the contrary to the previous quantization method, $P_s(f|z)$ is fixed to hold either the overcomplete training data or quantized samples of source s as a set of topics.

Similarly to the previous derivation, we also skip the first layer EM, since $P_s(f|z)$ is fixed, and marginalization of the second layer variable s is trivial.

The second layer complete data log-likelihood for t-th input $\langle \mathcal{L}_t \rangle$ is defined as follows:

$$\langle \mathcal{L}_t \rangle = \sum_{f,z,s} X_{f,t} P_t(z,s|f) \Big\{ \ln P_t(z|s) + \ln P_t(s) \Big\} + \lambda_1 \{1 - \sum_z P_t(z|s)\} + \lambda_2 \{1 - \sum_s P_t(s)\} + \text{Constants}, \tag{3.9}$$

where λ_1 and λ_2 are Lagrange multipliers for the sum to one constraint as usual. We get the posterior probabilities $P_t(s,z|f)$ from the second layer E-step:

$$P_t(s,z|f) = \frac{P_s(f|z) P_t(z|s) P_t(s)}{\sum_s P_t(s) \sum_{z \in \mathcal{N}_t^s} P_s(f|z) P_t(z|s)}. \tag{3.10}$$

In the M-step, we find the parameters that maximize $\langle \mathcal{L}_t \rangle$ as follows:

$$P_t(z|s) = \frac{\sum_f X_{f,t} P_t(s,z|f)}{\sum_{f,z} X_{f,t} P_t(s,z|f)}, \quad P_t(s) = \frac{\sum_f X_{f,t} \sum_{z \in \mathcal{N}_t^s} P_t(s,z|f)}{\sum_f X_{f,t} \sum_s \sum_{z \in \mathcal{N}_t^s} P_t(s,z|f)}. \tag{3.11}$$

It is obvious that the update rules of the proposed interpolation method eventually become analogous to those of sparse PLSI in (2.13), but the difference of defining the selection parameter $P_t(z|s)$ makes the proposed method behave uniquely. Instead of imposing sparsity on the completely defined parameter $P_t(z|s)$ for all z indices, the neighbor set $\mathcal{N}_t^s$ lets the procedure focus only on the current neighbors. In other words, $P_t(z|s)$ is not smooth as it is zero for $z \notin \mathcal{N}_t^s$, and so is $P_t(s,z|f)$ for $z \notin \mathcal{N}_t^s$ in the M-step, consequently. The smaller the number of neighbors is, the more local the reconstruction is. Likewise, in the proposed interpolation model sparse coding is achieved by finding running neighbors at every iteration, which are K-nearest samples from the current estimation of each source for t-th input, $P_t(f|s) = \sum_{z \in \mathcal{N}_t^s} P_s(f|z) P_t(z|s)$:

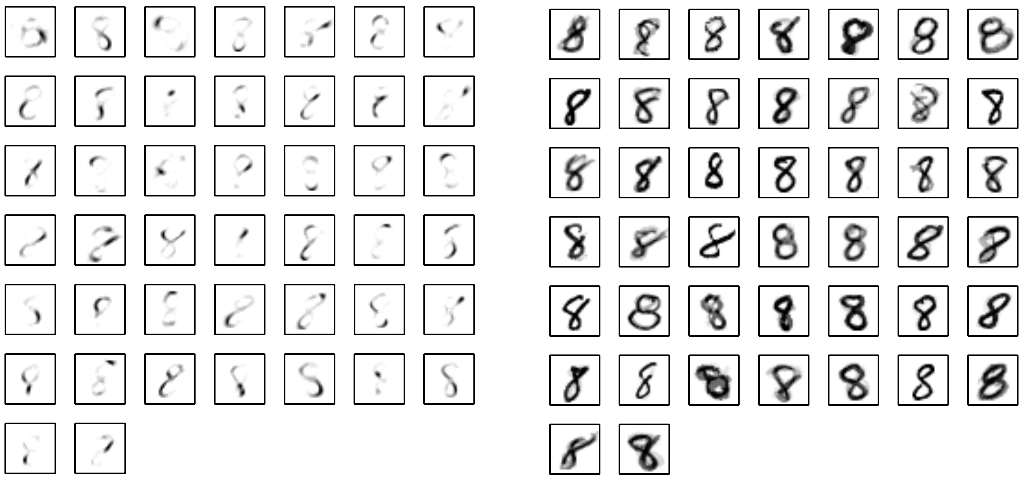
$$\mathcal{N}_t^s = \Big\{ z_k : \mathcal{E}\Big[P_s(f|z_k) \big\| P_t(f|s) \Big] < \mathcal{E}\Big[P_s(f|z' \notin \mathcal{N}_t^s) \big\| P_t(f|s) \Big] \Big\}, \tag{3.12}$$

where the integer index $1 \leq k \leq K$, and z' indicates all the possible topics. $\mathcal{E}[A\|B]$ can be any divergence measure, but we use cross entropy, which is a natural choice in the simplex domain,

$$\mathcal{E}[A\|B] = -\sum_i A_i \log B_i. \tag{3.13}$$

Computational complexities

Each EM iteration for a mixture vector X_t^M of the sparse PLSI model (2.13) runs in time $\mathcal{O}(SFZ)$, where S, F, and Z, stand for the number of sources, features, and topics. Therefore, reducing Z to a small set



(a) (b)

Figure 3.7: Comparison of probabilistic topics and manifold preserving samples. (a) 44 basis topic multinomials learned from ordinary PLSI. (b) 44 manifold samples drawn from the proposed quantization.

of manifold samples rZ with sampling rate $r < 1$ can mitigate the computational cost of the separation procedure. The quantization can be done beforehand, so its complexity is negligible.

The interpolation method can further reduce rZ to the size of neighbors K. Finding neighbors using (3.12), which is a sorting operation with $\mathcal{O}(rZ \log rZ)$, is usually a lot less complex than $\mathcal{O}(SFrZ)$ with small sampling rate r. However, since calculating the error function (3.13) requires additional $\mathcal{O}(SFrZ)$, the complexity of the manifold interpolation is $\mathcal{O}(SFrZ)$, not $\mathcal{O}(SFK)$.

3.2.2 Empirical results

Quantization of hand-written digits

The first experiment is to show the behavior of the quantization model at a sampling rate is 5% – 44 samples out of 876 images of hand-written digit, "8", from MNIST dataset [50]. For the comparison, we learned the same number of ordinary PLSI topics, which are the corners of the convex hull that wraps the images in the digit "8" class.

Figure 3.7 presents the results. As we can expect, and reported in [21], ordinary topic models without the concept of sparsity give a parts-based representations of the data, which can be seen as building blocks to be additively combined to reconstruct the input data. It is intuitive as the corners of the convex hull that surround the data points would be more likely to be near the margins, edges, or corners of the simplex, where more elements are suppressed than around the middle of the simplex, so that only several entries of the topic are activated. That is why we see some strokes of the digit "8" in Figure 3.7 (a) rather than holistic

representations.

Although the parts-based representation encourages the model to flexibly combine the topical bases, carefully quantized samples can be better representatives of the data, especially when the original data points lie on high-dimensional manifolds. Figure 3.7 (b) clearly shows the difference of the proposed sampling method from the results in (a). If we use the samples as the topics of sparse PLSI, which would be sparsely activated to recover unseen inputs, the model can confine the estimation in the manifold of the training data.

Interpolation for classifying handwritten digits

If the quantization is successful, it can be used instead of the whole training dataset or its convex hull. In this section we employ the proposed manifold interpolation method to recover the missing data between neighboring samples. First, given the 10 digit groups we do classification with 10-fold cross validation. Each class has around 1,000 images. We learn manifold samples from each class at different sampling rates with parameters set to: $\alpha = \beta = 1.2$ and $\gamma_1 = \gamma_2 = 0.001$. For a test handwritten digit, we reconstruct it using the proposed manifold preserving interpolation with several pre-defined number of neighbors[3]. Therefore, the class, where the test vector is best approximated in terms of cross entropy, is assigned as the estimated class label.

For the comparison, we also conducted a K-Nearest Neighbor (KNN) classification, which uses cross entropy as its divergence measure. During KNN classification, we also consider manifold samples as our training data. For instance, the bars in the back in Figure 3.8 (a) is the case of 100% sampling rate, where we do the ordinary KNN with the whole training data. We can first check that the proposed quantization performs better than or comparable to (88.2% at sampling rate 1%) the case of using whole data (less than 85%) if we carefully set the number of neighbors.

However, with interpolation we can generally get better classification accuracy, which ranges between 90 to 95% as in Figure 3.8 (b). Furthermore, we can also resolve the issue of sensitivity to the number of neighbors at the low sampling rates by tying up the neighboring samples to reconstruct the input rather than choosing the best one from here and there. Note that there are cases when the number of neighbors is bigger than that of samples (front-left bars with zero height).

[3]Note that in this case we assume the test input is not a mixture of multiple classes. Hence, we do the EM updates and decision of neighbors for each class separately by fixing s in (3.10), (3.11), and (3.12).

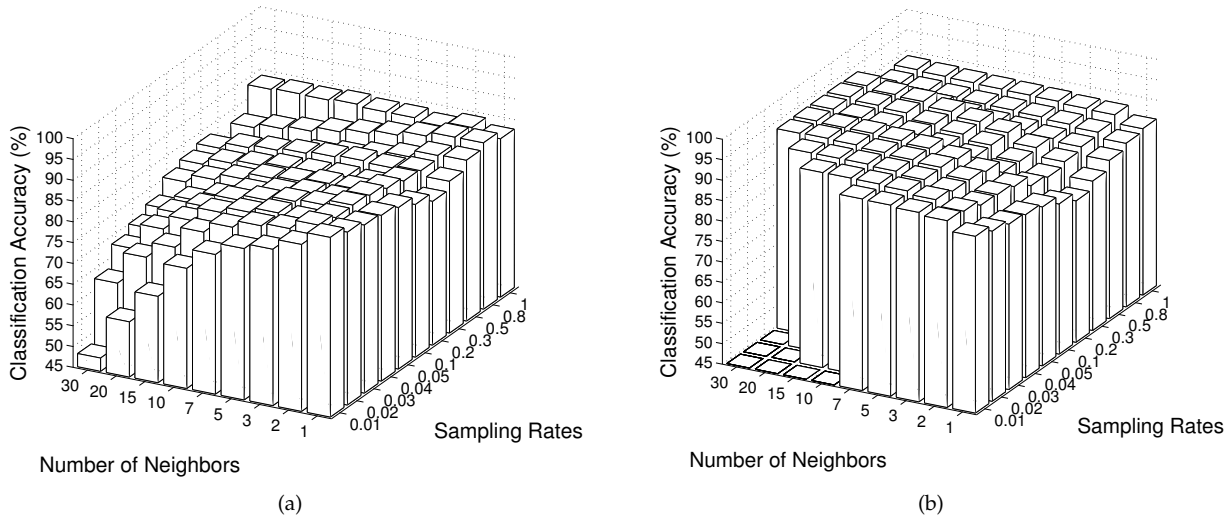


Figure 3.8: Handwritten digits classification results with (a) KNN and (b) the proposed interpolation method.

Quantization of speech signals

We further discuss the advantage of the proposed quantization by using speech signals. In this experiment a female speaker is selected from TIMIT speech corpus [51]. Spectrums of a concatenated nine spoken sentences, each of which is 2 to 3 seconds-long, are used as overcomplete training data. The concatenated training signals are converted into the matrix forms, i.e. spectrograms, by using magnitudes of short-time Fourier transform, with 64 ms window size and 32 ms overlaps. Therefore, the training matrix consists of 832 spectra (column vectors), each of which has 513 frequency elements. Now we use the spectrogram matrix X as input vectors to the manifold preserving quantization system. Moreover, we use X as the parameter $P(f|z)$ as they are, and fix them during the process.

Figure 3.9 shows the sum of the reconstruction errors in terms of cross entropy for the three different systems at seven different sampling rates. For example, when the sampling rate equals to 1%, the number of samples is $8 \approx 0.01 \times 832$. The proposed method produces the samples as a form of sparse linear combination of the whole training points using the selection parameter, $\sum_{t'} X_{f,t'} P(t'|y)$, but provides reconstructions of the input X at the same time as in (3.6). For the parameters, α, β, γ_1, and γ_2, we once again use the same values as in the MNIST experiment. We also randomly choose the same number of samples for comparison. After the random sampling, we use the closest sample for each input column as an oracle reconstruction. Lastly, ordinary PLSI learns same number of topics with the samples. As this is not an experiment for the separation, PLSI's convex hull covers the largest area, so that it provides the best pos-

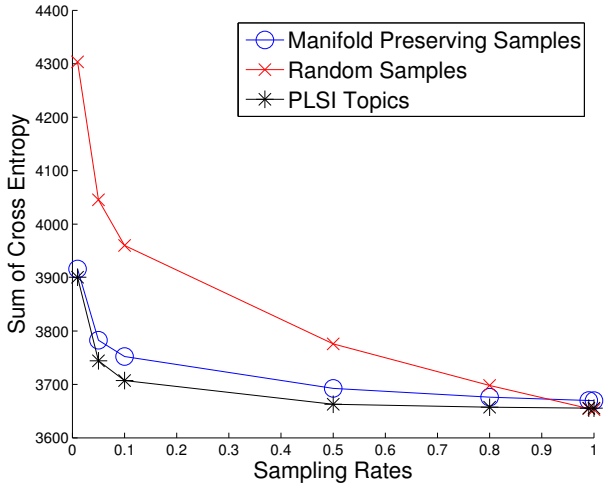


Figure 3.9: Sum of cross entropy between inputs and the reconstructions from the proposed quantization, oracle random samples, and ordinary PLSI.

sible reconstruction among the three systems. Note that PLSI does not work well as such, if the inputs are mixture of more than two sources (speakers) as shown in Figure 2.1. Experiments are repeated 10 times to average out the variance of sampling results.

In the figure, the proposed sampling method can provide representative samples, which recovers input vectors better than the manually chosen closest random samples. All the three methods provide better representation (less cross entropy) as the sampling rate increases, but the proposed sampling provides good performance at low sampling rates, which are better than those of oracle random samples. Also, its results are generally comparable to PLSI topics, which basically can recover the whole convex hull.

Separation of crosstalk using interpolation

We introduce additional male speaker to build up the crosstalk cancellation problem, on top of the speech from the female speaker. One sentence per a speaker is picked up, and then mixed up. We learn manifold samples at various sampling rates from the spectra of other 9 training sentences of each speaker. They play the role of two sets of pre-learned topics $P_{s=\text{female}}(f|z)$ and $P_{s=\text{male}}(f|z)$. After the EM updates in (3.10) and (3.11) plus the neighborhood search (3.12), we get the converged posterior probabilities $P_t(s,z|f)$, where z is marginalized out to finally get the source-specific posterior probabilities $P_t(s|f)$. For the given t-th input mixture $X^M_{f,t}$, the source s can be recovered by multiplying the resulted posterior, $X^M_{f,t}P_t(s|f)$, and then converting back to the time domain.

Figure 3.10 shows the separation performance in terms of Signal-to-Interference Ratio (SIR) [48], whose

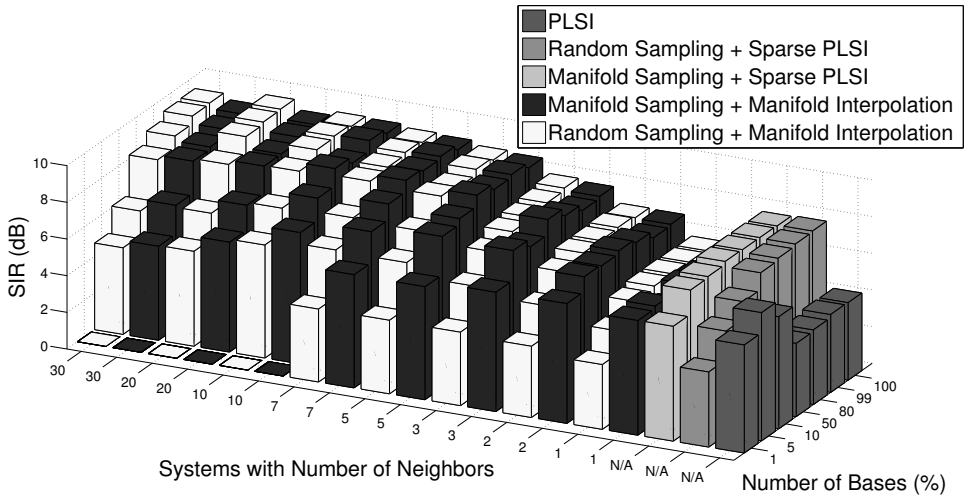


Figure 3.10: SIR of the crosstalk cancellation results with the proposed quantization and interpolation method compared with random sampling, sparse PLSI, and ordinary PLSI.

value is zero when the energies of interfering source and the recovered source are same, and can be infinity if the source estimation is perfect. All the algorithms were iterated 100 times. First of all, we use standard PLSI by using the concept in 2.2.1, but without the sparsity constraint. We set the number of topics, i.e. the corners of PLSI convex hulls, to match the sampling rate. In the shown figure (the right most bars), we can see that PLSI does not give good results with many topics, but its performance is the best (7.0 dB) at around 5% sampling rates (42 topics). On the contrary, sparsity constraints improve the results by around 1 dB (the second and third rightmost bars). It is also noticeable that manifold samples that are only 5% of the entire training data provide almost same separation performance, while random samples start to lost representativeness below 50%.

Both random and manifold samples along with the interpolation techniques further enhance the sparse PLSI results. However, it is also observed that manifold preserving samples are better than random samples at lower sampling rates when they are coupled with interpolation (three frontal bars of manifold interpolation cases). Although as the sampling rate gets higher the merit of manifold sampling vanishes, manifold-preserving interpolation plays a role for the better separation performances (up to about 9.5dB) than both ordinary and sparse PLSI.

3.2.3 Summary

In this work we proposed a manifold preserving quantization method. By adding a latent variable to the common probabilistic topic model for selecting representatives of overcomplete input, and trying to reduce the reconstruction error at the same time, the method could give better way of compressing the

high-dimensional overcomplete data. We showed that the manifold quantization can replace the whole dataset with acceptable loss of approximation power, but with better image classification and speech separation performances compared with existing topic models. On top of that, another model with the explicit neighborhood selection is proposed to compensate the quantization error. This local interpolation technique further improve the classification and separation results whether it is applied to the sampled data or the original entire observations that sometimes does not fully contain the manifold structure of the data.

3.3 Manifold Preserving Source Separation: A Hashing-Based Speed-Up

A disadvantage of this manifold consideration is the requirement of larger training data sets for robust local reconstruction. Retaining a large number of training samples, instead of the convex hulls defined by their simplicial corners only, makes learning computationally heavy and demands a larger amount of memory. These issues were addressed by hierarchical topic models in [44]. In this model, a middle-layer variable was presented to divide the model into two parts: local selection (the lower level) and global approximation (the higher level). In the local selection part, the overcomplete dictionary elements are multiplied with the sparsely encoded middle-level selector variable to produce a set of *hyper topics*, each of which represents a participating source spectrum. The hyper topics are then combined to approximate the mixture input mimicking the audio mixing procedure. Although this model provides a more direct and convenient way to couple manifold learning and topic modeling, its sparse coding step on the selector variable still demands a lot of computation.

In this section, we propose a technique to integrate hashing with manifold preserving source separation, where hashing speeds up the sparse encoding of the middle-level selector matrix. To the best of our knowledge, this is the first work to examine the use of hashing for audio topic models. To this end, we adopt WTA hashing [37], a particular type of locality sensitive hashing [39], which has shown to be efficient in image search [38]. Similar to the usage in image search, we hash the dictionary elements to promptly provide a candidate subset that is much smaller than the entire database, so that the subsequent exhaustive search can only focus on this reduced set. During this construction of the candidate set, we rely on the fast and efficient Hamming distance calculation on hash code bits where the speed-up happens. On the other hand, it is not straightforward to apply this hash codes matching technique to the source separation procedure

Part of Section 3.3 has been published in [52].

directly. In the case of image search, the goal is to match a query image to the closest matching example images. In the case of audio source separation however, the goal is to simultaneously match a mixture spectrum to heterogeneous sets of clean spectra from different sources. Our key contribution is developing a technique to do this. Another important advantage of employing the hashing technique is that its cheap fixed-point operations can extend the applicability of the topic model based audio analysis techniques to the devices with more restricted conditions.

3.3.1 WTA hashing for manifold preserving source separation

We saw in Section 3.2 that we can respect the manifold of the data during topic modeling by allowing only a small number of local neighbors to participate in the reconstruction of the hyper topics. Sparsity on the selection parameter $P_t(z_y|y)$ is critical for this procedure, but sparse selection on a large number of training samples, Z_y, primarily accounts for the computational complexity of the algorithm. One naïve approach to using hashing in order to reduce this complexity is to replace the active set of topics, i.e. the nearest neighbors $\mathcal{N}_y^t$, with the ones with lowest Hamming distance to the hyper topics. However, due to the mismatch between the approximated rank ordering measure and the original cross entropy, it can cause inaccurate results.

Instead of solely relying on Hamming distance as our distance metric, we use WTA hashing as a pre-processing step. After the hashing part reduces the search space from Z_y to N topics, we perform a K nearest neighbor search on these reduced N topics to refine the results. The key idea is to keep an up to date set of candidates $\mathcal{Z}^{N\times Y}$ whose y-th column vector holds the indices of N closest candidates to the y-th hyper topic in terms of the Hamming distance. If we set $K < N \ll Z_y$, the final estimation of $P_t(z_y|y)$ requires consideration of only N elements rather than the probabilities of the entire Z_y topics. Unless N is too small to include the K important topics as candidates, or the Hamming distance defined on the WTA hash codes is significantly different from the original distance measure, some spurious candidates included in the candidate set should not be of significant consequence. In other words, the exhaustive nearest neighbor search that follows the hashing step is able to pick out the final nearest neighbors anyway with or without hashing, but a proper hashing results can speed up this by providing good candidate solutions.

Algorithm 3 describes the separation procedure assisted by WTA hashing. In there, we use notation $A_{:,i}$ to indicate i-th column of the matrix A. In Algorithm 3 each source has its own set of training samples that are indexed by z_y, and hashed in advance (line 4 to 6). Then, we separate each t-th mixture frame independently. In order to conduct simultaneous hash code matching on multiple source dictionaries with an unseen mixture spectrum only, at every iteration we do a tentative separation first, and then do hashing

Algorithm 3 Manifold preserving source separation with WTA hashing

1: Initialize a permutation table $\mathcal{P} \in \mathbb{R}^{L,M}$.
2: Initialize dictionaries $P(f|z_y)$ with source-specific training spectra or their quantized versions.
3: Define the parameters N and K with inequalities $K < N < Z_y$.
4: **for** $y \leftarrow 1$ to Y and $z_y \leftarrow 1$ to Z_y **do**
5: $\quad C_{:,z_y,y} \leftarrow$ WTA_hash($P(f|z_y)_{:,z_y}, \mathcal{P}$)
6: **end for**
7: **for** $t \leftarrow 1$ to T **do**
8: $\quad$ Initialize $P_t(f|y), P_t(z_y|y)$ and $P_t(y)$ with random numbers, and normalize to sum to one.
9: $\quad$ **repeat**
10: $\quad\quad$ **for** $y \leftarrow 1$ to Y **do**
11: $\quad\quad\quad c \leftarrow$ WTA_hash($P_t(f|y)_{:,y}, \mathcal{P}$)
12: $\quad\quad\quad$ Find a set of N candidate topics for y-th source, $z_y \in \mathcal{Z}_{:,y}$, with least Hamming distance to c: $Hamming(c, C_{:,z_y,y})$.
13: $\quad\quad\quad \mathcal{N}_y^t \leftarrow \left\{ z_y \middle| z_y \in \mathcal{Z}_{:,y}, \quad \mathcal{E}\left[P(f|z_y)\middle\|P_t(f|y)\right] < \mathcal{E}\left[P(f|z'_y \notin \mathcal{N}_y^t)\middle\|P_t(f|y)\right] \right\}$
14: $\quad\quad\quad$ **for all** $f \in \{1 \cdots F\}, z_y \in \mathcal{N}_y^t$ **do**
15: $\quad\quad\quad\quad P_t(z_y, y|f) \leftarrow \frac{P(f|z_y)P_t(z_y|y)P_t(y)}{\sum_{z_y,y} P(f|z_y)P_t(z_y|y)P_t(y)}, \quad P_t(z_y|y) \leftarrow \frac{\sum_f X_{f,t} P_t(z_y,y|f)}{\sum_{f,z_y} X_{f,t} P_t(z_y,y|f)},$
$\quad\quad\quad\quad P_t(y) \leftarrow \frac{\sum_{f,z_y} X_{f,t} P_t(z_y,y|f)}{\sum_{f,y,z_y} X_{f,t} P_t(z_y,y|f)}, \quad P_t(f|y) \leftarrow \sum_{z_y \in \mathcal{N}_y^t} P(f|z_y) P_t(z_y|y).$
16: $\quad\quad\quad$ **end for**
17: $\quad\quad$ **end for**
18: $\quad$ **until** Convergence
19: **end for**

with the source estimates. First, current source estimates $P_t(f|y)$ (randomly initialized vectors at the first iteration) are hashed to be compared with their corresponding dictionaries (line 11). Once again, Hamming distance is used to speed up this comparison by taking advantage of the lower complexity of bit-pattern comparisons (line 12). The learned N candidates per each source are used to reduce the size of the subsequent nearest neighbor search at every EM iteration. The reduced search is only on those $N \ll Z_y$ candidates (line 13). The actual EM updates are not affected by this procedure, since they are already defined by excluding non-neighbors (line 14 to 16). The proposed harmonization of hashing and the topic model relies on the reconstruction of the normalized source spectra $P_t(f|y)$ at every EM iteration (line 15). It could be a tentative solution to the separation problem before the convergence, but at the same time it serves as a query at the next iteration to update the neighbor sets: the candidate set $\mathcal{Z}$ and the nearest neighbors $\mathcal{N}_y^t$.

Computational complexity

The complexity of WTA hashing for a given vector with length F is $\mathcal{O}(LM)$. Hence, the entire hash code generation for the overcomplete dictionary (line 4 to 6) runs in $\mathcal{O}(LMZ_yY)$. However, it can be ignored, because it is a one-time procedure that can be done in advance during training. Since EM updates (line 15) are still on the compact set of K final nearest neighbors, their complexity $\mathcal{O}(FKY)$ remains same. The

actual speed-up happens in line 13 where we do the cross entropy based nearest neighbor search, but now on a much reduced set with only N candidates rather than the entire training spectra Z_y. Therefore, the complexity of line 13 is $\mathcal{O}(FNY)$, which reduces original complexity $\mathcal{O}(FZ_yY)$ if $N < Z_y$.

Hashing introduces additional complexity, but it is still less complex than $\mathcal{O}(FNY)$. WTA hashing for Y hyper topics (line 11), calculation of Hamming distance between the hyper topics and the training data, and construction N candidates (line 12), run in $\mathcal{O}(YLM)$, $\mathcal{O}(YLZ_y)$, and $\mathcal{O}(Z_yN)$, respectively. However, because usually the size of the permutation table is small, $L < F$ and $M < N$, in turn $\mathcal{O}(YLM) < \mathcal{O}(FNY)$. We can also disregard the Hamming distance calculation with $\mathcal{O}(YLZ_y)$ since it can be run with cheap bit-operations. Therefore, the complexity of each iteration for the source y is governed by $\mathcal{O}(Z_yN)$ or $\mathcal{O}(FNY)$ depending on the inequality between Z_y and FN, while neither of them is more complex than the original $\mathcal{O}(FZ_yY)$ since usually $N < Z_y$ and $N < FY$.

3.3.2 Numerical experiments

In this section we compare several models discussed so far:

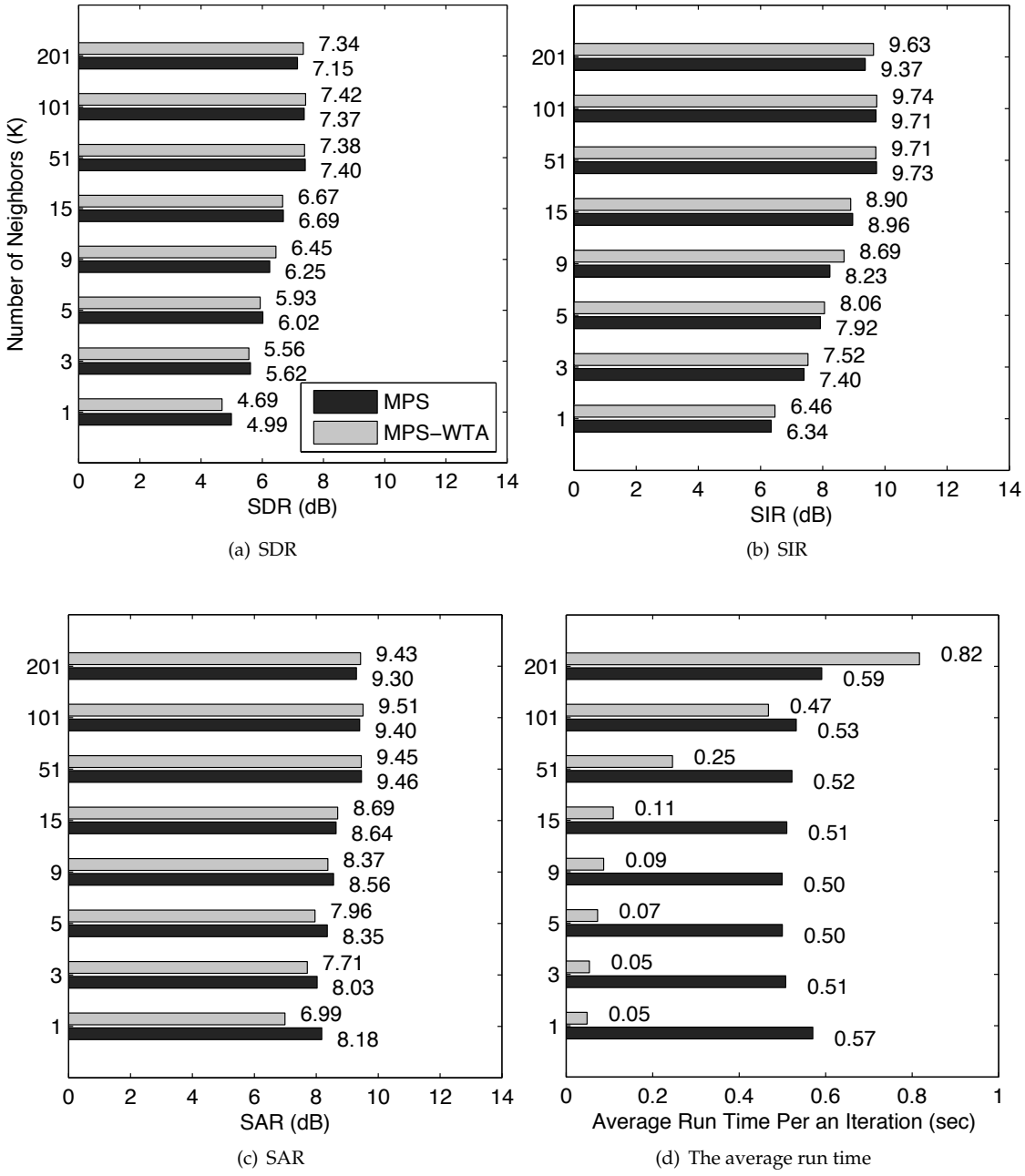
- Sparse PLSI
- The comprehensive Manifold Preserving Separation without hashing (MPS)
- The proposed Manifold Preserving Separation with WTA hashing (MPS-WTA)

We first compare the proposed MPS-WTA system with its comprehensive counterpart, MPS, to check that the proposed harmonization with hashing does not significantly reduce the cross-talk cancellation (separation of a target and interference speaker) performance. We then apply the proposed model to a more realistic scenario – speech denoising without *a priori* knowledge about neither the noise nor the speaker, which therefore requires a larger dictionary to cover the unseen speaker's characteristics.

The separation quality is measured with the standard source separation metrics [48]. The higher the SIR value, the greater the separation. However, more separation tends to introduce more artifacts, which decrease the Signal-to-Artifact Ratio (SAR). SDR is an overall measurement of both separation and artifacts. Throughout the experiments, the size of the candidate set N is set to be proportional to that of $\mathcal{N}_y^t$, i.e. $N = 5K$. A permutation table $\mathcal{P}$ is defined with $L = 100$ and $M = 4$, and shared among all hash executions.

Cross-talk cancellation

For the cross-talk cancellation experiment, we first concatenate nine random sentences per a TIMIT speaker as our training data. Each concatenated signal is then transformed into a matrix using STFT with 64 ms



(a) SDR

(b) SIR

(c) SAR

(d) The average run time

Figure 3.11: The average cross talk cancellation results of ten random pairs of speakers by using the comprehensive MPS and its hashing version, MPS-WTA, in terms of (a) SDR (b) SIR (c) SAR. (d) Average run time of individual iterations. We implemented the algorithms with MATLAB® and ran them in a desktop with 3.4 GHz Intel® Core™ i7 CPU and 16GB memory.

Hann windowing and 32 ms overlap. We take the magnitudes of the matrix and normalize them to make sure the column vectors sum to one. A sentence per each speaker is set aside for testing. We randomly select a pair of male and female speaker for an experiment and mix their test sentences. We repeat this for ten different pairs. Depending on the random choices of speakers and the sentences, the number of the column vectors (spectra) in the training matrices varies from around 700 to 1,000, while the number of frequency bins is fixed to 513. We run both algorithms for 200 iterations, where we observe convergence. We tried different numbers of neighbors $K = \{1, 3, 5, 9, 15, 51, 101, 201\}$ to control the model complexity.

Figure 3.11 shows the separation results of MPS-WTA and the comprehensive MPS methods. What we want to show is that the two methods share similar degrees of separation performance. In the first three sub-figures we can see that their separation performances in terms of (a) SDR, (b) SIR, and (c) SAR, are not significantly different between the two methods. Therefore, we can conclude that MPS-WTA gives comparable crosstalk cancellation performance to MPS. It is a favorable observation for the proposed method, as it can perform the task with reduced computation, and in less average run time per an iteration as shown in Figure 3.11 (d).

From the figures, we observe that with the proposed method, we achieve a significant speed-up by reducing the number of neighbors with a modest decrease in separation performance. For instance, by reducing the number of neighbors from 51 (above which there seems to be no gain in performance) to 5 we can triple the speed, but the separation performance decreases only by about 1.5dB. When N is set to be larger than the number of the training samples, hashing does not provide with the compact candidate set anymore, but merely increases the computation with its redundant searching (see the topmost bar in (d)).

Unsupervised speech enhancement using a large training set

In the previous clause we saw that the proposed hashing technique produces a comparable quality of audio source separation results to its counterpart that does not use the hashing concept. In this section, we apply them to a more realistic and difficult audio source separation problem where we need an even larger training data set.

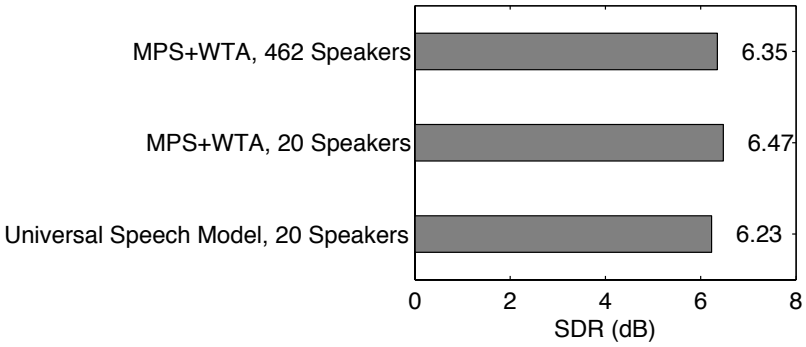
A concept called "semi-supervised source separation" was proposed in [53] to handle the situation in which we have training data corresponding to only a subset of the sources in a mixture. For speech enhancement, this can be the case in which we have isolated training data corresponding to noise, and can construct a dictionary for the noise, but not one for the speaker. On the other hand, if we do not know anything about the noise, but have isolated training data corresponding to the speaker, we can construct a speech dictionary. In general, if we have only one of the two dictionaries, the other one can be learned

during the separation from the test data. Usually we can fix a subset of topics with the known dictionary and let the others topics adapt to the rest of the signal (where the rest of the signal itself can be comprised out of multiple sources).

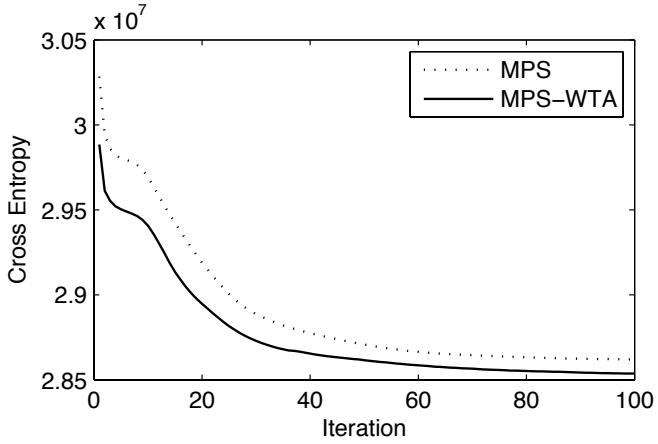
A challenging case is one in which we do not have training data corresponding to the speaker or the noise. One approach to this "unsupervised" case, is to perform semi-supervised separation using an inaccurate dictionary learned from, perhaps, another speaker. Moreover, we might also want to use a training set by gathering speech signals from as many anonymous people as possible to cover the variance of the unseen signal. However, ordinary convex hull models, such as PLSI, could fail to learn a discriminative dictionary in this case, because the learned hull could be an unnecessarily big polytope that is apt to cover spurious speech spectra as well. On top of that, the bigger the convex hull becomes, the more unwanted regions are included, which could be overlapped with a hypothetical noise hull. Therefore, this is a good example where we actually need the fast and efficient version of the MPS model. MPS is preferred, because it focuses only on the small area where the estimate is best reconstructed with only several local neighbors. Furthermore, we need hashing harmonized to it, so that it can run significantly faster than MPS.

We borrow the experimental setup proposed in [45], where five speakers from the TIMIT test set are randomly selected as the ground truth sources. They are artificially contaminated with 10 different non-stationary noise signals used in [29] with the same volume. Therefore, there are 50 different noisy signals as a test set. Because we assume that the speaker identity is not known, we prepare 20 arbitrary speakers (16,758 spectra) from the TIMIT training set that are different from the ones used for the ground truth. We use these spectra as our overcomplete dictionary input to the MPS-WTA system (the second bar in Figure 3.12 (a)). Also, the entire TIMIT training signals from 462 speakers (393,116 spectra) are also examined for the comparison (the third bar). We set $K = 15$ for these experiments. The goal of this comparison is to show that the proposed method provides comparable results to the state-of-the-art system, the universal speech model [45], but with a reduced complexity. In the universal speech model 10 topics per each random training speaker are learned, which finally amount to 200 topics for 20 speakers. The universal speech model performs comparably to the case in which isolated training data of a given speaker is available, because during the separation the model activates only a few most similar speakers' dictionaries with the help of block sparsity on the activations. Figure 3.12 (a) shows that the proposed hashing techniques are comparable to the universal speech model (the first bar) in its separation performance.

Although the computational complexity analysis already shows the merit of the hashing technique, we can further discuss the run time of the algorithms. For example, it takes only 2.5 seconds per an iteration for MPS-WTA to get the second bar in Figure 3.12 (a), while a corresponding run of MPS takes about 109



(a)



(b)

Figure 3.12: (a) Speech enhancement results of the proposed hashing method compared with the state-of-the-art system, and (b) its convergence behavior compared with the corresponding comprehensive technique that does not use hashing.

seconds[4]. This is a significant speed-up as we claim, and also noticeable considering their indistinguishable convergence behaviors reported in Figure 3.12 (b) – they converge at almost same number of iterations.

3.3.3 Summary

In this section we proposed an algorithm for hashing audio spectra that facilitates efficient learning of a manifold respecting probabilistic topic model. The model is characterized by the use of a middle-layer latent variable dedicated to learn sparse encoding of overcomplete dictionaries. Sparsity plays a big role in seeking a quality separation of audio, because it allows the only local neighbors to contribute to the solution. The proposed hashing technique further reduced the complexity of the topic model that is dependent on the number of training samples. To this end, the hashing method provided a set of candidates that can include the final nearest neighbors. In this reduced search space the topic model could achieve the separation of audio mixtures with no performance drops, but with a sensible speed-up. The proposed method was particularly useful in learning a compact representation or in quickly picking out the only relevant entries from a relatively large audio data set. Experiments on a cross-talk cancellation and an unsupervised speech denoising tasks showed the merit of the proposed method, showcasing both increased processing speed and comparable accuracy. We believe that the proposed method can be promising in devices with limited resources, and in analyzing big audio data.

[4]Core parts of both algorithms were written in C, and run on the same machine used in the previous clause.

Chapter 4

Big Audio Data Processing:
An IR System

In this chapter we will see some signal-to-signal matching algorithms, which can be used in the context of recognition, detection, and synchronization scenarios. Once again the algorithms are specially designed to work on some more compact representations than the ordinary dense real-valued matrices. Additionally, the proposed algorithms also care about the additive nature of an audio mixture, so that they can perform robustly in the presence of interference and noise.

Section 4.1 proposes a 2D hash function that converts a spectrogram into a bit string. Since it is a straightforward extension of WTA hashing, it inherits all the properties of WTA hashing while additionally providing robustness to the temporal stretches of audio signals. Section 4.2 follows to propose a matrix decomposition method that works on the sparse representation, which can be seen as an irregular matrix, since those rare non-zero elements might as well be encoded by using their position vectors rather than with a dense matrix if we care about the efficiency. Finally, Section 4.3 extends this model to 2D deconvolution case, where we find basis images and their 2D sparse activations to approximate a 2D sparse input space.

4.1 Keyword Spotting Using Spectro-Temporal WTA Hashing

Keyword spotting is a detection task where we locate a particular term in an observed utterance. For example, we might want to find the sentence in which a person of interest was mentioned in broadcast news; students might review a lecture and want to skip to the specific moment in which the lecturer started talking about a specific topic; companies need to categorize customers' inquiries based on keywords recorded through an automated call center, etc. All of these applications can be seen as sub-problems of ASR, where we want to recognize a very small number of words rather than trying to transcribe the entire content.

If we suppose that we use an ASR system for the keyword spotting task, we can see some unique challenges, since using a full ASR system is computationally expensive. In a modern ASR system the word sequence is following an assumed language model. To this end, the system needs to have transcribed some preceding non-target words, and this procedure requires additional computation. Thus, compared to a

keyword spotter that focuses only on a keyword, ASR systems are apt to be more accurate while slower and more expensive. Therefore, in constrained environments we might prefer a light-weight keyword spotter to a comprehensive ASR system.

In this section we emphasize certain aspects of the keyword spotting task, namely the cases where cost-efficient processing is critical and the input is noisy. First, our primary goal is to minimize computational requirements. For instance, the larger a speech database is, the more challenging a reasonably fast retrieval is. We might also want a voice command system to run continuously in the background to instantly pick up the keyword that initiates the system, requiring extremely low cost algorithms. Second, in the real-world examples those speech signals could be contaminated with various kinds of noise. Because noise can seriously degrade the performance of traditional approaches, e.g. Hidden Markov Model (HMM), a noise-robust algorithm is always preferable.

Traditional keyword spotting systems are usually based on keyword-specific HMM learned from frame-by-frame Mel-Frequency Cepstrum Coefficient (MFCC) features [54, 55, 56, 57]. Instead, our approach in this section shares the scheme proposed in [58], which tries to match spectro-temporal features rather than frame-based representations. The main idea of spectro-temporal matching is to think of a speech mel-spectrogram as a combination of local blocks, each of which distinctively represents a speech phenomenon. These local patches are more suitable to represent dynamics of speech, which are hard to represent using MFCCs with delta and delta-delta extensions. Therefore, the ordered spectro-temporal patches serve as features that encode information invariant to possible warping along both time and frequency axes. During the testing phase, the best match is fed to a Support Vector Machine (SVM) classifier for the final decision about whether the utterance has the keyword or not.

In this section, we propose a hashing-based matching that also considers variations within the same keyword label. Instead of local patches, we randomly pick up a few elements from the data matrix, and record the index of the maxima. If we repeat this trials, those indices of maxima work as a hash code that encodes relational orderings of the selected data elements. It was shown that this particular type of Locality Sensitive Hashing (LSH), WTA hashing, is efficient to match image features for fast object recognition tasks [38]. We adopt this technique in place of comprehensive pattern matching in [58] to overcome the known problems of MFCC-HMM techniques as well, but in a simpler way.

The efficient hashing-based matching scheme can also allow the system to be harmonized with the existing more comprehensive keyword spotters, such as the one with deep neural networks [59]. Since we mainly focus on constructing hash codes that work as discriminative binary features, whose Hamming distance by itself can measure the level of matching, a subsequent refinement of the decision using an

additional classifier can further enhance the performance. For example, we can use this system to quickly decide whether to activate the main classifier or not for every given short period of the input speech.

The key contributions of the proposed method are as follows: (a) extraction of features (hash codes) is simple and fast with complexity $\mathcal{O}(LM)$, where L and M are the length of the binary hashcode and the number of elements to be compared for selecting a maximum, respectively. (b) the comparison of features is done simply by the inverse hamming distance between bitstreams, which consist of LM bitwise AND ($\wedge$) and addition operations. (c) its hash codes are relatively robust to additive noise.

4.1.1 Assumptions

In the proposed spectro-temporal WTA hashing technique, we first assume that the input is a time-frequency representation of a short speech excerpt rather than a vector. Although we do not confine the proposed method to a particular type of time-frequency transform, in this section we stick to the MFCC representation, since the discrete cosine transform can help smooth the spectrogram. Note that WTA hashing works for the possibly negative MFCC values as well, while the ordered spectro-temporal patches in [58] are only for mel-spectrograms.

Another assumption is about a big training dataset. The goal of this hashing is to produce similar hash codes for a given keyword regardless of all the possible time-frequency warps and speaker-specific variations. For now, we rely on the abundance of the training templates to cover the spectral variations and speaker characteristics. In other words, we assume that there is always a good match in the database once the test signal contains the keyword. Unless we learn a parametric model, such as an HMM or an SVM classifier during training, having more training templates to compare calls for significantly more computation during the detection. However, thanks to the cheap bit operations that we use, this downside can be mitigated.

Big audio datasets are beneficial for more quality pattern matching since they preserve the manifold of the data space, on which a variety of recordings of the same keyword lies. It was shown that manifold preserving counterparts provide more quality audio analysis than traditional latent variable models [36, 44]. Once again the main bottleneck of utilizing the entire training dataset is the computational and spatial burden to handle the big data. Hence, the proposed hashing technique plays a major role in overcoming this limitation.

Using the proposed spectro-temporal WTA hashing, we hash each training sample (an MFCC matrix) and get a code with LM bits regardless of the different lengths of the samples. On the other hand, as we do not have the boundary information of the keyword spoken in the test signal, we simply follow the scheme

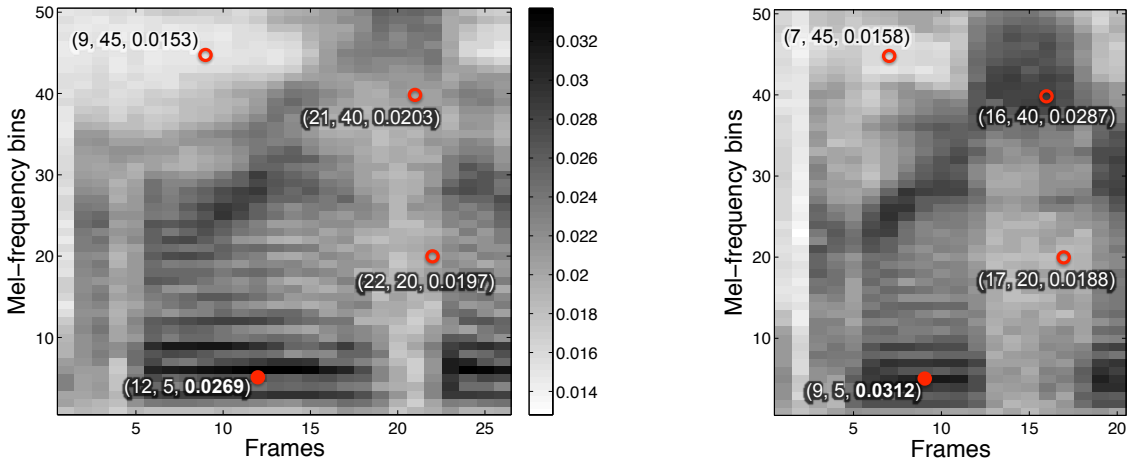


Figure 4.1: Examples of the spectro-temporal WTA hashing when $M = 4$. (a) A mel-spectrogram of the keyword `greasy` spoken by a female speaker. (b) Another one from a male speaker.

in [58] that assumes the length of the test keyword equals to the average of the training keywords.

4.1.2 Introductory examples

We extend the original WTA hashing into a spectro-temporal version so that temporal stretches can be covered. Figure 4.1 depicts the procedure of encoding a permutation in the proposed spectro-temporal WTA hashing. First, we can see that the two recordings of the same keyword have different lengths and fundamental frequencies. However, it is also noticeable that the two mel-spectrograms share some similar structures, such as a broadband noise part that corresponds to `g` and `r` sounds, and a long vowel region divided by a high pitched noise representing `ee` and `s` sounds, respectively. Red circles are the elements to compare whose positions were encoded in a row of the hash table $\mathcal{P}$. In this example, $\mathcal{P}$ has $M = 4$ columns, so we compare four different elements in the matrix at a time. Since the random indices are designed to reflect the proportional positions of the time axis, the resulting elements are likely to lie on the same local regions that roughly correspond to the spectro-temporal patches in [58].

In the example, the 2D random indices used are (0.35, 45), (0.46, 5), (0.85, 20), and (0.8, 40), where x-indices are not absolute, but proportional to the number of frames. This way, we are more likely to extract from the same regions regardless of the signal lengths. Therefore, the absolute x-coordinates, from which the matrix elements are extracted, are converted by multiplying the common proportional positions to the length of each signal: $[0.35, 0.46, 0.85, 0.8] \times 26 = [9, 12, 22, 21]$ and $[0.35, 0.46, 0.85, 0.8] \times 20 = [7, 9, 17, 16]$, respectively.

When we compare the elements on those positions, we see that the second ones at (0.46, 5) are the

Algorithm 4 Generating a spectro-temporal hash table

Input: L, M, K
Output: $\mathcal{P}^{L\times M\times 2}$
for $l \leftarrow 1$ to L and $m \leftarrow 1$ to M **do**
  Generate a random real number $0 < r < 1$
  Generate a random integer $i \in \{1, \cdots, K\}$
  $\mathcal{P}(l, m, 1) \leftarrow r, \quad \mathcal{P}(l, m, 2) \leftarrow i$
end for

Algorithm 5 Spectro-temporal WTA hashing

Input: $X \in \mathbb{R}^{K\times N}, \quad \mathcal{P}^{L\times M\times 2}$
Output: $z \in \{0, 1\}^{LM\times 1}$
Initialize $z \leftarrow \mathbf{0}^{LM\times 1}$
for $l \leftarrow 1$ to L **do**
  maxVal $\leftarrow -$ MAX_REAL
  \\The minimum negative real value in the machine.
  for $m \leftarrow 1$ to M **do**
    if $X\big(\lceil N\mathcal{P}(l, m, 1)\rceil, \mathcal{P}(l, m, 2)\big) >$ maxVal **then**
      maxVal $\leftarrow X\big(\lceil N\mathcal{P}(l, m, 1)\rceil, \mathcal{P}(l, m, 2)\big)$
      maxIdx $\leftarrow m$
    end if
  end for
  $z(M(l-1)+\text{maxIdx}) \leftarrow 1$
end for

maximum in both signals (filled red circles in the figures). Therefore, the hash codes share a common maximum index for this permutation, which in turn increases the hamming similarity by a bit. We repeat this encoding L times.

Note that in our experiments we use the MFCC representation, and the mel-spectrograms used in Figure 4.1 are merely for the illustrational purpose, since they are more intuitive to see.

4.1.3 The proposed keyword spotting scheme

Algorithm 4 shows the procedure of generating a spectro-temporal WTA hash table. We call this function once per a keyword to generate keyword-specific hash functions. Note that the proposed spectro-temporal hash table now has an additional third dimension that holds a pair of indices (r, i), where the x-coordinate r contains a proportional position in the time domain and i is the index along the y-axis, i.e. out of K MFCC coefficients in our case. By discretizing these indices, we can use the table in the following spectro-temporal WTA hashing as in Algorithm 5.

For a training data set with T recordings of the keyword w, we first extract MFCC, delta, and delta-delta to construct the input matrix X. Algorithm 4 follows to generate $\mathcal{P}_w$. Then, we hash all T recordings by using Algorithm 5 to generate $Z_{train} \in \{0, 1\}^{LM\times T}$ that consists of T binary hash codes.

For an unseen MFCC stream, we slide a window of N_w frames on it with the hopsize $0.2N_w$, where N_w equals to the average number of frames of the keyword w in the training MFCC matrices. For each window, we perform Algorithm 5 again with the same $\mathcal{P}_w$ and generate a hash code z_{test}. We consider it as a hit when z_{test} has a significantly similar template in the database with a threshold τ:

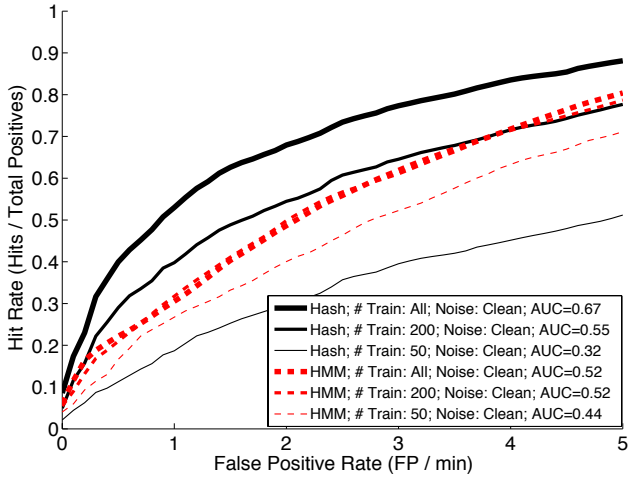
$$\max_t \sum_{j=1}^{LM} \Big(z_{test}(j) \ \wedge \ Z_{train}(j,t) \Big) > \tau. \tag{4.1}$$

4.1.4 Experiments

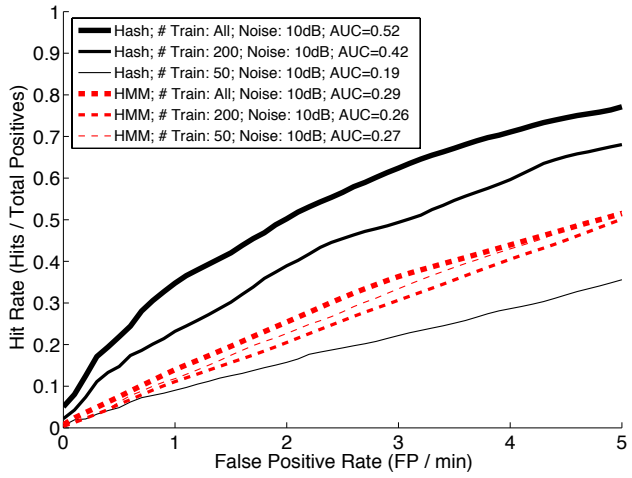
Experimental setups

The experimental setup we used is as follows:

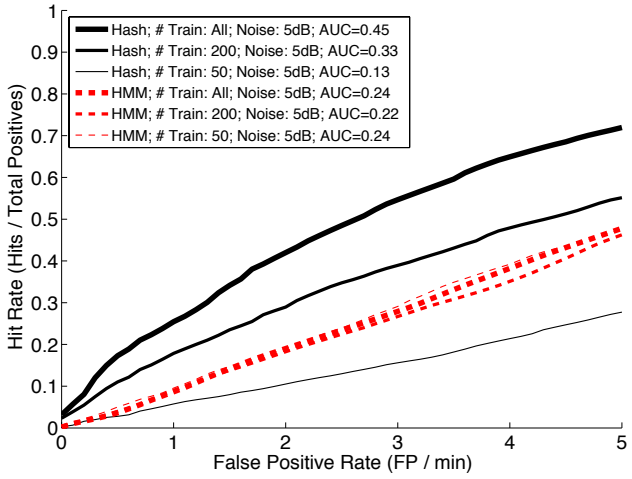
- X: The input matrix is 39 dimensional feature vectors from 13 MFCC, delta, and delta-delta.
- $T = \{462, 200, 50\}$: The entire 462 training samples per a keyword in the TIMIT corpus, or its smaller subsets, 200 and 50 samples, to test the performance with smaller training sets.
- Keywords w: From the two common sentences across all the speakers in TIMIT corpus, `sa1` and `sa2`, we select 10 words, {`dark`, `suit`, `greasy`, `wash`, `water`, `year`, `ask`, `carry`, `oily`, `rag`} as our keywords of interest while leaving out the "stop words", such as `to`, `an`, `in`, etc.
- Baseline: for the comparison we use a traditional HMM-based system with $C = 5$ hidden states. MFCC features are modeled with conditional Gaussian Mixture Model (GMM) with 3 components and diagonal covariances as in [58]. The log-likelihood of the windowed test signal measures the fit similarly to the hamming similarity in the proposed system.
- We also test the algorithms on noisy test signals with two different levels of Gaussian noise, 10 and 5 dB.
- Accuracy measures: we conduct the detection by changing the threshold τ on 128 positive and negative examples, respectively. Hit ratio (true positive) are therefore the number of hits divided by 128. The false positive rate is the number of negative examples that wrongly decided as hits divided by the summed lengths of the negative examples in minutes.
- $L = 512$ and $M = 2$: the hash code has 512×2 bits.



(a)



(b)



(c)

Figure 4.2: The averaged ROC curves. (a) Clean test speech. (b) Additional noise with 10dB SNR . (c) Additional noise with 5dB SNR .

Discussion and future work

Figure 4.2 shows the average of Receiver Operating Characteristic (ROC) curves of all the keywords. Each line stands for the average from each system. Figure 4.2 (a), (b), and (c) are the results from different levels of Gaussian noise added, whose Signal-to-Noise Ratio (SNR) are ∞ (clean), 10dB, and 5dB, respectively.

First, in all sub figures we can see that proposed hashing-based matching generally outperforms HMM in terms of the Area Under Curve (AUC) values (black lines versus red dashes) except when the number of training samples is not enough ($T = 50$). Also, the hashing technique is robust to the additive noise while HMM breaks down as soon as we start adding noise (compare (a), (b), and (c) in the increasing order of the noise level).

As expected, less training templates result in worse AUC values for both hashing and HMM cases in (a), while the HMM performances are not very different from each other in (b) and (c). This is because of the fact that HMM with the entire training samples starts from the low ROC curve already in the presence of noise. In the hashing case, the less number of training samples results in the loss of the data manifold (thicker lines versus thinner lines).

Apparently, since the HMM systems used are not the state-of-the-art keyword spotters, the comparison here is not to claim that the proposed system can outperform a classification-based system in general, but to show that even without the help of a proper classifier the proposed hash codes can provide a reasonable detection performance. It is promising that this hashing-only scheme can assist the more comprehensive systems by selectively activating the main classifier only when the hamming similarity exceeds the threshold.

The baseline HMM system first fits GMM on each 39 dimensional MFCC vector. Each fitting to one of the Gaussians requires $39 \times 5 = 195$ FLoating-point OPeration (FLOP) plus some constant, and we do that for 3 components and 5 hidden states. Hence, we need around $3000N_w$ FLOPs. Since we need to calculate this for only the last 20% of the new window, the final number is about $600N_w$. These are substituted with Algorithm 5 that takes only $LM = 1024$ FLOPs per a window.

The forward part of the forward-backward algorithm can be used to calculate the likelihood, which requires $60N_w - 55$ FLOPs per a sliding window with N_w frames and 5 hidden states [60]. In our naïve implementation we replace them with hamming similarity calculation: $LMT = 1024T$ bit-wise $\wedge$ and addition operations. However, one can easily construct a tree structure using hierarchical clustering on the hash codes, so that the comparison can be done in the order of $\mathcal{O}(LMH)$ where $H \ll T$ is the height of the tree. The tree structure can also reduce the linearly increasing spatial complexity of the hash code templates by keeping only some nodes closer to the root. Therefore, the proposed system has a clear merit

in an environment that prefers fixed-point operations. We leave this more careful implementation for our future research as well.

Additionally, in case we do not have enough training samples, we can modify the existing set using, for example, pitch shifting and equalizing. It would be also interesting for us to verify the case when we know about the speaker identity in the test signal and his or her clean keyword recordings are available for training.

4.1.5 Summary

We proposed a WTA hashing scheme that extracts spectro-temporal relationships among local speech features yielding better discrimination than HMM, particularly in the presence of noise. Experimental results on some keywords in TIMIT corpus show that the proposed method outperforms the HMM-based approach with the preference to the big training dataset. We believe that the proposed technique can be incorporated into the more comprehensive classification-based systems as a cheap pre-processor to decide whether to turn on the main classifier or not. By doing so, we can eventually decrease the computational complexity and the energy consumption of the keyword spotting task in the constrained environments, such as in mobile devices, which are in need of a lightweight keyword spotter.

4.2 Irregular Matrix Factorization

Latent component models on non-negative data have for a while been a very active area of research and have found numerous applications in a wide range of domains, from text analysis [30][32] and recommendation systems [33] to visual scene analysis [34] and music transcription [25]. Because many of these techniques trace their origins back to matrix decompositions, there is often the underlying assumption that the dimension axes of the input data are indexed using integers. Such an integer index is usually used to identify a word, a document, a pixel location, a Fourier frequency bin, etc., all of these quantities being discrete and countable. In other words, the inputs are designed so that they can be represented by a regular grid, most often represented by a matrix. Although this is a natural representation for many problems, e.g. a TF-IDF matrix, a spectrogram, or a digitized image, it is not a very flexible format for many continuous signal representations where the sampling or the representation can be irregular and/or parametric.

In this section we examine an approach that can analyze such inputs while maintaining the structure

Part of Section 4.2 has been published in [61].

of typical latent variable models, i.e. NMF. In particular we will focus on representations which are parametric, that is for each available data point we will have a real-valued number denoting its index in every dimension. We will be constraining our analysis to two-dimensional data, thereby directly extending techniques that operate on matrices (or two-dimensional distributions), but it is simple to extend this approach to arbitrary dimensions, any of which can be either discrete or real-valued.

In the remainder of this section we will introduce the basic model, a model that corresponds to NMF [21, 22]. We will show how to estimate such a model's parameters using data with real-valued dimensions and we will discuss the extra complications and options that arise. We will apply this technique to the analysis of time series and we will show that using such parametric-data approaches we can discover signal structure that would be otherwise invisible to traditional latent variable approaches.

4.2.1 NMF for irregularly-sampled data

Non-negative matrix factorization

A regular factorization of a time/frequency matrix is defined as:

$$\mathbf{X} \approx \mathbf{W} \cdot \mathbf{H} \tag{4.2}$$

where $\mathbf{X} \in \mathbb{R}_+^{M \times N}$ is a matrix containing time/frequency energies, and $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_Z] \in \mathbb{R}_+^{M \times Z}$ and $\mathbf{H} = [\mathbf{h}_1^\top, \mathbf{h}_2^\top, \cdots, \mathbf{h}_Z^\top]^\top \in \mathbb{R}_+^{Z \times N}$ represent Z frequency and time factors, respectively. NMF is a simple and useful factorization that estimates the two factors using the following iterative process:

$$\begin{aligned} \mathbf{P}_z &= \frac{\mathbf{w}_z \cdot \mathbf{h}_z}{\mathbf{W} \cdot \mathbf{H}}, \\ \mathbf{w}_z &= (\mathbf{X} \odot \mathbf{P}_z) \cdot \mathbf{1}^{N \times 1}, \\ \mathbf{h}_z &= \mathbf{1}^{1 \times M} \cdot (\mathbf{X} \odot \mathbf{P}_z), \end{aligned} \tag{4.3}$$

where $\mathbf{1}^{m \times n}$ is an $m \times n$ matrix of ones, $\odot$ and $\frac{[\cdots]}{[\cdots]}$ stand for element-wise multiplication and division, respectively. We normalize $\mathbf{w}_z$ by the sum of $\mathbf{h}_z$ at the end of every iteration in order to get a spectrum estimate that is unbiased by how much it appears over time. This also sets the magnitude of $\mathbf{W}$ so that we do not have multiple solutions that transfer energy between the two factors.

The downside of this formulation is that the frequency and time axes need to be sampled uniformly, meaning that at each time point we need to have an energy reading for all the frequency values, and vice versa. Unfortunately for certain types of time/frequency transforms, such as constant-Q transforms,

wavelets and reassigned spectrograms, this assumptions do not hold and the resulting time/frequency energies cannot be represented using a finite-sized matrix. For such representations we use a different format attaching to each energy value its exact frequency and time location. In order to factorize such transforms we need to redefine the factorization process to accept this new format.

Reformulation of NMF into a vectorized form

In this section we assume that the transforms that we use are regularly sampled as above, but we will use a different representation to allow us to extend this formulation to non-regularly sampled transforms later. Instead of using a matrix $\mathbf{X}$ to represent the time/frequency energies we will use three vectors, $\mathbf{f} \in \mathbb{Z}^{MN\times 1}$, $\mathbf{t} \in \mathbb{Z}^{MN\times 1}$, and $\text{vec}(\mathbf{X}) = \mathbf{x} \in \mathbb{R}_+^{MN\times 1}$, which will respectively hold the frequency coordinate, the time coordinate, and the energy value of each time/frequency point[1]. The elements of those vectors, $\mathbf{f}(i)$, $\mathbf{t}(i)$, and $\mathbf{x}(i)$, are indexed by $i = \{1, 2, \cdots, MN\}$.

Using the newly introduced formulation we can rewrite the factorization process as follows:

$$\mathbf{x} = \sum_{z=1}^{Z} \mathbf{v}_z \odot \mathbf{g}_z, \tag{4.4}$$

where now the pair of vectors $\mathbf{v}_z \in \mathbb{R}_+^{MN\times 1}$ and $\mathbf{g}_z \in \mathbb{R}_+^{MN\times 1}$ correspond to the values of the factors $\mathbf{W}$ and $\mathbf{H}$ as they are evaluated at the frequencies and times denoted by $\mathbf{f}$ and $\mathbf{t}$. With this, the iterative multiplicative update rules turn into the following form:

$$\begin{aligned}
\mathbf{p}_z &= \frac{\mathbf{v}_z \odot \mathbf{g}_z}{\sum_{z'=1}^{Z} \mathbf{v}_{z'} \odot \mathbf{g}_{z'}} \\
\mathbf{v}_z(i) &= \sum_{\forall j: \boldsymbol{f}(j)=\boldsymbol{f}(i)} \mathbf{x}(j)\mathbf{p}_z(j) \\
\mathbf{g}_z(i) &= \sum_{\forall j: \boldsymbol{t}(j)=\boldsymbol{t}(i)} \mathbf{x}(j)\mathbf{p}_z(j)
\end{aligned} \tag{4.5}$$

It is easy to show that if the frequency/time indices lie on a regular integer grids, i.e. $\mathbf{f}(i) \in \{1, 2, \cdots, M\}$ and $\mathbf{t}(i) \in \{1, 2, \cdots, N\}$, respectively, we will be performing the same operations as in (4.3). We can

[1]The $\text{vec}(\cdot)$ operator concatenates all the columns of its input matrix to a single column vector.

furthermore rewrite (4.5) to process all components simultaneously as:

$$\begin{aligned}\mathbf{P} &= \frac{\mathbf{V} \odot \mathbf{G}}{(\mathbf{V} \odot \mathbf{G}) \cdot \mathbf{1}^{K \times K}} \\ \mathbf{V} &= \mathbf{D}_f \cdot (\mathbf{P} \odot \mathbf{X}) \\ \mathbf{G} &= \mathbf{D}_t \cdot (\mathbf{P} \odot \mathbf{X})\end{aligned} \tag{4.6}$$

where the matrices, $\mathbf{P}$, $\mathbf{V}$, and $\mathbf{G}$, contain Z concatenated column vectors, each of which is for a latent variable z, e.g. $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_Z]$. Additionally, $\mathbf{D}_f, \mathbf{D}_t \in \{0,1\}^{MN \times MN}$ denote two matrices defined as:

$$\begin{aligned}\mathbf{D}_f(i,j) &= \begin{cases} 1, & \mathbf{f}(i) = \mathbf{f}(j) \\ 0, & \mathbf{f}(i) \neq \mathbf{f}(j) \end{cases} \\ \mathbf{D}_t(i,j) &= \begin{cases} 1, & \mathbf{t}(i) = \mathbf{t}(j) \\ 0, & \mathbf{t}(i) \neq \mathbf{t}(j) \end{cases}\end{aligned} \tag{4.7}$$

Multiplying with these matrices results in summing over all the elements that have the same frequency or time value respectively. The only difference between the formulation in this section and in (4.3) is that we will obtain the two factors in a different format so that:

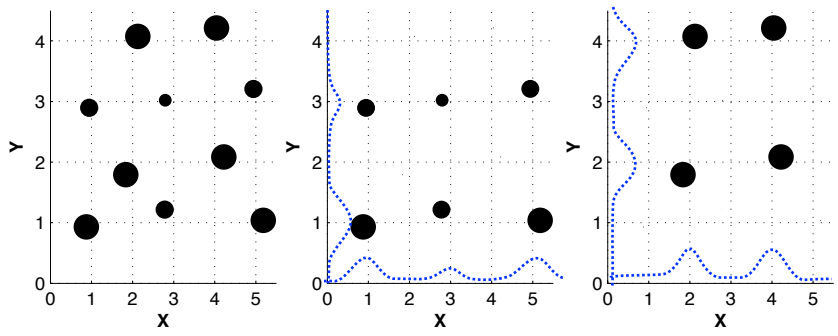
$$\begin{aligned}\mathbf{w}_z(m) &= \mathbf{v}_z(i), \quad \forall i : \mathbf{f}(i) = m \\ \mathbf{h}_z(n) &= \mathbf{g}_z(i), \quad \forall i : \mathbf{t}(i) = n \\ \text{vec}(\mathbf{w}_z \cdot \mathbf{h}_z) &= \mathbf{v}_z \odot \mathbf{g}_z\end{aligned} \tag{4.8}$$

where m and n are uniform indices defined in the ranges, $\{1, 2, \cdots, M\}$ and $\{1, 2, \cdots, N\}$, respectively.

Non-negative non-regular matrix factorization

The more interesting case is the one where the frequency and time vectors are real-valued and potentially comprised of unique elements. In this case the summations in (4.5) become meaningless since they will only sum over single points and will never capture the correlations that form as multiple frequencies get excited at roughly the same time.

To illustrate such a case let us consider the simple example as shown in Figure 4.3 (a), where we have $\mathbf{f} \in \mathbb{R}^{MN \times 1}$ and $\mathbf{t} \in \mathbb{R}^{MN \times 1}$, i.e. real-valued frequency/time indices. In this case we need to slightly amend the learning procedure. Previously we used co-activation information to update the learned components.



(a) (b) (c)

Figure 4.3: Example of a real-valued-index data set. In (a) we see a set of data that is not sampled on a grid, as is evident by the unaligned positioning of the data points. The size of the points indicates the magnitude of their assigned value $\mathbf{x}(i)$. In (b) and (c) we see two of the implied components that make up the data in (a), and their smoothed projections on both axes.

So, for example, if for two points $\mathbf{x}(i)$ and $\mathbf{x}(j)$ we had that $\mathbf{f}(i) = \mathbf{f}(j) = m$ and subsequently $\mathbf{D}_f(i,j) = 1$, we would perform a sum over them when we estimated $\mathbf{v}$. In the case above since all the frequencies are real-valued and potentially unique, this summation would never happen and instead the learned factors $\mathbf{v}$ and $\mathbf{g}$ would be uninformative. In order to alleviate that we redefine the two summing matrices such that $\mathbf{D}_f, \mathbf{D}_t \in \mathbb{R}_+^{MN \times MN}$ and:

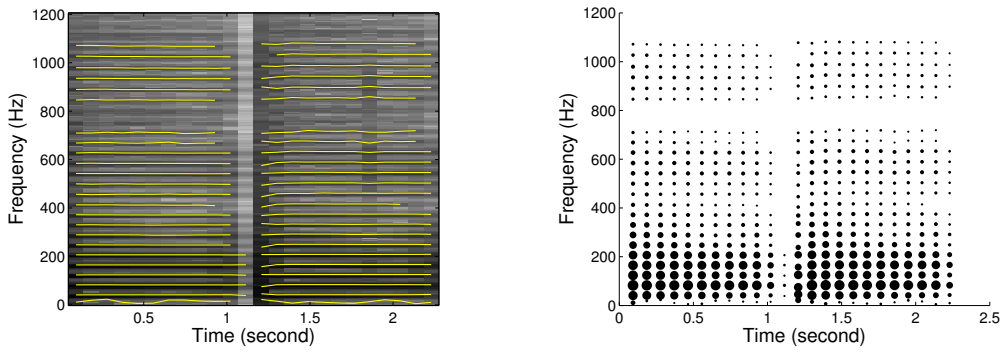
$$\mathbf{D}_f(i,j) = e^{\frac{-|\mathbf{f}(i)-\mathbf{f}(j)|^2}{\sigma_f^2}}, \quad \mathbf{D}_t(i,j) = e^{\frac{-|\mathbf{t}(i)-\mathbf{t}(j)|^2}{\sigma_t^2}} \tag{4.9}$$

This means that we still maintain that $\mathbf{D}_f(i,j) = 1, \forall i,j : \mathbf{f}(i) = \mathbf{f}(j)$ and $\mathbf{D}_t(i,j) = 1, \forall i,j : \mathbf{t}(i) = \mathbf{t}(j)$, but if we have the case where two frequency or time labels are close but not exactly the same we would still sum them, albeit using a lower weight. For distant points the corresponding values in these matrices will be very close zero, so no significant summation would take place.

Using this proposed approach, we obtain the results in Figure 4.3 (b) and (c). The discovered factorization successfully decomposes the non-uniformly spaced input samples into two intuitively correct latent components. This kind of input cannot be represented using matrix forms as the data indices are not integer-valued. Therefore, it is impossible to otherwise resolve this problem with any latent variable methods such as PLSI [30], PLCA [35], Latent Dirichlet Allocation (LDA) [32], or even matrix factorization methods such as NMF [21][22] and Singular Value Decomposition (SVD).

4.2.2 Experimental results

This section highlights the benefits of the proposed model by using some audio examples with parametric representations that are not amenable to analysis using matrix-based methods.



(a) (b)

Figure 4.4: Sinusoidal tracking example. (a) Zoomed-in STFT of a musical sound and estimated sinusoid components (yellow lines). (b) the frequency and intensity of sinusoids are represented with dots, where the size of the dot represents the intensity. Note that the frequency position of the dots is real-valued, but the time is sampled on a regular grid therefore is integer-valued.

An example with non-regular input along one-dimension

Suppose that we observe a non-regular input stream with a regular time interval. For instance, Figure 4.4 (a) is the result of a sinusoidal component estimation at every time frame of a series of STFT of a sound. We cannot represent that data using a matrix representation since each sinusoid is positioned on the vertical axis using a real-valued frequency estimate that will not necessarily line up with the integer-valued Fourier bins. In addition to that we have a different number of the sinusoids at different time frames which also makes it hard to force them into a matrix representation.

The sound that is being analyzed consists of two successive bass guitar notes at a low frequency range (around 41Hz), with a very small frequency difference between them (about 0.17 of a semitone). As is well known in the area of music analysis, if we decompose the STFT data of such a sound using an algorithm like PLSI, PLCA or NMF and request two components, we should see that each component will correspond to one of the notes played [5]. As we will see however, this particular sound is problematic with known techniques. Because of the low frequencies involved, we have to use a large Fourier analysis window (8192pt = 0.186 sec in this case) to obtain a high frequency resolution so that the two notes do not have an identical looking representation. Using a hop size of 50% and a Hann window we applied NMF with two components on the magnitude STFT of this sound and we decomposed it to two elements as shown in Figure 4.5. Upon examination we see that both components average out similar characteristics from both notes and fail to properly segment the input. This is because even with such a long analysis window the magnitude spectra of the two notes are not sufficiently different to be recognized as two components.

We now repeat this experiment, but instead of using the magnitude STFT data we use the sinusoidal analysis data from Figure 4.4 (b), which is real-valued on the frequency axis. This will provide the extra

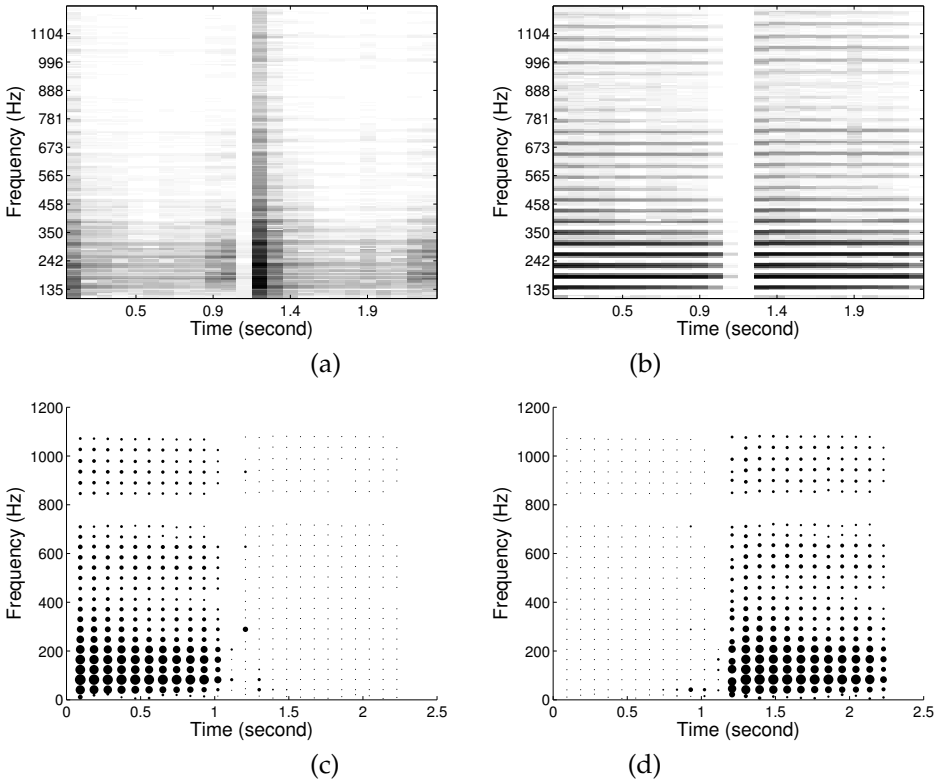


Figure 4.5: Separation results of regular and non-regular NMF. (a) First component estimate from regular NMF. (b) Second component estimate from regular NMF. (c) First component using non-regular NMF. (d) Second component using non-regular NMF.

frequency resolution we need, but will necessitate that we use the proposed algorithm to deal with the non-regular nature of our data. The decomposition results for two components are shown in Figure 4.5 (c) and (d), where bigger dots indicate more energy. We can see that this algorithm provides the desired decomposition, with each note being a discovered component. We note here that the better results are not a side effect of the algorithm, but rather of a better data representation that suits this problem. This algorithm only becomes necessary because this representation is not analyzable by other known methods.

Reassigned spectrogram: non-regular along both axes

In this section we will show an experiment where both axes of our input are real-valued. We will do so by making use of reassigned spectrograms. The reassignment method [62] provides an alternative representation of magnitude spectrograms by estimating a more accurate position of each spectrogram value and *reassigning* that value to a new more accurate time/frequency position. It basically breaks the grid structure by nudging each time/frequency bin out of an integer-valued location. Because of that nudging, the resulting spectrogram can exhibit infinite resolution in both frequency and time domains. This results a

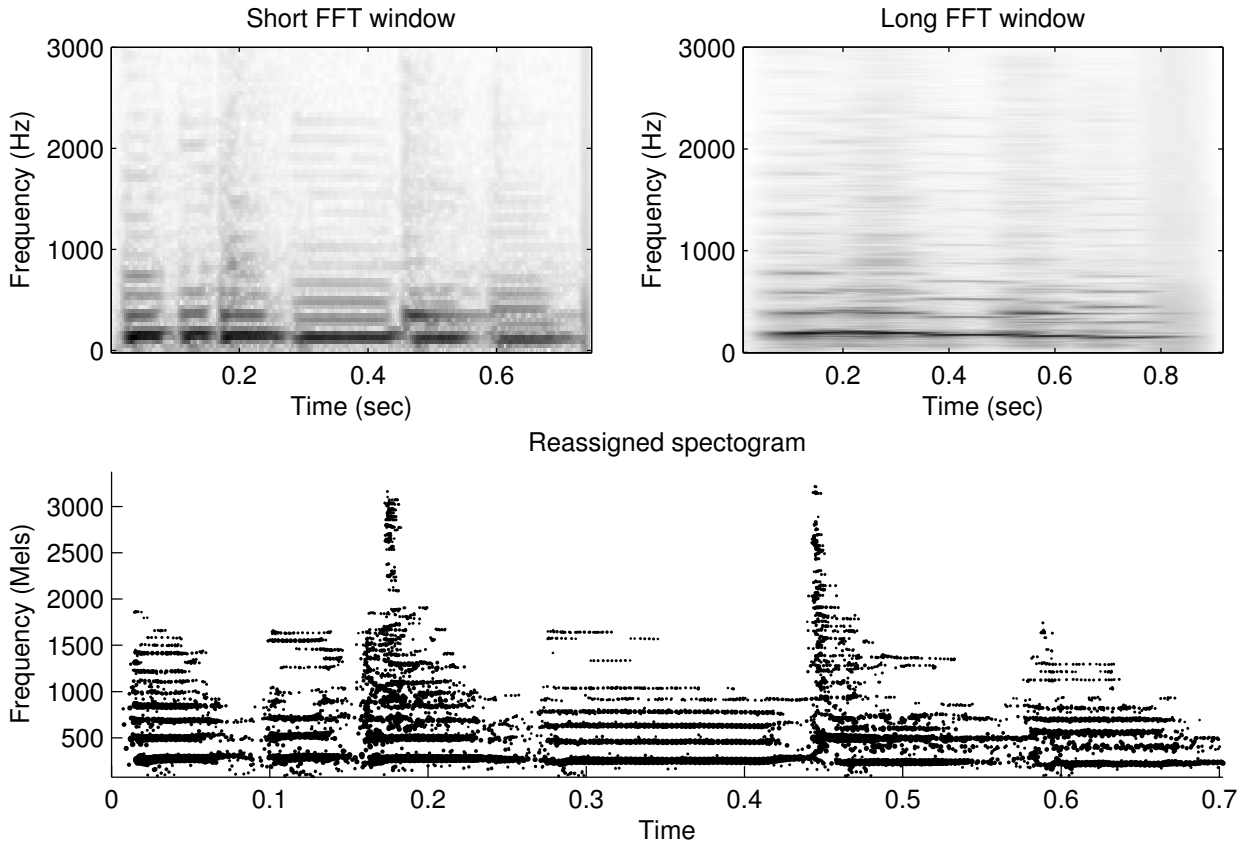


Figure 4.6: Comparison of a short-window STFT, a long-window STFT, and a reassigned spectrum. For the latter, the size of the points represents the amount of energy. For legibility we stretched the frequency axis to align with the Mel scale. Unlike the traditional spectrograms, this stretching is easy to do without any loss of information because of the parametric format of the reassigned spectra.

much more accurate time/frequency representation of a time series, but the data is now in a form that is very hard to decompose using traditional techniques.

To motivate using this representation, we use the first few seconds of the recording "Donna Lee" by Jaco Pastorius, which is a fast-paced bass solo with some percussion in the background. The played notes are $\{G_3, A_3, G_3, E_3, D_3, D_3^\flat\}$ and there are two different conga hits, one simultaneously with the third note and one with the fifth. Because of the low bass notes we would require high frequency resolution to be able to tell the notes part, but the fast note successions and percussion necessitate high temporal resolution. If we analyze this data using a traditional STFT we obtain the two representations shown at the top of Figure 4.6. We can see that for a short enough FFT size that provides good temporal resolution, the spectra of the bass notes are virtually indistinguishable, whereas for a large enough window where the note spectra become distinct the timing information is severely smeared. For any combination of STFT parameters it is impossible to obtain an NMF-style factorization that discovers the bass notes and the percussion hits. Alternatively

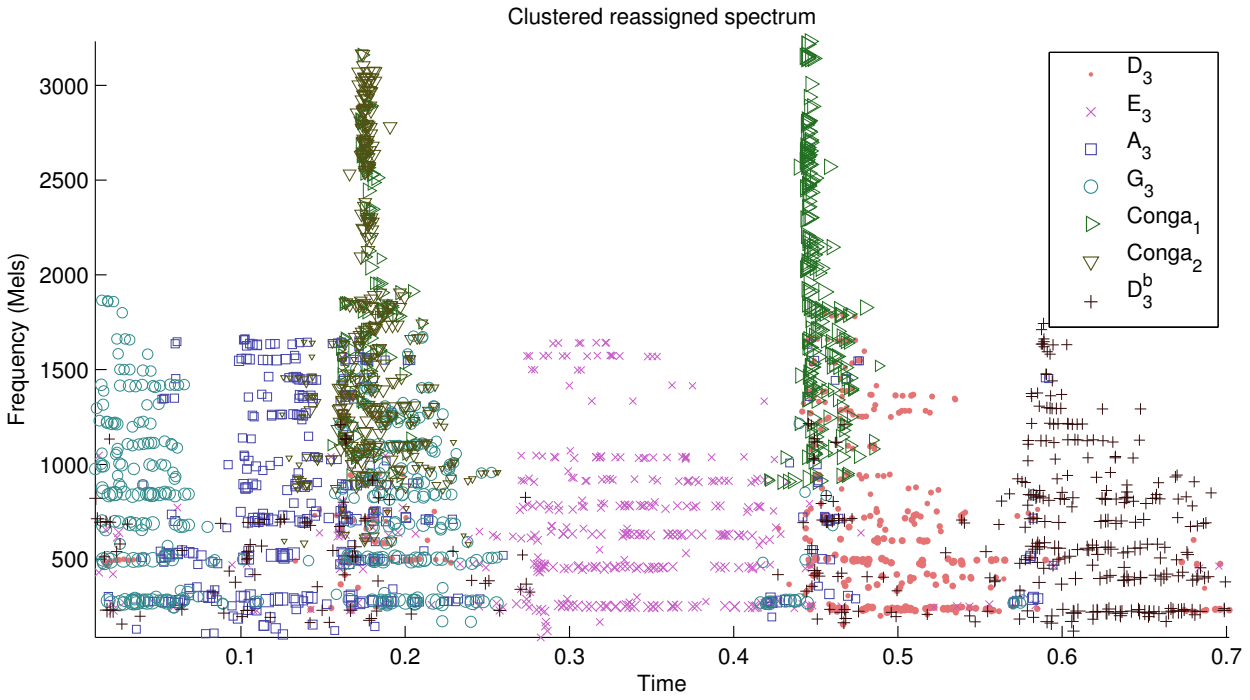


Figure 4.7: The reassigned spectrogram in Figure 4.6, with each point labelled by its component association, as denoted by both shape and color. In order to improve legibility not all input points are plotted. One can clearly see that the input is properly segmented according to the notes and the percussion hits.

we can use a reassigned spectrum as shown at the bottom of in Figure 4.6. In that representation it is easier to see the bass notes, as well as the two percussion hits.

In Figure 4.7 we show the reassigned spectrogram, with each point having been labeled according to which component is used to reconstruct it. As we would expect from an NMF-style analysis, the unique spectra of the different notes and the two percussion sounds should emerge as components. Although this is impossible to achieve using a standard STFT and NMF analysis due to time/frequency tradeoff constraints, using the proposed approach we successfully discover all the expected elements, despite their very close overlap in time and frequency.

Implementation notes

There are a couple of practical issues that we address in this section regarding the use of kernels. As should be evident variance of the Gaussian kernels $\mathbf{D}_f$ and $\mathbf{D}_t$ that we use can have a dramatic effect on the results. A very small variance will not fill the space enough to learn any structure, whereas too large a variance will blur the results. In the above cases we have a clear sense of the approximate spacing between our data so that we can make a good guess of the proper values, this might not always be the case though. An additional problem is that of computational complexity. Employing the two kernel matrices can be very

cumbersome when the number of data samples is in the tens of thousands. To alleviate that we clip small values of $\mathbf{D}_f$ and $\mathbf{D}_t$ to zero. By doing so we can use sparse matrix routines which accelerate computation significantly and also reduce memory footprint.

4.2.3 Summary

In this section we presented a latent component model that operates on inputs that do not lie on a regular grid. We formulated this as a vectorized form of matrix decomposition problem and derived a multiplicative update rules that are analogous to those of NMF. By running experiments on audio data representations that are parametric, we have shown that this algorithm performs as expected and is able to correctly analyze such irregular inputs that gridded-data techniques are not able to.

4.3 Irregular Nonnegative Factor Deconvolution (NFD)

Compact representation of the data can be helpful to speed up the pattern matching process in the audio applications. The usual way of handling audio signals is to convert a given short time frame of the signal to the frequency domain, i.e. STFT, and take the magnitude of the resulting complex valued matrix. This can be problematic in some cases:

- Irregular transform: it is addressed in [61] that sometimes the usual STFT grid does not provide desired resolution for the time or frequency dimension. We can make use of alternative irregular transforms to tackle this issue, but they result in non-matrix form data structures, which prevents the use of ordinary matrix-based techniques.
- Sparse landmarks: discarding all elements except the local maxima can be a way to get the compact representation. But, the resulting representation is a sparse matrix where most of elements are zeros. When we represent this kind of matrices with a pair of their positions and the value, we can get the compact representation, but they are not qualified for matrix-based techniques.

In this section we are greatly based on an existing technique where we can do NMF like decomposition on those irregular data types [61]. We first introduce the existing technique in section 2 and then introduce the proposed method in section 3.

4.3.1 NFD for irregularly-sampled data

In this section we propose an extended version of the non-regular NMF. What we want to do with this is that instead of assuming the linear decomposition model underlying in NMF algorithms we use a set of basis vectors as a basis image that can be convolved with filters. In this way, we can group the frequently adjacent basis vectors to represent a certain temporal structure of the data, which is hard to capture using NMF. Although this deconvolution model on the matrix inputs itself is not new at all, NFD on irregular data points is a novel approach.

Section 3.1 introduces the model where the convolution happens along only one direction (time). Section 3.2 introduces the model where the convolution happens along both time and frequency directions.

NFD along only one dimension (1D-NFD)

When we assume basis matrices, each of which holds a unique time-varying set of spectra, the NMF problem can be extended to a deconvolution model,

$$\mathbf{X} = \sum_{z=1}^{Z} \sum_{\tau=0}^{T-1} \mathbf{w}_z^\tau \cdot \overrightarrow{\mathbf{h}_z}^{\tau}, \tag{4.10}$$

where $\mathbf{w}_z^\tau$ is the τ-th one of the successive spectra of a basis matrix, and operation $\overrightarrow{\mathbf{h}_z}^{\tau}$ shifts the matrix $\mathbf{h}_z$ to the right by τ elements while filling the leftmost τ columns with zeros. Then, we reconstruct the input $\mathbf{X}$ with a sum of filtered basis matrices. Here we only need a filter $\mathbf{h}_z$ per a latent component, which is convolved with the basis matrix. This new reconstruction model leads us to a new set of update rules involving those temporal dynamics:

$$\mathbf{P}_z^\tau = \frac{\mathbf{w}_z^\tau \cdot \overrightarrow{\mathbf{h}_z}^{\tau}}{\sum_{z=1}^{Z} \sum_{\tau=0}^{T-1} \mathbf{w}_z^\tau \cdot \overrightarrow{\mathbf{h}_z}^{\tau}}, \tag{4.11}$$

$$\mathbf{w}_z^\tau = (\mathbf{X} \odot \mathbf{P}_z^\tau) \cdot \mathbf{1}^{N \times 1}, \tag{4.12}$$

$$\mathbf{h}_z^\tau = \mathbf{1}^{1 \times M} \cdot (\mathbf{X} \odot \mathbf{P}_z^\tau), \tag{4.13}$$

$$\mathbf{h}_z = \frac{1}{T} \sum_{\tau=0}^{T-1} \overleftarrow{\mathbf{h}_z^\tau}^{\tau} . \tag{4.14}$$

Reformulation of 1D-NFD into a vectorized form

As for the same types of vectorized inputs $\mathbf{f}(i)$, $\mathbf{t}(i)$, and $\mathbf{x}(i)$ as before, the deconvolution is defined as:

$$\mathbf{x} = \sum_{z=1}^{Z} \sum_{\tau=0}^{T-1} \mathbf{v}_z^\tau \odot \mathbf{g}_z^\tau, \tag{4.15}$$

where $\mathbf{v}_z^\tau \in \mathbb{R}_+^{MN\times 1}$ and $\mathbf{g}_z^\tau \in \mathbb{R}_+^{MN\times 1}$. The multiplicative update rules are now:

$$\mathbf{p}_z^\tau = \frac{\mathbf{v}_z^\tau \odot \mathbf{g}_z^\tau}{\sum_{z'=1}^{Z} \sum_{\tau'=0}^{T-1} \mathbf{v}_z^\tau \odot \mathbf{g}_z^\tau} \tag{4.16}$$

$$\mathbf{v}_z^\tau(i) = \sum_{\forall j:\mathbf{f}(j)=\mathbf{f}(i)} \mathbf{x}(j)\mathbf{p}_z^\tau(j) \tag{4.17}$$

$$\mathbf{g}_z^\tau(i) = \sum_{\forall j:\mathbf{t}(j)=\mathbf{t}(i)} \mathbf{x}(j)\mathbf{p}_z^\tau(j) \tag{4.18}$$

$$\mathbf{g}_z(i) = \frac{1}{T} \sum_{\tau=0}^{T-1} \sum_{\forall j:\mathbf{t}(j)=\mathbf{t}(i)+\tau} \mathbf{g}_z^\tau(j) \tag{4.19}$$

$$\mathbf{g}_z^\tau(i) = \sum_{\forall j:\mathbf{t}(j)=\mathbf{t}(i)-\tau} \mathbf{g}_z(j). \tag{4.20}$$

Note that we are not free to use the shift notation here as the input is not a grid anymore. But, $\mathbf{g}_z^\tau$ in (4.16) is a shifted version of $\mathbf{g}_z$ from the previous iteration using (4.20).

We can rewrite them with matrix notation as follow:

$$\begin{aligned}
\mathbf{P}^\tau &= \frac{\mathbf{V}^\tau \odot \mathbf{G}^\tau}{\sum_{\tau=0} \mathbf{V}^\tau \odot \mathbf{G}^\tau}, \\
\mathbf{V}^\tau &= \mathbf{D}_f \cdot (\mathbf{P}^\tau \odot \mathbf{X}), \\
\mathbf{G}^\tau &= \mathbf{D}_t \cdot (\mathbf{P}^\tau \odot \mathbf{X}), \\
\mathbf{G} &= \frac{1}{T} \sum_{\tau=0} (\mathbf{D}_\tau)^{-1} \cdot \mathbf{G}^\tau, \\
\mathbf{G}^\tau &= \mathbf{D}_\tau \cdot \mathbf{G},
\end{aligned} \tag{4.21}$$

where $\mathbf{X} = \mathbf{x} \cdot \mathbf{1}^{1\times Z}$. The kernel matrices $\mathbf{D}_f$, $\mathbf{D}_t$ are the same with the previous ones in (4.7), but $\mathbf{D}_\tau$, and

$(\mathbf{D}_\tau)^{-1}$ should consider time lacks τ and both operations of shifts to the left and right as well:

$$\mathbf{D}_\tau(i,j) = \begin{cases} 1, & \mathbf{t}(i) = \mathbf{t}(j) + \tau \\ 0, & \mathbf{t}(i) \neq \mathbf{t}(j) + \tau, \end{cases}$$
$$(\mathbf{D}_\tau)^{-1}(i,j) = \begin{cases} 1, & \mathbf{t}(i) + \tau = \mathbf{t}(j) \\ 0, & \mathbf{t}(i) + \tau \neq \mathbf{t}(j). \end{cases} \tag{4.22}$$

Non-regular 1D-NFD

The final non-regular version of 1D-NFD we propose uses the proposed vectorized update rules in (4.21) except that we replace kernel matrices in (4.22) with corresponding Gaussians, such as in (4.23):

$$\mathbf{D}_\tau(i,j) = e^{\frac{-|\mathbf{t}(i)-\mathbf{t}(j)-\tau|^2}{\sigma_\tau^2}}, \quad (\mathbf{D}_\tau)^{-1}(i,j) = e^{\frac{-|\mathbf{t}(i)+\tau-\mathbf{t}(j)|^2}{\sigma_\tau^2}}. \tag{4.23}$$

Experimental result

Figure 4.8 shows the deconvolution results on the toy data where two distinct basis images are repeating along the horizontal direction. Note that the resulting basis images contain temporal structures that need several untied basis vectors to represent it in the non-regular NMF case.

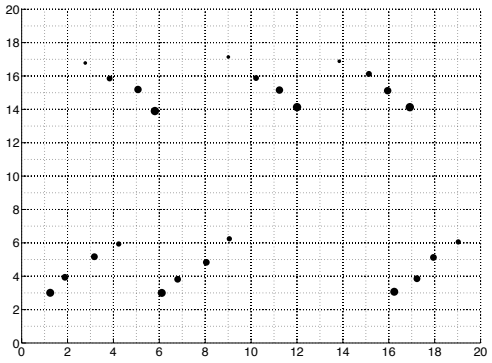
4.3.2 NFD along both dimensions (2D-NFD)

In Figure 4.8 we can see that the repeating basis images are captured by the proposed algorithm. But, it is also noticeable that the patterns are not really moving along the vertical direction. If they do, the proposed 1D-NFD will fail to capture the underlying patterns. In this section we further expand 1D-NFD to 2D version so that the kernel basis matrix are free to appear everywhere.

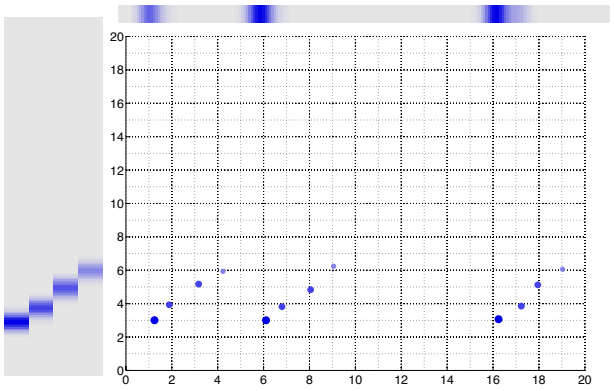
We start from the reconstruction model first:

$$\mathbf{X} = \sum_{z=1}^{Z} \sum_{\tau=0}^{T-1} \sum_{\phi=0}^{F-1} \mathbf{K}_z^{\phi,\tau} \overset{\uparrow\phi,\rightarrow\tau}{\mathbf{A}_z}, \tag{4.24}$$

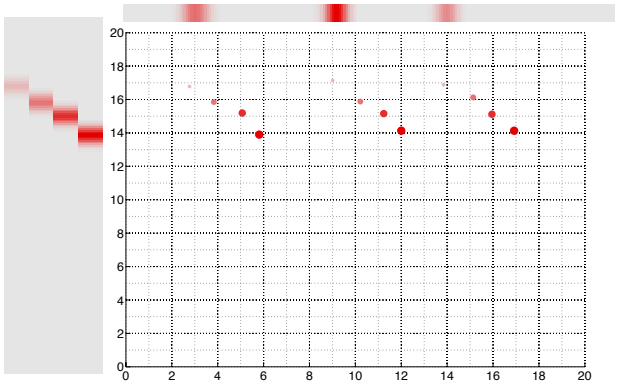
where $\mathbf{K}_z^{\phi,\tau} \in \mathbb{R}_+^{F\times T}$ is the discretized kernel at the position (ϕ,τ), and operation $\overset{\uparrow\phi,\rightarrow\tau}{\mathbf{A}_z}$ shifts the matrix $\mathbf{A}_z$ to the right by τ elements and to up by ϕ while filling the leftmost τ columns and bottom ϕ rows with zeros. Then, we reconstruct the input $\mathbf{X}$ with a sum of filtered basis matrices. Here we only need a 2D filter $\mathbf{A}_z$ per a latent component, which is convolved with the basis matrix. This new reconstruction model leads



(a) Input



(b) First component reconstruction



(c) Second component reconstruction

Figure 4.8: 1D-NFD results on two sets of repeating 2D patterns, which are irregularly located on the 2D surface.

us to a new set of update rules involving those temporal and frequency dynamics:

$$\mathbf{P}_z^{\phi,\tau} = \frac{\mathbf{K}_z^{\phi,\tau} \overset{\uparrow\phi,\rightarrow\tau}{\mathbf{A}_z}}{\sum_{z=1}^{Z}\sum_{\tau=0}^{T-1}\sum_{\phi=0}^{F-1}\mathbf{K}_z^{\phi,\tau} \overset{\uparrow\phi,\rightarrow\tau}{\mathbf{A}_z}}, \tag{4.25}$$

$$\mathbf{K}_z^{\phi,\tau} = (\mathbf{X} \odot \mathbf{P}_z^{\phi,\tau}) \cdot \mathbf{1}^{N\times M}, \tag{4.26}$$

$$\mathbf{A}_z^{\phi,\tau} = \mathbf{X} \odot \mathbf{P}_z^{\phi,\tau}, \tag{4.27}$$

$$\mathbf{A}_z = \frac{1}{TF}\sum_{\tau=0}^{T-1}\sum_{\phi=0}^{F-1} \overset{\downarrow\phi\leftarrow\tau}{\mathbf{A}_z^{\phi,\tau}}. \tag{4.28}$$

Note that now the posterior probabilities $\mathbf{P}$ is a five dimensional tensor with axis for z, t, f, ϕ, and τ.

Reformulation of 2D-NFD into a vectorized form

As for the same types of vectorized inputs $\mathbf{f}(i)$, $\mathbf{t}(i)$, and $\mathbf{x}(i)$ as before, the deconvolution is defined as:

$$\mathbf{x} = \sum_{z=1}^{Z}\sum_{\tau=0}^{T-1}\sum_{\phi=0}^{F-1}\mathbf{K}_z^{\phi,\tau}\mathbf{g}_z^{\phi,\tau}, \tag{4.29}$$

where $\mathbf{g}_z^{\phi,\tau} \in \mathbb{R}_+^{MN\times 1}$. The multiplicative update rules are now:

$$\mathbf{p}_z^{\phi,\tau} = \frac{\mathbf{K}_z^{\phi,\tau}\mathbf{g}_z^{\phi,\tau}}{\sum_{z=1}^{Z}\sum_{\tau=0}^{T-1}\sum_{\phi=0}^{F-1}\mathbf{K}_z^{\phi,\tau}\mathbf{g}_z^{\phi,\tau}} \tag{4.30}$$

$$\mathbf{K}_z^{\phi,\tau} = \sum_{\forall i}\sum_{\phi=0}^{F-1}\sum_{\tau=0}^{T-1}\mathbf{x}(i)\mathbf{p}_z^{\phi,\tau}(i) \tag{4.31}$$

$$\mathbf{g}_z^{\phi,\tau}(i) = \mathbf{x}(i)\mathbf{p}_z^{\phi,\tau}(i) \tag{4.32}$$

$$\mathbf{g}_z(i) = \frac{1}{FT}\sum_{\tau=0}^{T-1}\sum_{\phi=0}^{F-1}\sum_{\substack{\forall j:\mathbf{t}(j)=\mathbf{t}(i)+\tau \\ \mathbf{f}(j)=\mathbf{f}(i)+\phi}}\mathbf{g}_z^{\phi,\tau}(j) \tag{4.33}$$

$$\mathbf{g}_z^{\phi,\tau}(i) = \sum_{\substack{\forall j:\mathbf{t}(j)=\mathbf{t}(i)-\tau \\ \mathbf{f}(j)=\mathbf{f}(i)-\phi}}\mathbf{g}_z(j). \tag{4.34}$$

Note that we are not free to use the shift notation here as the input is not a grid anymore. But, $\mathbf{g}_z^{\phi,\tau}$ in (4.30) is introduce to represent the shifted version of $\mathbf{g}_z$ along both directions from the previous iteration using (4.34).

We can rewrite (4.33) and (4.34) with matrix notation as follow:

$$\mathbf{g}_z = \frac{1}{FT}\sum_{\tau=0}^{T-1}\sum_{\phi=0}^{F-1}(\mathbf{D}_{\phi,\tau})^{-1}\cdot\mathbf{g}_z^{\phi,\tau},$$
$$\mathbf{g}_z^{\phi,\tau} = \mathbf{D}_{\phi,\tau}\cdot\mathbf{g}_z, \tag{4.35}$$

The kernel matrices $\mathbf{D}_{\phi\tau}$ and $\mathbf{D}_{\phi\tau}^{-1}$ should consider time lacks τ and frequency shifts ϕ, so they are defined by:

$$\mathbf{D}_{\phi,\tau}(i,j)\begin{cases}1, & \mathbf{t}(i)=\mathbf{t}(j)+\tau \quad \text{and} \quad \mathbf{f}(i)=\mathbf{f}(j)+\phi\\ 0, & \text{otherwise,}\end{cases}$$
$$\mathbf{D}_{\phi,\tau}(i,j)^{-1}\begin{cases}1, & \mathbf{t}(i)+\tau=\mathbf{t}(j) \quad \text{and} \quad \mathbf{f}(i)+\phi=\mathbf{f}(j)\\ 0, & \text{otherwise.}\end{cases} \tag{4.36}$$

Non-regular 2D-NFD

The final non-regular version of 2D-NFD we propose uses the proposed vectorized update rules from (4.30) to (4.32), and (4.35) except that we replace kernel matrices in (4.36) with corresponding Gaussians, such as in (4.23):

$$\mathbf{D}_{\phi,\tau}(i,j) = e^{\frac{-|\mathbf{t}(i)-\mathbf{t}(j)-\tau|^2}{\sigma_\tau^2}+\frac{-|\mathbf{f}(i)-\mathbf{f}(j)-\phi|^2}{\sigma_\phi^2}},$$
$$(\mathbf{D}_{\phi,\tau})^{-1}(i,j) = e^{\frac{-|\mathbf{t}(i)+\tau-\mathbf{t}(j)|^2}{\sigma_t^2}+\frac{-|\mathbf{f}(i)+\phi-\mathbf{f}(j)|^2}{\sigma_\phi^2}}. \tag{4.37}$$

4.3.3 Experimental result

Figure 4.9 shows the deconvolution results on the toy data where two distinct basis images are repeating along the horizontal and vertical directions. Note that the resulting basis images contain temporal structures and now it can capture activations in different frequency regions.

4.3.4 Summary

In this section we also showed that the irregular matrix factorization model can be extended to the factor deconvolution models as well.

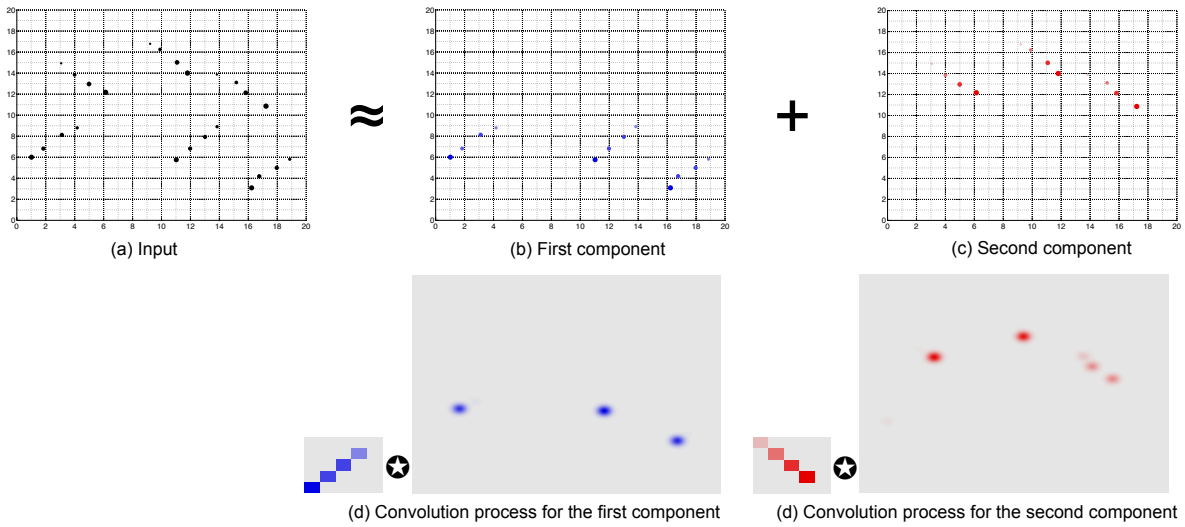


Figure 4.9: 2D-NFD Results on two sets of repeating 2D patterns, which are irregularly located on the 2D surface.

Chapter 5

Big Ad-Hoc Sensor Array Processing:
Collaborative Audio Enhancement

Because of widespread use of hand-held devices, we often find many overlapping recordings of an audio scene. Our goal in this chapter is to fully utilize these low cost noisy data by extracting common audio sources from them so as to produce a higher quality rendering of the recorded event. Hence, it can be seen as a collaborative approach to audio enhancement sharing some similar concepts with crowdsourcing methods [63, 64]. The first step towards unifying these recordings is to synchronize them, something we can easily achieve using one of the efficient and robust synchronization methods proposed in the past [65, 66]. Once this is done, one could simply use the best available recording at any point in time, assuming there is an automated way of quality-ranking the signals. This can be the simplest implementation of *collaborative audio enhancement*, where we can take advantage of other people's recordings to improve ours. However, such simple reasoning does not work for many common cases, so we will address this problem using a different approach.

Figure 5.1 shows a case where the obvious approach might fail. Between the two synchronized recordings, we cannot simply choose one because both are deficient, albeit in a different way. The bottom recording has a poor high frequency response, which could be the effect of a low-cost microphone or aggressive audio coding. On the other hand, the full bandwidth recording at the top has some interference in the 3 – 4.3 second region, which is however not present in the bottom one.

As the number of input recordings increases, the unique distortions in each recording make choosing a single best recording difficult, if not impossible. One could encounter various types of nonlinear artifacts or interferences, e.g., the audience chatter near the microphone, lens zooming noises, button clicks, clipping, band-pass filtering, etc. Eventually we would like to solve this problem by using information from all recordings and combine it appropriately in order to produce a higher quality render.

Nonnegative Matrix Partial Co-Factorization (NMPCF) was proposed to extract common spectral components out of multiple music excerpts in the past. Its several versions focussed on various characteristics of drum sounds that are expected to be common across multiple signals: spectral similarity between the drum solo signals and the drum source components in the music mixture [67], repeatability of drum source

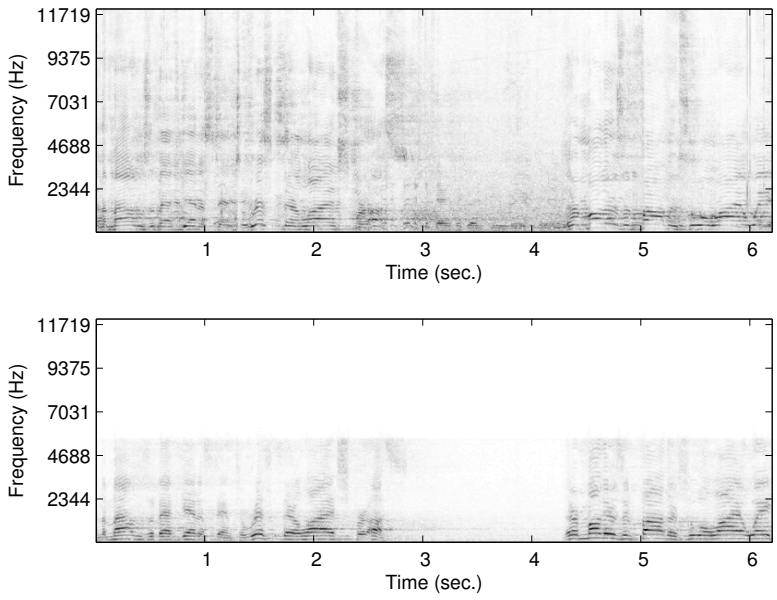


Figure 5.1: An example of a difficult scenario, when a synchronization and selection method can easily fail to produce a good recording. In this case we observe unwanted interference (top) and the other is band-limited (bottom).

components across all the chunks of the song [68], and their unified version [26]. Convolutive Common Nonnegative Matrix Factorization (CCNMF) was recently introduced to recover the common music and effect parts from multiple soundtracks with different languages [69]. CCNMF differs from NMPCF in that it shares both basis vectors and corresponding encodings of the common components to extract the music and effects while the set-aside track-specific ones capture dialogues in particular languages.

In Section 5.1 Probabilistic Latent Component Sharing (PLCS) is proposed to handle this problem. In PLCS we assume that there is a dominant and common source across all the recordings, which can be captured by some shared and fixed parameters during the simultaneous topic modeling for all the recordings. At the same time, PLCS also sets asides some individual parameters per model to capture the recording-specific interference. A Bayesian approach to make use of a prior knowledge about the source can be also incorporated, along with a post-processing to take care of the band-limited signals, too.

In Section 5.2 we speed up the Collaborative Audio Enhancement (CAE) procedure by using a hashing technique that can adaptively select the best set of recordings given the current estimation of the dominant source. The smaller nearest neighborhood set serves as an input to the PLCS algorithm for that iteration, so that the amount of computation during the EM updates for PLCS is greatly reduced while providing with better sound quality, because now the algorithm focuses more on the better recordings while largely

ignoring too noisy inputs.

5.1 Probabilistic Latent Component Sharing (PLCS)

The proposed method, Probabilistic Latent Component Sharing (PLCS), is based on the probabilistic counterparts of NMF [21, 22], such as PLSI [30, 31] and PLCA [35]. PLCS extends PLCA with the common component sharing concept. PLCS differs from the NMPCF-based methods in that it decomposes each input matrix into three parts, rather than just two, so that we can share both bases and encoding matrices while providing slack in the model by letting the weights of the components to not be shared. Because PLCS controls the contribution of the latent components with probabilistic weights, $P^{(l)}(z)$, it gives more intuitive interpretation of the roles of components in the reconstruction whereas in the CCNMF model [69] they are absorbed in the filtering factor. Moreover, because the whole process is based on the probabilistic model, we could explicitly take advantage of Bayesian approaches, which is not straightforward in either NMPCF or CCNMF. The Bayesian approach provides a straightforward way to involve a certain prior knowledge about the bases, which we can get in advance from the cleaner, but different versions of the similar sources. Finally, we propose an additional post processing method to efficiently consolidate recording-specific reconstructions.

5.1.1 Symmetric Probabilistic Latent Component Analysis (PLCA)

Given the magnitude of an input spectrogram, $V = |X|$, with elements $V_{f,t}$ indexed by the frequency bin f and the time frame t, symmetric PLCA tries to maximize the log-likelihood $\mathcal{P}$ of observing the energy quanta of $V_{f,t}$,

$$\begin{aligned}\mathcal{P} = \sum_{f,t} V_{f,t} \log P(f,t) &= \sum_{f,t} V_{f,t} \log \sum_{z} P(f,t|z)P(z) \\ &= \sum_{f,t} V_{f,t} \log \sum_{z} P(f|z)P(t|z)P(z).\end{aligned}$$

To get the second equality, the component-specific distributions $P(f,t|z)$ is further factorized into three factors: the frequency distribution $P(f|z)$, its temporal activations $P(t|z)$, and the component specific weights $P(z)$. Note that the term "symmetric" came from this tri-factorization [30], which eventually let us have control over additional temporal distributions of components as well as frequency distributions. This being

Part of Section 5.1 has been published in [70].

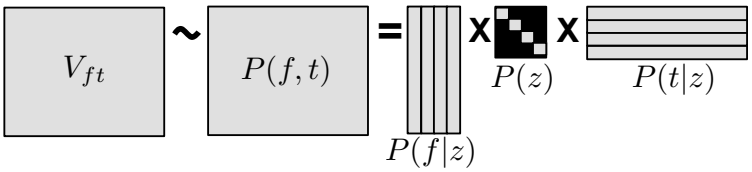


Figure 5.2: A matrix representation of the PLCA with four components. Note that the weights $P(z)$ are represented as a diagonal matrix.

a latent variable model, we use the Expectation-Maximization (EM) algorithm to estimate its parameters. In the E-step we find a posterior probability of the latent variable z given the time and frequency indices,

$$P(z|f,t) = \frac{P(f|z)P(t|z)P(z)}{\sum_z P(f|z)P(t|z)P(z)}.$$

In the M-step the expected complete data log-likelihood is maximized, which yields to the following update rules:

$$P(f|z) = \frac{\sum_t V_{f,t}P(z|f,t)}{\sum_{f,t} V_{f,t}P(z|f,t)}, \quad P(t|z) = \frac{\sum_f V_{f,t}P(z|f,t)}{\sum_{f,t} V_{f,t}P(z|f,t)}, \quad P(z) = \frac{\sum_{f,t} V_{f,t}P(z|f,t)}{\sum_{f,t,z} V_{f,t}P(z|f,t)}. \tag{5.1}$$

Figure 5.2 depicts the relationship between the input matrix V and the estimated joint distribution $P(f,t)$ from which the observations were drawn.

5.1.2 PLCS algorithms

Let us assume that there are L input magnitude spectrogram matrices, corresponding to L available recordings in the collaborative audio enhancement application. We partition the values of the latent components in the l-th recording $z^{(l)}$ into two disjoint subsets, $z^{(l)} = z_C \cup z_I^{(l)}$, where z_C is the subset that contains indices of the common components shared across all recordings, and $z_I^{(l)}$ contains those of all the other components present only in the l-th recording. Now, the log-likelihood $\mathcal{P}$ of observing L given recordings can be written as:

$$\mathcal{P} = \sum_l \sum_{f,t} V_{f,t}^{(l)} \log \left\{ \sum_{z \in z_C} P_C(f|z)P_C(t|z)P^{(l)}(z) + \sum_{z \in z_I^{(l)}} P_I^{(l)}(f|z)P_I^{(l)}(t|z)P^{(l)}(z) \right\}. \tag{5.2}$$

The main new feature in (5.2) is to fix both the spectral and the temporal distributions to be same across all inputs for $z \in z_C$, which are specified as the common variables $P_C(f|z)$ and $P_C(t|z)$. On the other hand, components indicated by $z \in z_I^{(l)}$ represent recording-specific sound components, such as interferences,

characterized by parameters $P_I^{(l)}(f|z)$ and $P_I^{(l)}(t|z)$. We refer to this model as PLCS, for which the E-step is:

$$P^{(l)}(z|f,t) = \frac{P^{(l)}(f|z)P^{(l)}(t|z)P^{(l)}(z)}{\sum_{z\in z^{(l)}} P^{(l)}(f|z)P^{(l)}(t|z)P^{(l)}(z)}, \forall z \in z^{(l)}.$$

Note that the parameters $P^{(l)}(f|z)$ and $P^{(l)}(t|z)$ can either refer to the common parameters $P_C(f|z)$ and $P_C(t|z)$ when $z \in z_C$ or $P_I^{(l)}(f|z)$ and $P_I^{(l)}(t|z)$ when $z \in z_I^{(l)}$, respectively.

Using Lagrange multipliers to ensure that the probability distributions sum to one, we maximize the expected complete data log-likelihood with following update rules as the M-step:

For $z \in z_I^{(l)}$

$$P_I^{(l)}(f|z) = \frac{\sum_t V_{f,t}^{(l)} P^{(l)}(z|f,t)}{\sum_{f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}, \; P_I^{(l)}(t|z) = \frac{\sum_f V_{f,t}^{(l)} P^{(l)}(z|f,t)}{\sum_{f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}, \tag{5.3}$$

For $z \in z_C$

$$P_C(f|z) = \frac{\sum_{l,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}{\sum_{l,f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}, \quad P_C(t|z) = \frac{\sum_{l,f} V_{f,t}^{(l)} P^{(l)}(z|f,t)}{\sum_{l,f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}, \tag{5.4}$$

For $z \in z^{(l)}$

$$P^{(l)}(z) = \frac{\sum_{f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}{\sum_{z,f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t)}.$$

Note that the updates for $P_C(f|z)$ and $P_C(t|z)$ include summation over l to involve all the reconstructions of common components.

5.1.3 Incorporating priors

It is often useful to involve prior knowledge about the parameters in probabilistic models. For instance, we can have a clean recording of the same content as in the provided inputs, albeit recorded at a different time (e.g. a studio recording of a song whose recordings we obtain from a concert). Or, it is also possible to assume that the interferences are a certain kind of sources, e.g. human voice. On the other hand, we cannot simply learn the bases of those a priori signals and fix them as our target parameters, $P_C(f|z)$ or $P_I^{(l)}(f|z)$, as there is no guarantee that the a priori known signals have exactly the same spectral characteristics with the target sources. To address this problem we follow a Bayesian approach to derive a MAP estimator of the parameters.

First, we learn the bases of the magnitude spectrograms of the similar sources and interferences by directly applying PLCA update rules in (5.1). The learned bases vectors $P_{\text{source}}(f|z)$ and $P_{\text{interf}}^{(l)}(f|z)$ are

used in the PLCS model to construct a new expected complete data log-likelihood

$$\langle \mathcal{P} \rangle = \sum_{l,f,t} V_{f,t}^{(l)} \Bigg\{ \sum_{z \in z_C} \Big(P^{(l)}(z|f,t) \log P_C(f|z) P_C(t|z) P^{(l)}(z) + \alpha P_{\text{source}}(f|z) \log P_C(f|z) \Big) + \sum_{z \in z_I^{(l)}} \Big(P^{(l)}(z|f,t) \log P_I^{(l)}(f|z) P_I^{(l)}(t|z) P^{(l)}(z) + \beta P_{\text{interf}}^{(l)}(f|z) \log P_I^{(l)}(f|z) \Big) \Bigg\},$$

where α and β controls the amount of influence of the prior bases. Once again, by using proper Lagrange multipliers, we can derive the final M-step with priors as follow:

For $z \in z_I^{(l)}$

$$P_I^{(l)}(f|z) = \frac{\sum_t V_{f,t}^{(l)} P^{(l)}(z|f,t) + \beta P_{\text{interf}}^{(l)}(f|z)}{\sum_{f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t) + \beta P_{\text{interf}}^{(l)}(f|z)}, \tag{5.5}$$

For $z \in z_C$

$$P_C(f|z) = \frac{\sum_{l,t} V_{f,t}^{(l)} P^{(l)}(z|f,t) + \alpha P_{\text{source}}(f|z)}{\sum_{l,f,t} V_{f,t}^{(l)} P^{(l)}(z|f,t) + \alpha P_{\text{source}}(f|z)}. \tag{5.6}$$

E-step and the other M-step update rules are not changed from the original PLCS model. Figure 5.3 summarizes the whole PLCS process on three different inputs: low-pass filtered, high-pass filtered, and mid-pass filtered inputs. All three inputs also contain unique distortions represented with different noise patterns in the figure.

Note that the first common component of $l = 1$ case (first row) degrades the reconstruction as its basis vector has high frequency energy while $V^{(1)}$ was low-pass filtered. Therefore, the first weight in the diagonal matrix $P^{(1)}(z = 1)$ has a very low (dark) value. Similarly, $P^{(2)}(z = 4)$, $P^{(3)}(z = 1)$, and $P^{(3)}(z = 4)$ are also those weights that *turn off* inactive common components. Note also that the a priori learned bases $P_{\text{source}}(f|z)$ are full-banded and have somewhat different spectral shapes from the common bases, so they cannot replace the common bases as they are.

To recover the magnitudes of the desired sources, we multiply the sum of the posterior probabilities of $z \in z_C$ to the input complex-valued spectrograms $X_{f,t}$,

$$\hat{S}_{f,t}^{(l)} = X_{f,t}^{(l)} \sum_{z \in z_C} P^{(l)}(z|f,t),$$

where $\hat{S}^{(l)}$ is the spectrogram of the separated sources from the l-th input.

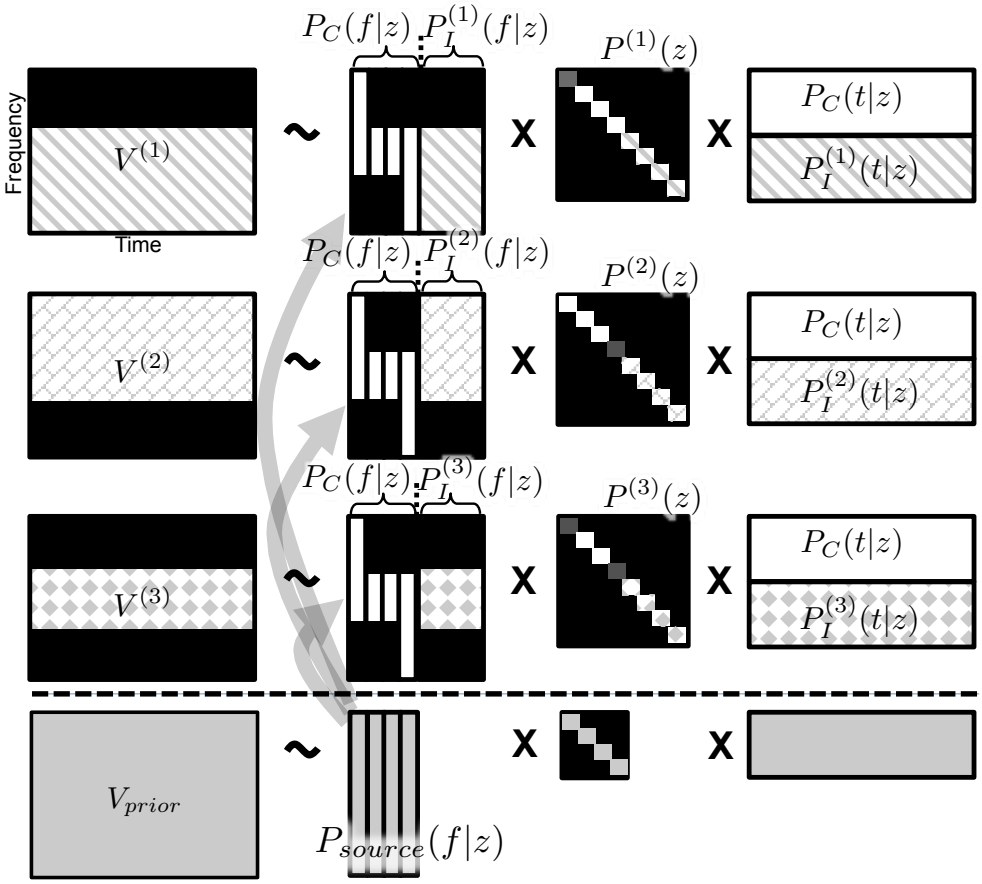


Figure 5.3: An example of common source separation process using PLCS on three defected input matrices and prior information.

5.1.4 Post processing

It is possible that the recorded signals exhibit non-uniform frequency responses due to recording device and format specifications. The PLCS method can identify the isolated common sources, but it is not expected to ameliorate effects like frequency response losses, since that information will be coded in the basis vectors and is not readily accessible as an artifact. We propose a collaborative post processing step to address this issue. Our approach is motivated by the fact that even if most of the recordings are filtered in some way, one recording that did not go through such filtering can give us enough information to recover the full-banded reconstruction. Suppose that we get L recovered spectrograms $\hat{S}^{(l)}_{f,t}$ using PLCS. The post processing begins with getting the normalized average magnitude spectrum of those reconstructions $y^{(l)}_f = \frac{\sum_t |\hat{S}^{(l)}_{f,t}|}{\sum_{f,t} |\hat{S}^{(l)}_{f,t}|}$. Now, we can obtain some global weights by considering the balance among different recordings in each particular frequency bin, $w_f = \frac{\sum_l y^{(l)}_f}{\max_l y^{(l)}_f}$.

Then, the final complex spectrogram of the desired output is obtained by dividing the sum of the band-limited reconstructions with the corresponding elements of the global weight:

$$\hat{S}_{f,t} = \frac{\sum_l \hat{S}^{(l)}_{f,t}}{w_f}. \tag{5.7}$$

5.1.5 Experimental results

In this section, we compare the proposed PLCS models and other relevant methods in terms of SDR [48], because we desire to reduce all the artifacts, noises, and interferences. To this end, we use five different single channel songs with 44.1kHz sampling rate and 16 bit encoding, each of which has a pair of versions: a 15 seconds-long clean live recording S as the source and a 30 seconds-long clean studio recording S_{prior} as the prior information of the source. The professional live recording S goes through three different sets of artificial deformations to simulate usual recording scenarios. The resulting three mixture spectrograms are:

- $X^{(1)}$: Low-pass filtering at 8kHz (a recording with a low sampling rate) / additional female speech as an interference
- $X^{(2)}$: High-pass filtering at 500Hz / additional female speech different from $X^{(1)}$ as an interference
- $X^{(3)}$: Low-pass filtering at 11.5kHz / high-pass filtering at 500Hz / clipping.

Short-time Fourier transform was applied to the signals with following settings: 1024 sample frame-length and 512 sample of hop size. For the priors, we get 100 bases for the source prior $P_{\text{source}}(f|z)$ from

the studio recording S_{prior} while 50 interference prior bases $P_{\text{interf}}(f|z)$ are learned from anonymous female speeches [51].

We compare five different models, including:

- Median: Pixel-wise medians of L input magnitude spectrograms are calculated as follows

$$|\hat{S}_{f,t}| = \text{median}(V^{(1)}_{f,t}, V^{(2)}_{f,t}, \cdots, V^{(L)}_{f,t}).$$

  The phase information of the sum of inputs is used to invert the reconstruction to the time domain.

- Oracle PLCA with ideal bases: We run PLCA on the source $|S|$ to get the ideal bases $P^{(l)}_{\text{ideal}}(f|z)$ that perfectly represent spectral characteristics of the desired source. And then, we apply PLCA again on each $V^{(l)}$ by initializing and fixing some of the bases with $P^{(l)}_{\text{ideal}}(f|z)$ while learning the others to capture interferences and artifacts. We get 100 bases for $P^{(l)}_{\text{ideal}}(f|z)$ from the first PLCA, and learn 50 individual components in the second round. Note that we also use the compensation process from Section 5.1.4 for this model.

- PLCS with initialization: This model learns the common bases and encodings using the PLCS model and the post processing. It initializes its bases with $P_{\text{source}}(f|z)$ learned from the studio recording, but learn the bases $P_C(f|z)$ using the usual update rules (5.3) and (5.4) rather than (5.5) and (5.6). We randomly initialize 50 individual bases for $P^{(l)}_I(f|z)$.

- PLCS with the source prior: This PLCS model uses the source priors $P_{\text{source}}(f|z)$ both to initialize and to learn $P_C(f|z)$ using (5.6). 50 individual bases are randomly initialized for $P^{(l)}_I(f|z)$ and learned using (5.3).

- PLCS with the source and interference priors: This full PLCS model uses both the source and interference priors, $P_{\text{source}}(f|z)$ and $P_{\text{interf}}(f|z)$ to initialize them and learn them using both (5.5) and (5.6). Note that we do not assume the interference prior for $X^{(3)}$, because it is riddled with clipping, not an additional interference.

Figure 5.4 shows the SDR improvements caused by the proposed systems. The most noticeable observation is that the medians of the three mixture spectrograms do not provide good results as there is no guarantee that the median of the contaminated pixels is from the common source. For the given five songs, we can also see that the proposed PLCS models outperform the maximal performance bound of the PLCA model with ideal bases, mainly because of the strong sharing of both spectral and temporal aspects of the

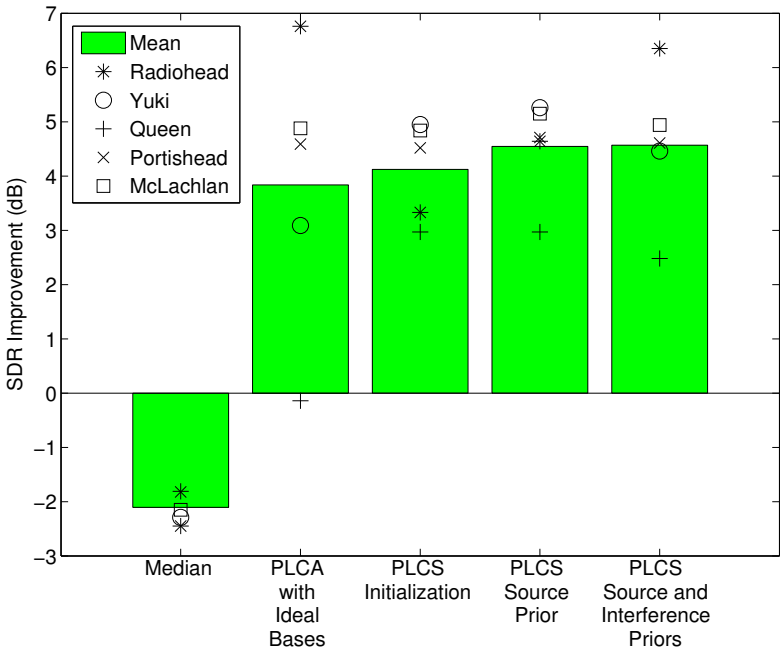


Figure 5.4: The mean and song-specific improvements of SDR by each model for the consolidated reconstruction.

common components. Moreover, PLCS models exhibit less variance than PLCA. On top of that, PLCS with priors can provide better performance than the usual initialization method by gently reducing the impact of prior information as the iteration i increases ($\alpha = \beta \propto e^{-i}$). Although it is not observable in this objective quality measurements, adding the interference priors improves the perceptual sound quality of the result.

5.1.6 Summary

We propose the PLCS model for collaborative audio enhancement, where common audio sources are constructed out of multiple noisy recordings of the same audio scene. The model is characterized by its ability to share both spectral and temporal marginal factors while providing a level of flexibility by not sharing the weight probabilities $P^{(l)}(z)$. PLCS also models individual artifacts from the common sources by not sharing some components. The advantage of this sharing concept was shown by experiments on commercial music signals by outperforming the ordinary PLCA model even with ideal bases which are not available in realistic cases. The proposed model is also equipped with a consolidation process that can harmonize the recording-specific reconstructions. Finally, prior information can be easily use in this model, which further improves the performance in our simulations.

5.2 Neighborhood Searches for Efficient PLCS on Massive Crowdsourced Recordings

CAE tries to extract the most significant source out of a set of noisy observations, *crowdsourced recordings*, potentially collected from socially shared data. In the CAE scenario, we assume that each observed recording can be uniquely contaminated, e.g. by an artifact from aggressive audio coding, an interfering sound captured only by that sensor, a bad frequency response of the microphone, clipping, etc [70]. Such recordings can also be thought of as signals from an ad-hoc microphone array in the sense that the channels are not synchronized and the sensor locations and characteristics are unknown [17, 18].

One challenge is to synchronize such a large set of signals. An effective approach is to assume a calibration signal in all the recordings as a guide to align with [17]. Another more realistic approach extracts noise-robust landmarks from audio spectrograms to identify videos [72], or to synchronize audio-visual signals [65], where robust matching is done efficiently using integer operations. In this section we assume that all signals are already aligned using one of the aforementioned methods, and instead we focus on the enhancement part.

Another challenge is to recover the geometric information. A closed form solution for the sensor location estimation problem using signals contaminated with Gaussian noise was proposed in [17]. A beamforming technique for an ad-hoc microphone array was also proposed in [18] in the presence of localized noise sources, while the clustering-based calibration of sensors does not scale up well to the much bigger and heterogeneous array setup that we assume in this section. More recently, an ad-hoc array calibration method was proposed [73], which enhances a noisy distance matrix using a low-rank approximation. The matrix completion is effectively done by NMF [21, 22] particularly if some elements in the distance matrix are missing. However, it was evaluated when at least more than half of the locations are known, and the source separation performance was not examined.

PLCS allows some topics, or equivalently latent components in the context of the other latent variable models, to share same parameters during simultaneous latent variable modeling on the synchronized magnitude spectrograms [70], as a probabilistic version of NMPCF [26]. Since CAE assumes that all recordings have captured some elements of the most common source, PLCS estimates the dominant source as a mixture of shared components, while the unshared components at each recording-specific topic model are used to explain the unique interferences that are to be discarded. Unfortunately, PLCS does not scale satisfac-

Part of Section 5.2 has been published in [71].

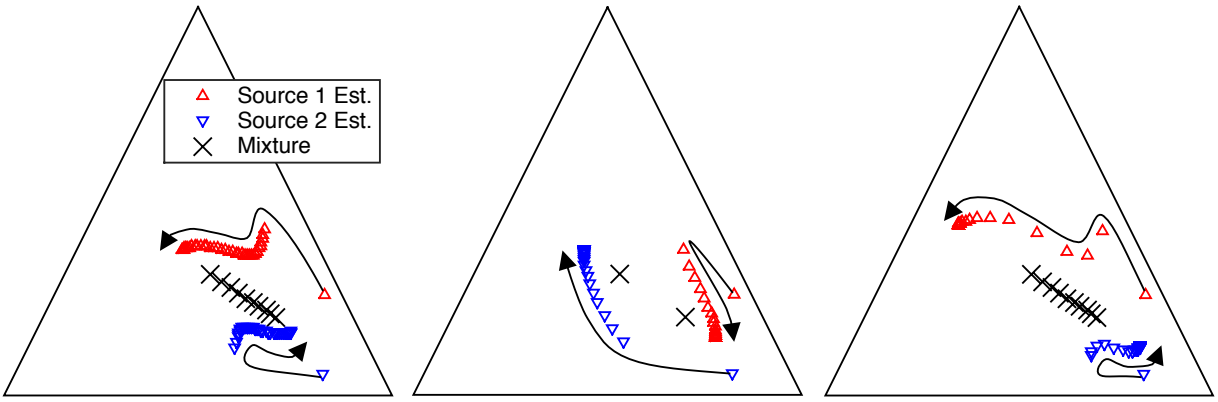


Figure 5.5: PLSI topic modeling with various conditions: (a) Ordinary PLSI (b) An oracle PLSI only on the data samples that are closest to the optimal solutions (c) PLSI updates only on the running nearest neighbors to the current estimates of the sources. In the figure, all the source estimates are shown from every iteration, and their order in time is represented with a curved arrow. All parameters start from the same position for comparison.

torily so as to allow analysis of a large number of input recordings: the complexity linearly increases as the number of recordings gets bigger. Furthermore, more social data is not always helpful, because poor recordings at locations far away from the source of interest may include a great deal of interferences and not contribute to the reconstruction of the desired source significantly.

We propose a streamlined PLCS algorithm that adaptively uses a subset of the available recordings rather than the whole set. The selection is done at every EM update based on cross entropy between the normalized magnitude spectrograms of the dominant source estimation and the noisy observations. Additionally, we show how to employ hashing to improve the complexity of the requisite search. If Hamming distance between hashed spectrograms is correlated to the original distance, it can reduce the search space by providing with a candidate set, followed by the second search with the full metric only on those candidates. The proposed methods indeed give better performances in simulated CAE tasks by selectively utilizing crowdsourced data made up from up to a thousand individual recordings. Finally, the proposed methods converge to a better solution faster.

5.2.1 Neighborhood-based topic modeling

Probabilistic Latent Semantic Indexing (PLSI) [31, 30] has been widely used in audio research, and it has been known that sparse overcomplete coding is beneficial for the separation performance, since sparse

coding on the training spectra leads to a tighter convex hull for the source [74][1]. In this section we do not specifically focus on the sparseness of encoding, but it is also true for the proposed model that the data samples that are closer to the corners of the convex hull (or the normalized basis vectors of NMF) contribute more to the parameter estimation. For example, Figure 5.5 (a) shows that the ordinary PLSI learns a subspace that reconstructs the mixture samples as convex combination of the two converged sources. In other words, the mixture samples should lie on or near the line that connects the learned sources for a quality approximation. In (b) we do the same PLSI modeling, but only on the two mixture points closest to the optimal solutions. This is an oracle run, because usually we cannot know the optimal solution in advance. What we can see is that the two samples are just enough to recover the same subspace with a permutation ambiguity.

An alternative to the oracle scenario is to find the nearest neighbors at every iteration given a set of source estimates. Figure 5.5 (c) demonstrates this case. Once again, the source estimates converge to the same subspace as in the ordinary PLSI and the one with an oracle minimal set of inputs. Formally, we can intervene in the EM updates of original PLSI by forming a tentative set of neighbors so that the computation is done only on the set as follows:

$$P(f|z) \leftarrow \frac{P(f|z)\sum_{t\in\mathcal{N}_z} V_{f,t}P_t(z)}{\sum_z P(f|z)P_t(z)}, P(f|z) \leftarrow \frac{P(f|z)}{\sum_f P(f|z)}, \tag{5.8}$$

$$P_t(z) \leftarrow \frac{P_t(z)\sum_f V_{f,t}P(f|z)}{\sum_z P(f|z)P_t(z)}, P_t(z) \leftarrow \frac{P_t(z)}{\sum_z P_t(z)}, \tag{5.9}$$

where the M-step updates are equivalently reformulated to include the E-step, so that the posterior probabilities are updated more often. In (5.8) we do the summation only over the nearest neighbor set $\mathcal{N}_z$, which is a subset of all spectra whose elements are always closer to the z-th basis vector $P(f|z)$ than the non-neighbors as follows:

$$-\sum_f \hat{V}_{f,t\in\mathcal{N}_z} \log P(f|z) < -\sum_f \hat{V}_{f,t'\notin\mathcal{N}_z} \log P(f|z), \tag{5.10}$$

where the column vectors of $\hat{V}$ are normalized, i.e. $\sum_f \hat{V}_{f,t} = 1$, for the proper calculation of cross entropy. The set should be small enough to effectively represent the corners of the convex hull, similarly to the locality preservation issues in manifold learning, such as in [49]. We refresh $\mathcal{N}_z$ according to the new $P(f|z)$ after every M-step in (5.8).

Note that if this set is optimal from the start and does not change, the results will be similar to Figure 5.5

[1]NMF with KL-divergence as the error function is also known to be equivalent to PLSI with a proper parameter normalization [75]. Therefore, the proposed methods can be directly used in NMF-based systems as well.

(b). If $\mathcal{N}_z$ always includes all data samples, then the updates in (5.8) and (5.9) are equivalent to the ordinary PLSI's by resulting in Figure 5.5 (a).

5.2.2 Neighborhood-based PLCS

Neighborhood-based extension

Neighborhood-based extensions of a probabilistic topic model were proposed in [36, 44] for manifold preserving source separation. However, those models are designed to find out a sparse code from an overcomplete dictionary, where the dictionary is a large collection of clean source spectra. Therefore, the search is on the clean spectra, not on the noisy spectrograms.

In this section we propose a new PLCS model employing a neighbor search on the spectrograms. At every EM iteration, we update the parameters by analysing only a subset of the recordings. The subset can be also refreshed every time according to the distances between the recordings and the new source estimation. In the CAE scenario, the source estimate is not just a component anymore, but a convex combination of the shared components across the multiple simultaneous probabilistic topic models.

First, for the l-th recording, we assume that its magnitude spectrogram is generated from a distribution that can be decomposed into the common and individual topics:

$$V^l_{f,t} \sim \sum_{z \in z_C} P_C(f|z) P_C(t|z) P^l(z) + \sum_{z \in z^l_I} P^l_I(f|z) P^l_I(t|z) P^l(z), \tag{5.11}$$

where the parameters with C as subscripts, but with no superscript, are common across all models, while the subscript I stands for the recording-specific individual parameters along with a superscript l to distinguish recordings. We use a symmetric PLSI version [30] in which the temporal encoding $P(t|z)$ represents probabilities over time, and consequently requiring an addition weight $P(z)$ that governs the global activation of the component. The global weights are grouped into two, i.e. z_C and z^l_I, which denote the sets of indices for the common and individual components, respectively.

The parameters are updated with the EM algorithm as in [70], but this time we introduce the neighborhood concept as in Section 5.2.1. Therefore, the updates use only some selected recordings that belong to the neighbor set $\mathcal{N}_S$ that are nearest to the common source. The series of M-steps are as follows:

For $l \in \mathcal{N}_S$ and $z \in z_I^l$,

$$P_I^l(f|z) \leftarrow \frac{P_I^l(f|z)P^l(z)\sum_t V_{f,t}^l P_I^l(t|z)}{\sum_{z' \in z_C \cup z_I^l} P^l(f|z')P^l(t|z')P^l(z')}, \quad P_I^l(f|z) \leftarrow P_I^l(f|z) / \sum_f P_I^l(f|z), \tag{5.12}$$

$$P_I^l(t|z) \leftarrow \frac{P_I^l(t|z)P^l(z)\sum_f V_{f,t}^l P_I^l(f|z)}{\sum_{z' \in z_C \cup z_I^l} P^l(f|z')P^l(t|z')P^l(z')}, \quad P_I^l(t|z) \leftarrow P_I^l(t|z) / \sum_t P_I^l(t|z), \tag{5.13}$$

For $l \in \mathcal{N}_S$ and $z \in z_C$,

$$P_C(f|z) \leftarrow \sum_{l \in \mathcal{N}_S} \frac{P_C(f|z)P^l(z)\sum_t V_{f,t}^l P_C(t|z)}{\sum_{z' \in z_C \cup z_I^l} P^l(f|z')P^l(t|z')P^l(z')} + \beta\alpha_{f,z}, \quad P_C(f|z) \leftarrow P_C(f|z) / \sum_f P_C(f|z), \tag{5.14}$$

$$P_C(t|z) \leftarrow \sum_{l \in \mathcal{N}_S} \frac{P_C(t|z)P^l(z)\sum_f V_{f,t}^l P_C(f|z)}{\sum_{z' \in z_C \cup z_I^l} P^l(f|z')P^l(t|z')P^l(z')}, \quad P_C(t|z) \leftarrow P_C(t|z) / \sum_t P_C(t|z), \tag{5.15}$$

For $l \in \mathcal{N}_S$ and $z \in z_C \cup z_I^l$

$$P^l(z) \leftarrow \frac{P^l(z)\sum_{f,t} V_{f,t}^l P^l(f|z)P^l(t|z)}{\sum_{z' \in z_C \cup z_I^l} P^l(f|z')P^l(t|z')P^l(z')}. \tag{5.16}$$

Note that the E-step is absorbed in the listed M-steps as well. For the parameters notated without subscripts C or I their associations should be obvious from the context, or it does not matter whether they are shared or not. For the shared basis vectors, which make up the dominant source, we can also use prior knowledge if the nature of the target source is known, e.g. clean bases from female speech, from a studio recording of the same song played in the scene, etc. We use a conjugate prior $\alpha_{f,z}$ for this, whose contribution is controlled by β as in [70].

Sharing is achieved by doing another average over l for those common parameters in (5.14) and (5.15). Therefore, when it comes to hundreds of recordings, this summation would be computationally burdensome had it not been for using the neighboring subset $\mathcal{N}_S$. Focusing only on $\mathcal{N}_S$ also helps speed up learning the individual parameters in (5.12) and (5.13), because we can largely skip all the recording-specific modeling when l does not belong to $\mathcal{N}_S$. Once again, $\mathcal{N}_S$ is a set based on cross entropy relationships, but between the observed noisy spectrogram and the estimated source spectrogram as follows:

$$-\sum_{f,t} \hat{V}_{f,t}^{l \in \mathcal{N}_S} \log \hat{S}_{f,t} < -\sum_{f,t} \hat{V}_{f,t}^{l' \notin \mathcal{N}_S} \log \hat{S}_{f,t}, \tag{5.17}$$

where the spectrograms $\hat{S}$ and $\hat{V}^l$ are normalized along both axes, e.g. $\sum_{f,t} \hat{V}_{f,t}^l = 1$. The source is estimated

from the average of the common components over the participating recordings:

$$S_{f,t} \leftarrow \frac{1}{|\mathcal{N}_S|} \sum_{l \in \mathcal{N}_S} V_{f,t}^{(l)} \frac{\sum_{z \in z_C} P_C(f|z) P_C(t|z) P^{(l)}(z)}{\sum_{z \in z_C \cup z_I^{(l)}} P_C(f|z) P_C(t|z) P^{(l)}(z)}. \tag{5.18}$$

We refresh $\mathcal{N}_S$ at every iteration, and therefore, S should be updated before that, too. Note that (5.18) is suboptimal, because the amount of the contribution from each recording to the final result is not known.

The proposed two-stage method

We define F, T, Z, and L to denote the number of rows, columns, components, and recordings, respectively. Then, the computational complexity of an EM update in (5.12)-(5.15) is in the order of $\mathcal{O}\big(FTZ|\mathcal{N}_S|\big)$. In original PLCS $\mathcal{N}_S = L$, but in the proposed systems we set $|\mathcal{N}_S| < L$, so that the use of $\mathcal{N}_S$ is beneficial. The complexity for the construction of the neighbor set in (5.17) and the source in (5.18) are $\mathcal{O}(FTL)$ and $\mathcal{O}\big(FT|\mathcal{N}_S|\big)$, respectively.

In this section we propose a two-stage neighborhood search method that further decreases the complexity of (5.17) from $\mathcal{O}(FTL)$ down to $\mathcal{O}\big(FT|\mathcal{N}_H|\big)$, by introducing a set of candidate neighbors $\mathcal{N}_H$, where $|\mathcal{N}_S| < |\mathcal{N}_H| < L$ and $\mathcal{N}_S \subset \mathcal{N}_H$. Construction of $\mathcal{N}_H$ is cheaper, because we use binary operations.

To this end, we propose to hash all the recordings into a binary representation that can be more efficient in some arithmetic operations. Any hash function can be used if it meets the conditions for the family of locality sensitive hashing: (a) originally closer data points are more probable to collide into the same hash code (b) Hamming distance between the codes approximates the original distance metric [39]. We are particularly interested in WTA hashing, which also holds those properties [37, 38].

In the recent application of WTA hashing for speeding up the sparse encoding process of dictionary-based source separation, a few candidates of the overcomplete dictionary items are selected based on the WTA Hamming distance in the first place, and then the selection is refined using cross entropy [52]. There are pros and cons of the two searches. Cross entropy is more accurate, yet costly due to the floating-point operations and logarithm. On the other hand, Hamming distance is cheap to calculate thanks to the binary representation of the hash codes, while inaccurate. Therefore, the two-stage search on the dictionary consists of (a) the first round that constructs a bigger candidate set, e.g. $3K$ items, using Hamming distance (b) the second round only on the $3K$ candidates rather than the whole dictionary based on cross entropy, where $K = |\mathcal{N}_S|$.

A WTA hash code is generated by randomly choosing M elements out of the input vector, and then writing down the index of the maximum among them by flipping the corresponding element of an all-zero

binary vector of length M as the indicator. By repeating this experiment Q times, the total bits used for an input vector are QM bits[2]. Hamming distance among the two binary hash codes $x, y \in \mathbb{B}^{QM\times 1}$ can be calculated by a bitwise AND operation followed by bit counting, i.e. $\sum_i x_i \wedge y_i$, and then inverting the result. Since this procedure approximates a rank order metric, a relative ordering of the input elements, it can still be a discriminative feature even with its binary representations. On the other hand, the mismatch between Hamming distance and cross entropy hinders WTA hash codes from replacing the original distance measure.

We start from hashing all the recordings in advance by using the WTA hash function ϕ: $\tilde{v}^l \leftarrow \phi(\mathrm{vec}(V^l))$, where vec() vectorizes a matrix. Now after every source update (5.18), we update the source' hash code as well: $\tilde{s} \leftarrow \phi(\mathrm{vec}(S))$. In the first-round search, using this new hash code along with the already prepared ones for the recordings, we calculate another candidate set $\mathcal{N}_H$, which meets an inequality as follows:

$$\sum_n^{QM} \left(\tilde{v}_n^{l\in\mathcal{N}_H} \wedge \tilde{s}_n\right) > \sum_n^{QM} \left(\tilde{v}_n^{l'\notin\mathcal{N}_H} \wedge \tilde{s}_n\right), \tag{5.19}$$

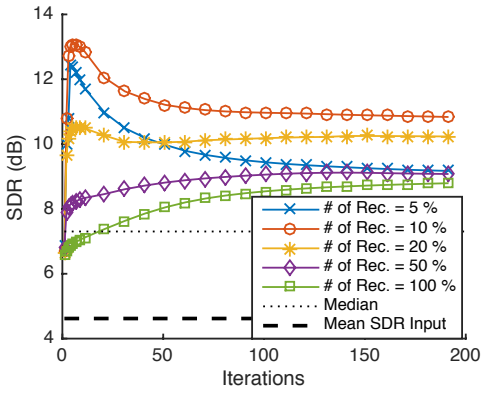
Now that we have a candidate set, the second search is limited only on $\mathcal{N}_H$ rather than all the L recordings. In other words, we still do the search based on (5.17), but now the inequality is guaranteed only in the candidate set $l, l' \in \mathcal{N}_H$, assuming $\mathcal{N}_S \subset \mathcal{N}_H$.

5.2.3 Experiments

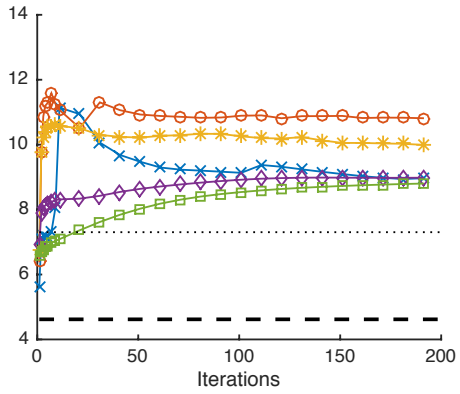
For experimental validation, we simulate an audio scene where 30 sources are randomly scattered in a square space of 50 meters by 50 meters. Among the sources, one speech source is randomly chosen and placed at the center with 20dB louder volume than the others. 30 sources consist of 10 male and 10 female TIMIT speech signals, and 10 non-stationary noise signals used in [76]. All sources are either cut to 10 seconds long if longer than that or repeated otherwise, and sampled at 16KHz. $L = \{100, 500, 1000\}$ are the number of sensors that are randomly placed. Therefore, a recording is a mixture of all the sources with different sound pressure levels depending on the distance between the source and the sensor. This is a challenging setup that models real-world situations because the sensors far from the dominant source can exhibit significantly louder interference. For now we assume that the signals are already aligned, mixing is instantaneous, and no additional artifacts are present.

A short-time Fourier transform is done with 1024 pt Hann windowing with 75% overlap. We set up 50 components for sharing while 10 others are assigned per recording to capture interferences. The a prior

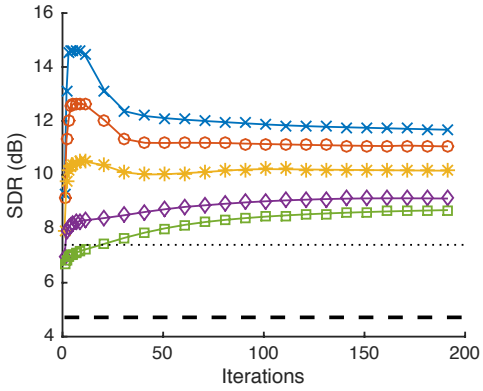
[2]For example, $3 = 0100$ when $M = 4$, while in the usual binary representation with $\lceil \log_2(M) \rceil$ bits per integer, $3 = 11$. This consumes more bits, but the Hamming distance calculation is more convenient.



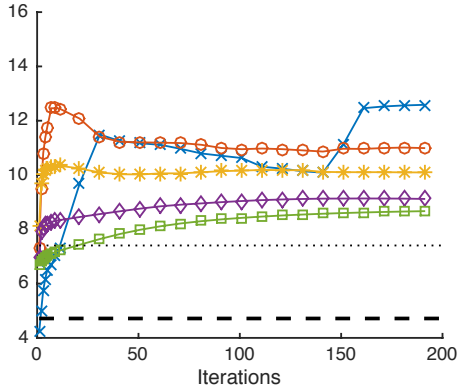
(a) L=100; Cross entropy only



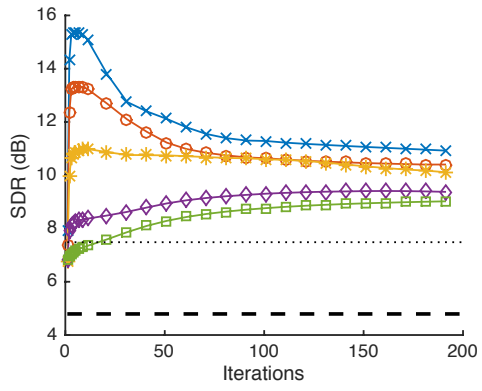
(b) L=100; Two-stage



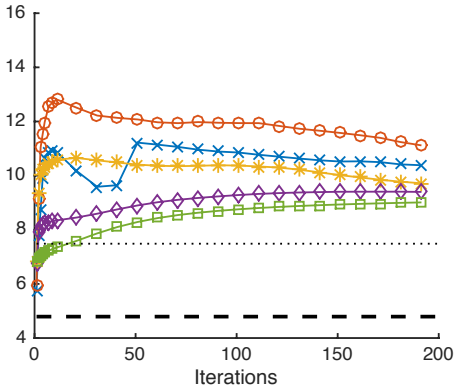
(c) L=500; Cross entropy only



(d) L=500; Two-stage



(e) L=1000; Cross entropy only



(f) L=1000; Two-stage

Figure 5.6: Average SDR performances of the systems with different numbers of input signals and nearest neighboring measures.

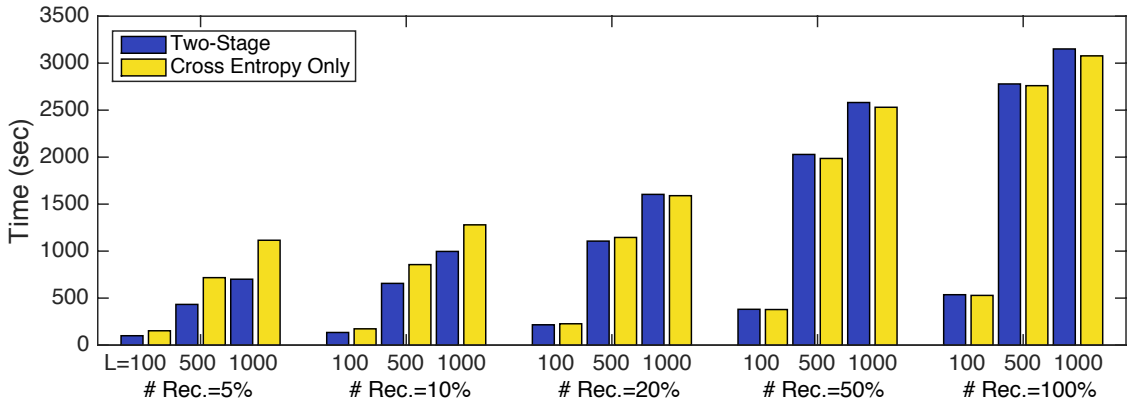


Figure 5.7: Run-time analysis of the PLCS system and the proposed neighborhood-based methods.

bases α for the common components were trained from 20 different female speakers using an ordinary PLSI. We choose a big number, e.g. 5000, to initialize β, and decay it exponentially during the updates. We change the number of neighbors to be one of $|\mathcal{N}_S| = L \times \{0.05, 0.1, 0.2, 0.5, 1\}$. The number of WTA candidates are set to be $|\mathcal{N}_H| = \min(3|\mathcal{N}_S|, L)$. WTA parameters Q and M are set to be $\lceil \log_2 FT \rceil = 2^{19}$ and 2, respectively.

Each sub-figure in Figure 5.6 is an average of five repeated experiments using five different choices of the dominant female source, and randomized geometric configurations of the other sources and sensors accordingly. SDR was measured up to 200 EM iterations as an overall separation quality score. First thing we notice in (a), (c), and (e) is that the neighbor set in terms of cross entropy successfully enhances the performance using only a small set of nearest neighbors: all the choices using this neighborhood concept converge to a better solution earlier than the full PLCS that uses 100% of the recordings (green squares). When there are enough number of recordings (500 or 1000), only 5% of them are needed to obtain the best results (blue crosses). All the systems including the full PLCS perform better than the average of the input SDRs, or the SDR of a the median spectrogram of all recordings, i.e. $V_{f,t}^{\text{median}} = \text{median}([V_{f,t}^1, V_{f,t}^2, \cdots V_{f,t}^L])$.

Two-stage methods in (b), (d), and (f) also show on average better performance than the full PLCS model or the median spectrogram of the recordings. However, it is noticeable that its convergence is not stable when only 5% are used. We believe that this fluctuation can be mitigated when we increase the size of $\mathcal{N}_H$, but it comes with the cost of increased run-times. 10% and 20% cases are more stable than the 5% case and comparable to their cross entropy counterparts.

Figure 5.7 compares the average run-times of all the cases. If the neighbor sets are reasonably small (5% or 10% in the first two bar groups), we see a speed-up by using the two-stage method. Furthermore, the gap becomes larger as L increases. However, if $|\mathcal{N}_S|$ is too big ($\geq$ 20%) the two-stage method starts to add

more overhead than the desired speed-up. Overall, both neighborhood methods with a reasonably smaller neighborhood set, $\mathcal{N}_S \leq 0.2L$, reduce the run-times down to from around 16% to 50% of the the full PLCS model. It is a significant saving given the fact that our MATLAB-based implementation is not well-suited for a fair comparison in this case, penalizing the efficiency of bitwise operations.

5.2.4 Summary

We proposed a neighborhood-based extension for the PLCS model to handle a large number of recordings that can be found in abundance through social data. We proposed two different neighbor search schemes, one using the comprehensive cross entropy between a tentative source spectrogram and the noisy recordings, and the other first reduces the set down to some candidates based on WTA hash codes followed by the comprehensive search only on those candidates. Experimental results clearly supported the merit of the proposed methods in terms of separation performances, convergence behaviors, and run-times. Although separation was successful, the robustness of the neighborhood-based extension to the other types of recording artifacts, such as band-pass filtering, clipping, reverberation, etc, is not shown here, and we leave it for future work.

Chapter 6

Bitwise Neural Networks

6.1 Introduction

During the last decade, artificial neural networks regained a lot of attention in machine learning thanks to the discovery that greedy layer-wise training schemes can help train deeper networks efficiently [78]. According to the universal approximation theorem, a single hidden layer with a finite number of units can approximate a continuous function with some mild assumptions [79, 80]. While this theorem implies a shallow network with a potentially intractable number of hidden units when it comes to modeling a complicated function, Deep Neural Networks (DNN) achieve the goal by learning a hierarchy of features in their multiple layers [81].

Although DNNs are extending the state-of-the-art results for various tasks, such as image classification [82], speech recognition [83], speech enhancement [84], etc, it is also the case that the relatively bigger networks with more parameters than before call for more resources (processing power, memory, battery time, etc), which are sometimes critically constrained in applications running on embedded devices. Examples of those applications span from context-aware computing, where a device collects and analyzes a variety of sensor signals [85], to always-on computer vision applications (e.g. on Google glasses), to keyword spotting to start up speech-driven personal assistant services, such as "Hey, Siri" and "OK, Google." A primary concern that hinders those applications from being more successful is that they assume an always-on pattern recognition engine on the device, which will drain the battery fast unless it is carefully implemented to minimize the use of resources. Additionally, even in an environment with the necessary resources being available, speeding up a DNN can greatly improve the user experience when it comes to tasks like searching big databases [86]. In either case, a more compact yet well-performing DNN is a welcome improvement.

Efficient computational structures for deploying artificial neural networks have long been studied in the literature. Most of the effort has been focused on training a set of quantized weights with a minimal loss of performance. One of the typical ways to train a network with this consideration is to utilize the

Parts of this chapter have been published in [77].

quantized weights during training by using quantized weights in the feedforward step instead of the continuous ones. In this way, the network is aware of the particular quantization scheme and is robust to the known quantization noise caused by a limited precision, while a naïve quantization on the trained network parameters often results in worse-performing networks [87, 88]. It was also shown that 10 bits and 12 bits are enough to represent gradients and storing weights for implementing the state-of-the-art maxout networks even for training the network [89]. However, in those quantized networks one still needs to employ the usual arithmetic operations, such as multiplication and addition, on fixed-point values. Even though faster than floating point, they still require relatively complex logic and can consume a lot of power.

With the proposed Bitwise Neural Networks (BNN), we take a more extreme view that every input node, output node, and weight, is represented by a single bit. For example, a weight matrix between two hidden layers of 1024 units is a 1024×1025 matrix of binaries rather than quantized real values (including the bias). Although learning those bitwise weights as a Boolean concept is an NP-complete problem [90], the fully bitwise networks have been studied in the limited setting, such as μ-perceptron networks where an input node is allowed to be connected to one and only one hidden node and its final layer is a union of those hidden nodes [91]. A more practical network was proposed in [92] recently, where the posterior probabilities of the weight values were sought using the Expectation Back Propagation (EBP) scheme, which is similar to backpropagation in its form, but has some advantages, such as parameter-free learning and a straightforward discretization of the weights. Its promising results on binary text classification tasks however, rely on the real-valued bias terms and averaging of predictions from differently sampled parameters.

This chapter presents a completely bitwise network where all participating variables are bipolar binaries. Therefore, in its feedforward only XNOR and bit counting operations are used instead of multiplication, addition, and a nonlinear activation on floating or fixed-point variables. For training, we propose a two-stage approach, whose first round is typical network training with a weight compression technique that helps the real-valued model easily be converted into a BNN. To train the actual BNN in the second round, we use those compressed weights to initialize the BNN parameters, and do noisy backpropagation based on the tentative bitwise parameters. Since we also assume that an input node takes only a single-bit input value, we need to a quality binary feature extraction technique, e.g. fixed-point representations and hash codes. Regardless of the binarization scheme, each input node is given only a single bit at a time, as opposed to a bit packet representing a fixed-point number. This is significantly different from the networks with quantized inputs, where a real-valued signal is quantized into a set of bits, and then all those bits are fed to an input node in place of their corresponding single real value. Lastly, we apply the sign function

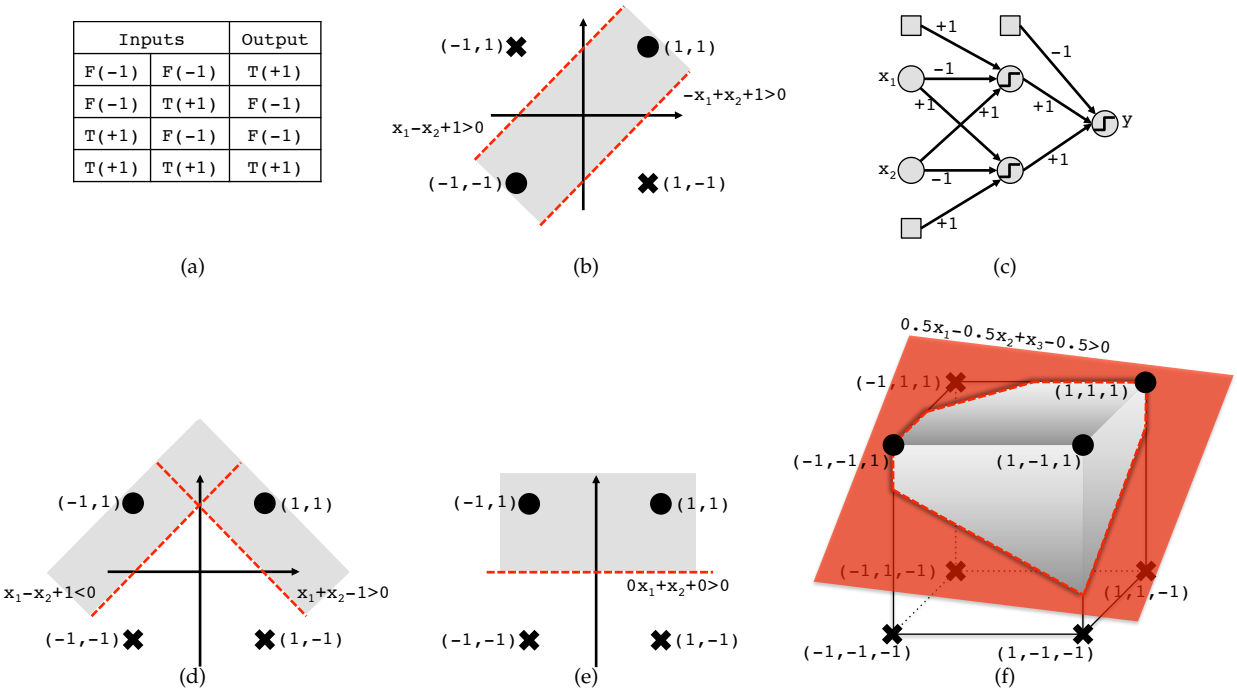


Figure 6.1: (a) An XNOR table. (b) The XOR problem that needs two hyperplanes. (c) A multi-layer perceptron that solves the XOR problem. (d) A linearly separable problem while bitwise networks need two hyperplanes to solve it. (e) A bitwise network with sparsity that solves the problem with a single hyperplane. (f) Another linearly separable case with real-valued coefficients ($0.5x_1 - 0.5x_2 + x_3 - 0.5 > 0$) that however needs more than one hyperplanes in BNN.

as our activation function instead of a sigmoid to make sure the input to the next layer is bipolar binary as well. We compare the performance of the proposed BNN with its corresponding ordinary real-valued networks on hand-written digit recognition and phoneme classification tasks, and show that the bitwise operations can do the job with an acceptably small performance loss, while providing a large margin of improvement in terms of the necessary computational resources.

6.2 Feedforward in Bitwise Neural Networks (BNN)

6.2.1 Notations and setup: bipolar binaries and sparsity

Throughout the chapter, we use bipolar binaries where the two Boolean values, "true" and "false", are represented as $+1$ and -1 rather than 1 and 0. This representation is useful, because a particular Boolean operation, XNOR, which we denote with $\otimes$, is equivalent to multiplication on the bipolar binary inputs (see Figure 6.1 (a) for an XNOR table). For example, without the bipolar representation, the multiplication between 0 and 0 is 0, while their XNOR should be "true", or 1. Bipolar binaries are free from this mismatch,

whose multiplies match XNOR results, e.g. $(-1) \times (-1) = +1$.

The bipolar binaries are more expressive as well. For example, in Figure 6.1 (b), the two hyperplanes (red dashes) can be fully defined with bipolar binary coefficients and biases, while 0-1 binaries call for a real-valued bias to separate the corresponding 0-1 binary inputs.

Finally, we can make use of zeros to explain the sparsity concept by representing "false" with -1. Similarly to the real-valued weight matrices that are often redundant, the fully bitwise coefficients can be less efficient as well. By introducing the sparsity in the coefficient domain, we can *turn off* some dimensions that are redundant. Since our coefficients are either $+1$ or -1, zeros are natural choice to express inactivity.

Note that these bipolar bits will in practice (e.g. in hardware chips) be implemented using $0/1$ binary values, where the activation in (6.2) is equivalent to counting the number of 1's and then checking if the accumulation is bigger than the half of the number of input units plus 1. With no loss of generality, in this chapter we will use the ± 1 bipolar representation.

6.2.2 The feedforward process

It has long been known that any Boolean function, which takes binary values as input and produces binary outputs as well, can be represented as a bitwise network with one hidden layer [93], for example, by merely memorizing all the possible mappings between input and output patterns. We define the forward propagation procedure as follows based on the assumption that we have trained such a network with bipolar binary parameters:

$$a_i^l = b_i^l + \sum_j^{K^{l-1}} w_{i,j}^l \otimes z_j^{l-1}, \tag{6.1}$$

$$z_i^l = \text{sign}(a_i^l), \tag{6.2}$$

$$\mathbf{z}^l \in \mathbb{B}^{K^l}, \mathbf{W}^l \in \mathbb{B}^{K^l \times K^{l-1}}, \mathbf{b}^l \in \mathbb{B}^{K^l}, \mathbf{a}^l \in \mathbb{Z}^{K^l} \tag{6.3}$$

where $\mathbb{B}$ is the set of bipolar binaries, i.e. ± 1, and $\otimes$ stands for the bitwise XNOR operation. l, j, and i indicate a layer, input, and output units, respectively. We use bold characters for a vector (or a matrix if capicalized). K^l is the number of input units at l-th layer. Therefore, $\mathbf{z}^0$ equals to an input vector, where we omit the sample index for the notational convenience. We use the sign activation function to generate the bipolar outputs. The input to the activation function a_i^l is an integer whose value can be from $-K^{l-1} - 1$ to $K^{l-1} + 1$.

We can check the prediction error $\mathcal{E}$ by measuring the bitwise agreement of target vector $\mathbf{t}$ and the

output units of L-th layer using XNOR as a multiplication operator,

$$\mathcal{E} = \sum_{i}^{K^{L+1}} \left(1 - t_i \otimes z_i^{L+1}\right)/2, \tag{6.4}$$

but this error function can be tentatively replaced by involving a softmax layer during the training phase.

The XNOR operation is a faster substitute of binary multiplication. Therefore, (6.1) and (6.2) can be seen as a special version of the ordinary feedforward step that only works when the inputs, weights, and bias are all bipolar binaries.

6.2.3 Linear separability and bitwise hyperplanes

Sometimes a BNN can solve the same problem as a real-valued network without any size modifications, but in general we should expect that a BNN could require larger network structures than a real-valued one. For example, the XOR problem in Figure 6.1 (b) can have an infinite number of solutions with real-valued parameters once a pair of hyperplanes can successfully discriminate $(1, 1)$ and $(-1, -1)$ from $(1, -1)$ and $(-1, 1)$. Among all the possible solutions, we can see that binary weights and bias are enough to define the hyperplanes, $x_1 - x_2 + 1 > 0$ and $-x_1 + x_2 + 1 > 0$ (red dashes). Likewise, the particular BNN defined in (c) has the same separability once the inputs are binary as well.

Figure 6.1 (d) shows another example where BNN needs more hyperplanes than a real-valued network. Only one proper hyperplane is enough to solve this linearly separable problem, for example $-0.1x_1 + x_2 + 0.5 > 0$, but it is impossible to describe such a hyperplane with binary coefficients. We can instead come up with a solution by combining multiple binary hyperplanes that will eventually increase the perceived complexity of the model. However, even with a larger number of nodes, the BNN is not necessarily more complex than the smaller real-valued network. This is because a parameter or a node of BNN requires only one bit while a real-valued node generally requires more than that, up to 64 bits. Moreover, the simple XNOR and bit counting operations of BNN bypass the computational complications of a real-valued system, such as the power and spatial consumption of multipliers and adders for the floating-point operations[1], various dynamic ranges of the fixed-point representations, erroneous flips of the most significant bits, etc.

One way to reduce the model complexity of BNNs is to introduce sparsity on the bitwise parameters. For example, for an inactive element in the weight matrix $\mathbf{W}$ due to the sparsity, we can simply ignore the computation for it similarly to the operations on the sparse representations. In this way, we do not

[1] In our preliminary hardware simulation, an XNOR-based inner product for the activation of a hidden unit uses less than 1.5% of space and power than for a corresponding 64bit floating-point unit.

have to consume an additional bit to encode the parameter activity, while there can be an overhead to keep the address of the active parameters. Or, it is always possible to assign an additional bit to tell if the parameter is active, calling for 2bits per parameter in the worst case. Conceptually, we can say that those inactive weights serve as zero weights, so that a BNN can solve the problem in Figure 6.1 (d) by using only one hyperplane as in (e). From now on, we will use this extended version of BNN with inactive weights, yet there are some cases where BNN needs more hyperplanes than a real-valued network even with the sparsity, such as in Figure 6.1 (f).

6.3 Training BNN

To train BNN, we first train a corresponding real-valued network just to use its weights to initialize the subsequent BNN parameters. To better match the dynamic range of the real-valued weights to the ones in BNN, we use the weight compression technique. In second phase of the training we finally learn the bitwise weights using noisy backpropagation and some regularization.

6.3.1 The first round: real-valued networks with weight compression

First, we train a real-valued network that takes either bitwise inputs or real-valued inputs ranged between -1 and $+1$. A special part of this network is that we constrain the weights to have values between -1 and $+1$ as well by wrapping them with the hyperbolic tangent function, $\tanh(\cdot)$. Similarly, if we choose $\tanh(\cdot)$ for the activation, we can say that the network is a relaxed version of the corresponding bipolar BNN. With this weight compression technique, the relaxed forward pass during training is defined as follows:

$$\bar{a}_i^l = \tanh(\bar{b}_i^l) + \sum_j^{K^{l-1}} \tanh(\bar{w}_{i,j}^l)\bar{z}_j^{l-1}, \tag{6.5}$$

$$\bar{z}_i^l = \tanh\left(\bar{a}_i^l\right), \tag{6.6}$$

$$\bar{\mathbf{z}}^l \in \mathbb{R}^{K^l}, \bar{\mathbf{W}}^l \in \mathbb{R}^{K^l \times K^{l-1}}, \bar{\mathbf{b}}^l \in \mathbb{R}^{K^l}, \bar{\mathbf{a}}^l \in \mathbb{R}^{K^l}, \tag{6.7}$$

where all the binary values and the integer sum in (6.1) and (6.2) are real for the time being. The bars on top of the notations are for the distinction.

Weight compression requires some changes in the backpropagation procedure. In a hidden layer we

calculate the error for the n-th training sample,

$$\delta_j^l(n) = \Big(\sum_i^{K^{l+1}} \tanh(\bar{w}_{i,j}^{l+1}) \delta_i^{l+1}(n) \Big) \cdot \Big(1 - \tanh^2 \left(\bar{a}_j^l \right) \Big). \tag{6.8}$$

Note that the errors from the next layer are multiplied with the compressed versions of the weights. Hence, the gradients of the parameters in the case of batch learning are

$$\nabla \bar{w}_{i,j}^l = \Big(\sum_n \delta_i^l(n) \bar{z}_j^{l-1} \Big) \cdot \Big(1 - \tanh^2 \left(\bar{w}_{i,j}^l \right) \Big), \tag{6.9}$$

$$\nabla \bar{b}_i^l = \Big(\sum_n \delta_i^l(n) \Big) \cdot \Big(1 - \tanh^2 \left(\bar{b}_i^l \right) \Big), \tag{6.10}$$

with the additional term from the chain rule on the compressed weights.

6.3.2 The second round: training BNN with noisy backpropagation and regularization

Now that we have trained a real-valued network with a proper range of weights, the next step is to train the actual bitwise network. The training procedure is similar to the ones with quantized weights [87, 88], but now all the input signals and weights are constrained to be bipolar binaries, so that the operations on them are bitwise. The full binary setting is of particular importance since it can avoid the computations in-between fixed-point variables that are common forms after a quantization procedure. Furthermore, we regularize the parameters so that they are more likely to have extreme values rather than being centered around zeros.

To this end, we first initialize all the real-valued parameters, $\bar{\mathbf{W}}$ and $\bar{\mathbf{b}}$, with the ones learned from section 6.3.1. Then, we set a sparsity parameter ρ which says the proportion of the zeros after the binarization. Therefore, ρ decides the boundaries β, which is used to divide the parameters into three groups: $+1$, 0, or -1, for example, $w_{ij}^l = -1$ if $\bar{w}_{ij}^l < -\beta$. Note that the number of zero weights, $|\bar{w}_{ij}^l| < \beta$, at l-th layer equals to $\rho K^l K^{l-1}$.

In accordance with this relaxation, the training error should be defined with multiplications between real values,

$$\mathcal{E} = \sum_i^{K^{L+1}} (1 - t_i z_i^{L+1})/2. \tag{6.11}$$

$\mathbf{W}^l$ and $\mathbf{b}^l$ are further regularized to have more extreme values away from zero, which is sometimes difficult to do by relying only on the sigmoid functions. We use $\ell 2$-norm regularization for this. Along with

the error function defined in (6.11), the final objective function $\mathcal{J}$ becomes,

$$\mathcal{J} = \sum_n \mathcal{E}(n) - \lambda \sum_{l=1}^{L+1} \sum_i^{K^l} \Big({b_i^l}^2 + \sum_j^{K^{l-1}} {w_{i,j}^l}^2 \Big),$$

where $\mathcal{E}(n)$ is now with the sample index n. λ controls the degree of the regularization. Note that the regularization term is with minus sign, since we penalize too small parameters.

The main idea of this second training phase is to let the network know about the binarization. This is done by binarizing the weights after a backpropagation step and feedforward using the bitwise operations on them as in (6.1) and (6.2). Also, the binarized parameters participate in the noisy backpropagation as well, where we calculate the errors and gradients using those binarized weights and signals. In the output layer we calculate the error,

$$\delta_j^{L+1}(n) = \big(-t_j/2 \big) \cdot \Big(1 - \tanh^2 \big(a_j^{L+1} \big) \Big), \tag{6.12}$$

while the other errors and gradients in l-th layer are found by

$$\delta_j^l(n) = \Big(\sum_i^{K^{l+1}} w_{i,j}^{l+1} \delta_i^{l+1}(n) \Big) \cdot \Big(1 - \tanh^2 \big(a_j^l \big) \Big), \tag{6.13}$$

$$\nabla \bar{w}_{i,j}^l = \sum_n \delta_i^l(n) z_j^{l-1}, \tag{6.14}$$

$$\nabla \bar{b}_i^l = \sum_n \delta_i^l(n). \tag{6.15}$$

Note that in the calculation the parameters without a bar on the top, i.e. the bitwise parameters, are used. In this way, the gradients and errors properly take the binarization of the weights and the signals into account. Also, the non-differentiable sign activation function is relaxed with tanh. Since the gradients can get too small to update the binary parameters $\mathbf{W}$ and $\mathbf{b}$, we instead update their corresponding real-valued parameters,

$$\bar{w}_{i,j}^l \leftarrow \bar{w}_{i,j}^l - \eta \nabla \bar{w}_{i,j}^l, \quad \bar{b}_{i,j}^l \leftarrow \bar{b}_i^l - \eta \nabla \bar{b}_i^l, \tag{6.16}$$

with η as a learning rate parameter. Finally, at the end of each update we binarize the parameters again with β:

$$w_{i,j}^l = \begin{cases} +1 & \text{if } \bar{w}_{i,j}^l > \beta \\ -1 & \text{if } \bar{w}_{i,j}^l \leq -\beta \\ 0 & \text{otherwise} \end{cases} \tag{6.17}$$

We repeat this procedure at every epoch.

6.3.3 Adjustment for classification

In case the target variable is not binary, a different type of the error function will be needed instead of the bitwise agreement in (6.4) and (6.11). Since classification with multiple mutually exclusive classes is the main application of this chapter, a softmax output layer is a common choice, where the output layer for training is defined by

$$z_c^{L+1} = \frac{\exp(a_c^{L+1})}{\sum_{c'=1}^{C} \exp(a_{c'}^{L+1})}, \tag{6.18}$$

which is to replace (6.6) when $l = L + 1$. We use c to indicate one of the C possible classes instead of K^{L+1} notation. Then, we replace the training error (6.11) with cross-entropy between the target with the 1-of-C coding scheme[2] and the output as follows:

$$\mathcal{E}(n) = -\sum_{c=1}^{C} t_c(n) \ln \left(\frac{z_c^{L+1}(n)}{t_c(n)} \right). \tag{6.19}$$

Consequently, the error of the output layer (6.12) becomes

$$\delta_c^{L+1}(n) = z_c^{L+1}(n) - t_c(n). \tag{6.20}$$

6.3.4 Dropout

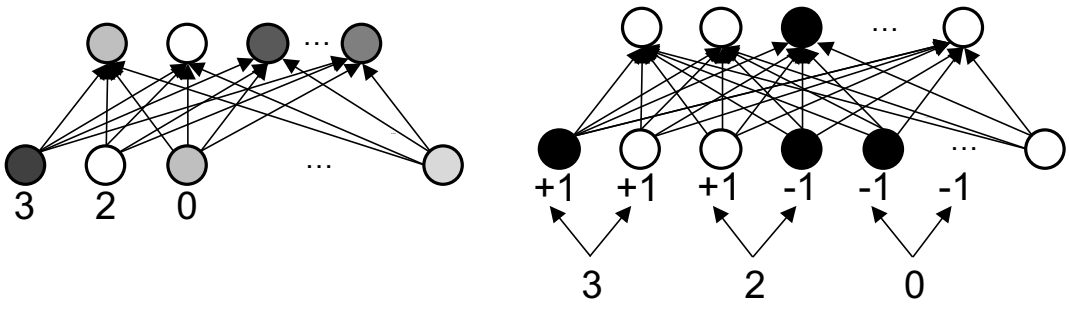
We found that a dropout scheme is helpful for training BNN [94]. During testing, in general it is suggested to reduce the weights by multiplying the Bernoulli parameter to cancel out the effect of increased weights to compensate dropouts, while actually not applying dropout to the test units. This weight re-scaling effect, however, is absolved in the binarization process in BNN and has no effect in test accuracies.

6.3.5 Quantization-and-dispersion for binarization

In a rigorous definition of BNN, we assume all the input and output signals are bit-patterns, too. However, sometimes we need to deal with continuous signals either as the input or output of the network. In those cases, we need to come up with a binarization technique that can convert a continuous value into a bit string.

An obvious discretization can be population-based quantization, such as Lloyd-Max's quantization [95], where we can minimize the mean-squared error before and after the quantization. If we were to use these quantized values as the input, we can simply convert the d-dimensional real-valued vector into a

[2] With this coding scheme a target vector is all zero, but only c-th element is 1 to indicate the class the sample belongs to.



(a) (b)

Figure 6.2: (a) A network with an integer input vector (b) the quantization-and-dispersion technique for the bitwise input layer.

d-dimensional vector of integers, as shown in 6.2 (a). With this quantized input, the network does not have to change its structure, although the Boolean algebra and geometric understanding we have relied on does not stand anymore.

The dispersion technique can convert this ordinary network with integer input vectors into a fully bitwise network. During the dispersion step, after the quantization, we treat each bit of the fixed-point quantized value as a bitwise input feature. For example, as for the first three integer values of the input vector in 6.2 (a), 3, 2, and 0, which we quantized with 2 bits per value, we disperse the bits to input units, so that an input unit takes a single bit rather than an integer. In this way, instead of using the 2bit-long integer as a low-precision substitute for the corresponding real-valued feature, we disperse the 2 bits into 2 corresponding input nodes (see 6.2 (b)). A downside of this dispersion technique is the fact that we have to enlarge the size of the input layer N times bigger than usual, where N is the number of bits.

Note that we can also convert the target signal into bit strings by using the same technique, although we may want to weigh more on the most significant bits when we setup the error function. We can also rely on some hashing techniques that preserves the original similarity between the data points in the hash code domain, such as spectral hashing [96], semantic hashing [86], and WTA hashing [37, 38]. We leave this direction of research to future work.

Table 6.1: Classification errors for real-valued and bitwise networks on the bipolarized MNIST dataset.

NETWORKS	3×1024	3×2048	2×4096	2×8192
After the first round training (64-bit floating point)	1.25%	1.15%	1.14%	1.08%
BNN after the second round training	1.43%	1.33%	1.30%	1.21%
Dropout network (ReLU, max-norm) [94]	1.06%	1.04%	1.01%	0.95%

6.4 Experiments

6.4.1 Hand-written digit recognition on the MNIST dataset

In this section we go over the details and results of the hand-written digit recognition task on the MNIST data set [50] using the proposed BNN system.

From the first round of training, we get a regular dropout network with the same setting suggested in [94], except the fact that we used the hyperbolic tangent for both weight compression and activation to make the network suitable for initializing the following bipolar bitwise network. We stopped at the number of iterations from 500 to $1,000$, although it is recommended to iterate more (e.g. up to a million updates) in the original dropout network setting. The first row of Table 6.1 shows the performance of the first round real-valued network with 64-bit floating-point parameters. As for the input to the first round, we rescale the pixel intensities into the bipolar range, i.e. from -1 to $+1$, for the first round, and then apply the sign function to make them bipolar binaries as the second round input.

Now we train the new BNN with the noisy backpropagation technique as described in section 6.3.2. The second row of Table 6.1 shows the BNN results. We see that the bitwise networks perform well with very small additional errors, less than 0.2% point. Compared with the performance of the state-of-the-art dropout networks with better activation function (rectified linear units) with similar network topology, the proposed bitwise network is no worse than 0.3% point with enough hidden units (except the case with 1024 units where the performance gap is 0.37%).

For the MNIST experiments, we set $\lambda = 0$ since the proposed regularization has no significant effects. The sparsity was set to be $\rho = 0.8$. We use SGD to train both rounds with minibatch size 100 with a fixed learning rate $\eta = 1 \times 10^{-5}$ for the first round and a smaller learning rate $\eta = 1 \times 10^{-8}$ for the second round.

Table 6.2: Frame-wise phoneme classification errors for real-valued and bitwise networks on the TIMIT dataset.

NETWORKS	3×1024	3×2048	5×1024	5×2048
Baseline dropout network	43.32%	41.59%	43.73%	41.53%
After the first round training (on 4bit-quantized MFCC)	47.91%	44.92%	–	–
BNN after the second round training (on 4bit-quantized MFCC)	58.33%	46.34%	–	–

6.4.2 Phoneme Classification on the TIMIT dataset

We also evaluate the performance of BNN for phoneme classification tasks using TIMIT dataset. As suggested in [97, 98], we rearrange the English phonemes down to 39 classes. We also exclude "sa" sentences that are common in all the speakers, and use the rest of 3,696 train utterances. Instead of training the network with the features that reflect temporal patterns of the phones, we simply perform the classification in a frame-by-frame manner, where each frame is 250 ms long with 100 ms hop size. Therefore, a drop in the classification performance is expected compared to the ones with temporal dynamics. After discarding the too many silent frames except 3,000 of them, we finally collect 673,432 frames for training, 168,359 for validation, and 311,465 for testing.

Instead of working on the time domain signals, we convert each frame into the MFCC as our real-valued features. MFCCs have been standard features in speech processing, and we followed common settings for the transform, such as 13 features plus their delta and delta-delta. As a result, each frame is now with 39 features, which we further normalize to have zero mean and unit variance in all dimensions. Since MFCCs are already well-defined for the use in speech processing, they outperform features from unsupervised deep learning techniques, such as Convolutional Deep Belief Networks (CDBN), by 15.2% [98]. Therefore, we assume that extracting better binary features than MFCCs is a difficult job, making our comparison somewhat unfair for the BNN. Instead of investigating those options, we applied the quantization-and-dispersion technique discussed earlier in section 6.3.5. More specifically, we quantize an MFCC coefficient into 4 bits, and then we disperse the 4 bits into 4 corresponding input nodes. Therefore, the BNN takes $4 \times 39 = 156$ bitwise inputs, making its input layer 4 times bigger than usual.

Table 6.2 shows the classification errors of the systems. The first row is from a usual real-valued dropout network on the real-valued MFCC inputs as they are. Since we did not observe a significant performance improvement by adding more hidden layers, we tested BNNs on the 3-layer case only. The second and third rows take the quantized, but dispersed bit patterns as the input. Therefore, their input layers have

156 units plus a bias term, rather than 39 units. In the second row, we see some performance drop compared to the first row, which is mainly caused by the loss of information during the quantization, around $4 \pm 0.5\%$ point.

If the network is small only with 1024 units per layer, the performance drop after the second and final round training is rather significant (10.42 % point). However, BNN catches up the gap in the bigger network with 2048 hidden units, showcasing better performance than a real-valued 1024-unit network. Overall, we can see that the baseline networks on MFCCs apparently outperform BNN on the bitwise inputs by less than 5% point in the bigger networks. However, the BNN results are still promising considering the fact that its binarized parameters and their processing are cheap enough to compensate for its bigger network size. For instance, a 3×2048 BNN could be cheaper and faster than a 3×1024 real-valued network.

For the TIMIT experiments, we found that the regularization ($\lambda = 10$) improves the results. The sparsity was set to be $\rho = 0.7$, and the minibatch size was 1000. Attenuating the learning rate gradually as suggested in [94] helped the convergence in the second round.

6.4.3 Speech Enhancement Using BNN

In recent speech enhancement research, neural networks gained a lot of attention as a mechanism that takes a mixture (or noisy) input spectrum and predicts its cleaned-up version. There are a lot of different choices for the input and target signals, such as raw Fourier magnitudes [84, 99, 100, 27, 101], cochleagram [102], MFCC [103, 104], etc, for input features, and raw Fourier magnitudes [84, 99, 100, 27, 101], Ideal Binary Mask (IBM) [102, 105], Ideal Binary Mask (IRM) [103, 104], and phase-informed targets [106] for the target variables, respectively. In this section, for the speech enhancement task we use the quantization-and-dispersion technique proposed in section 6.3.5 for the input signal binarization. As for the output layer, we simply use IBM as the target variables, which is a natural way to turn the problem into a binary classification task.

In Figure 6.3 (a), we see an example of applying IBM on the mixture spectrum. With IBM we can think of the problem as a prediction task that tries to estimate which frequency bin belongs to which source: speech or noise. Since IBM can be easily calculated by comparing the magnitudes of the sources, e.g. IBM is 1 if the magnitude of speech is bigger than noise for that frequency bin, we can construct all the necessary pairs of input and output vectors for training if we knew the source spectra for training in advance. Eventually, the masking procedure corresponds to multiplying the IBM values (red dash in Figure 6.3 (a)) to the input magnitudes to turn off some unwanted frequency bins. Although this can be seen as a hard decision procedure, it works well in practice thanks to the W-disjoint property [107]. We convert the $0/1$ masks into

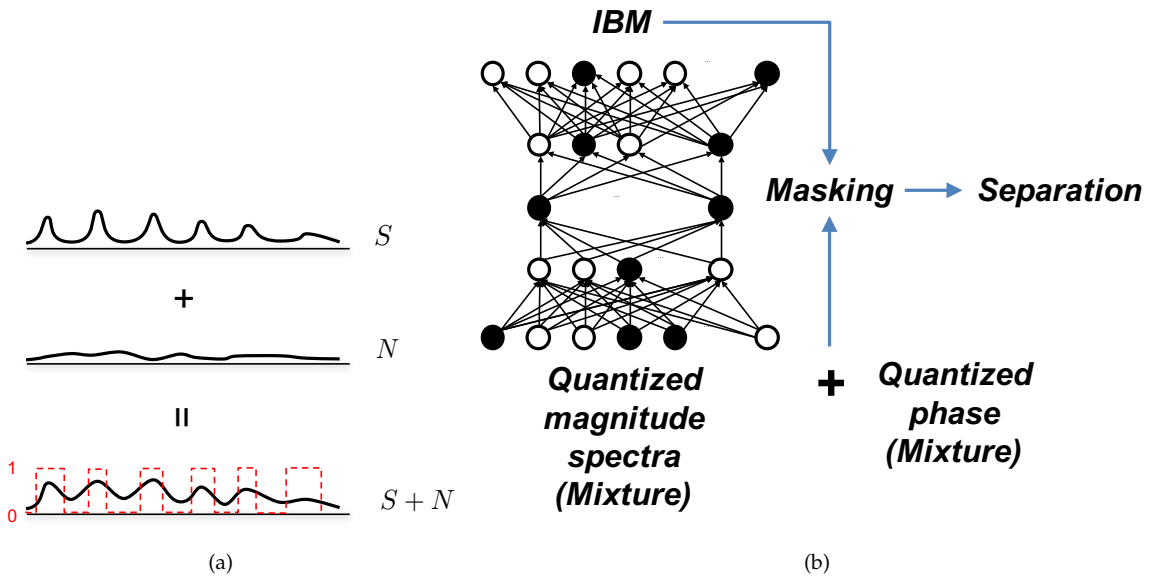


Figure 6.3: (a) Speech (S) and noise (N) mixing scenario with IBM (red dash) (b) the separation procedure with BNN as an IBM predictor.

bipolar binaries by turning zeros into -1's for our BNN.

Figure 6.3 (b) shows the separation scenario. For a given mixture spectrum, we first take off its phase and set it aside for the later use. As for the magnitudes, the Lloyd-Max's algorithm quantizes them using 4 bits. Each of the 4×(number of frequency bins) bits is fed to a input unit. The network is trained to produce a corresponding IBM vector. During the test time, we mask the complex-valued mixture spectrum by using the predicted IBM vector of the same size. Note that the complex-valued mixture spectrum is recovered from the quantized magnitudes and phase, each of which uses 4 bit to quantize a value. Finally, we inverse-transform this masked spectrum to recover the time domain signal.

1024 Hann-windowed samples are transformed with a 50% overlap. We train the network with mixtures of 5 randomly chosen utterances per one of 10 randomly chosen female speakers from the TIMIT corpus (50 clean utterances) and 10 noise signals used in [29], which amount to 500 noisy utterances. After STFT we have 99,770 training samples. As for testing, we collect another 250 noisy utterances from 5 randomly chosen female speakers.

We try two different network topologies, one with 2 hidden layers and 1024 hidden units per layer, and the other with 2048 units. In the first round of the training, we learn ordinary neural networks with the weight compression technique, on the quantized and dispersed bit pattern inputs. This first-round network serves not only as a baseline floating-point network, but as an initialization scheme. The DNN columns in Table 6.3 are the results from this first-round learning. Note that this network is handicapped in the sense

Table 6.3: A comparison of DNN and BNN on the speech enhancement performance. Numbers are in decibel (dB).

	2×1024		**2×2048**	
	DNN	BNN	DNN	BNN
SDR	8.60	8.00	9.08	8.55
SIR	28.48	24.91	28.41	26.24
SAR	8.87	8.43	9.35	8.88

that it did not take the original floating-point input magnitudes as the input. The sparsity of the BNN cases was set to be $\rho = 0.9$ for the 2×1024 case and $\rho = 0.95$ for the 2×2048 case, respectively.

In the second round, we train BNNs by initializing their parameters with the ones we trained in the first round. The BNN columns in Table 6.3 are the results of this. We evaluate the performance of the enhancement by using the standard BSS_TOOLBOX [48]. From the results, we can first see that the performance of BNN is slightly worse than DNN if they are with the same network structure. However, if we compare the BNN results with a larger structure (8.55 dB SDR from 2×2048) quickly catches up the performance of DNN (8.60 dB SDR from 2×1024). This is a promising results given the amount of cost reduction.

6.5 Summary

In this work we propose a bitwise version of artificial neural networks, where all the inputs, weights, biases, hidden units, and outputs can be represented with single bits and operated on using simple bitwise logic. Such a network is very computationally efficient and can be valuable for resource-constrained situations, particularly in cases where floating-point / fixed-point variables and operations are prohibitively expensive. In the future we plan to investigate a bitwise version of convolutive neural networks, where efficient computing is more desirable.

Chapter 7

Conclusions

In this dissertation, we presented some efficient machine learning algorithms that are designed to process as much data as needed while spending the least possible amount of resources, such as time, energy, and memory.

First, we discussed some manifold preserving topic models that were developed to perform better than the ordinary topic models in audio signal separation tasks. To this end, some hierarchical topic models were proposed where we assume an intermediate latent variable to capture the sparsely active examples from a relatively large dictionary of audio spectra. Similarly, the mixture of local dictionaries can be seen as a matrix factorization version of those topic modeling techniques, along with its special treatment of manifold preservation by forming a few locally clustered small dictionaries, which are then to be activated with the group-sparsity constraint. Although the manifold preserving models can provide a better performance, due to their nature of favoring a larger amount of training samples, and making use of the entire training samples during the test-time separation procedure, they often demand a lot of computational resources. To overcome this drawback, we also proposed to replace the bottleneck of the manifold preservation procedure, e.g. nearest neighbor searches at every iteration in the topic modeling case, with some integer (and consequently bitwise) operations. We employed some hashing techniques, for example WTA hashing, to replace the heavy KNN search on the floating-point Fourier magnitudes, to build a candidate neighborhood set, which eventually narrows down the search space greatly without sacrificing the separation performance.

We also studied that some special kinds of hash functions, such as LSH, can produce bit patterns as a noise robust binary feature extraction procedure, and showed its performance in the keyword spotting problem. Since it operates in a bitwise fashion, the speed and efficiency of the matching process are higher than a corresponding comprehensive model with floating-point operations, such as HMM. On top of that, a matrix factorization and its 2D deconvolution version were also proposed to perform source separation and detection at the same time on a sparse irregular representation of audio signals, such as the landmark representations of spectrograms. Since the number of those landmarks is usually a lot smaller than the

entire number of pixels in the spectrogram, we can expect that the procedure is more efficient than the corresponding full matrix factorization or deconvolution models.

CAE is a new big ad-hoc sensor array problem we defined in this dissertation, as a big audio data processing job. Due to the crowdsourced nature of the dataset, there are many interesting challenges in the tasks on this kind of datasets, among which we focused on the separation of the dominant source of the scene. To that end, we proposed PLCS, the component sharing algorithm that captures the common sources across all the simultaneous topic models (or matrix factorization tasks). This comprehensive model also extends to the case where we observe too many corrupted user-recorded signals, for example up to 1000 recordings, by utilizing the idea that only some of them are worth focusing on. We blend the nearest search procedure in the EM updates of this latent component sharing as well, by comparing the current source estimation with the input recordings, so that the heavy PLCS EM updates are applied to only those best recordings.

Finally, to describe an extremely optimized deep learning deployment system,BNN was also proposed. In BNNs, all the signals, parameters, and operations on them are defined in a bitwise fashion. Since its feedforward operations are now replaced with a bitwise correspondence, such as an XNOR gate instead of a multiplier, we believe that this can open a new possibility of deploying affordable deep learning engines for embedded systems, which often have to operate with a limited amount of resources. It is also noticeable that a straightforward extension of SGD and backpropagation works well to train this drastically quantized neural network by using the quantization noise injection and weight compression techniques.

References

[1] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. Plumbley, "Detection and classification of acoustic scenes and events: An ieee aasp challenge," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct 2013.

[2] J. Nam, G. J. Mysore, and P. Smaragdis, "Sound recognition in mixtures," in *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, 2012.

[3] A. L. Maas, Q. V. Le, T. M. O'Neil, O. Vinyals, P. Nguyen, and A. Y. Ng, "Recurrent neural networks for noise reduction in robust ASR," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, 2012, pp. 22–25.

[4] C. Weng, D. Yu, S. Watanabe, and B.-H. Juang, "Recurrent deep neural networks for robust speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2014, pp. 5532–5536.

[5] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, 2003, pp. 177–180.

[6] J.-L. Durrieu, G. Richard, B. David, and C. Févotte, "Source/filter model for unsupervised main melody extraction from polyphonic audio signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 564–575, 2010.

[7] A. Gkiokas, V. Katsouros, G. Carayannis, and T. Stajylakis, "Music tempo estimation and beat tracking by applying source separation and metrical relations," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2012, pp. 421–424.

[8] E. Tsunoo, T. Akase, N. Ono, and S. Sagayama, "Music mood classification by rhythm and bass-line unit pattern analysis," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Dallas, Texas, USA, 2010.

[9] E. A. Habets, "Single- and multi-microphone speech dereverberation using spectral enhancement," Ph.D. dissertation, Technische Universiteit Eindhoven, 2007.

[10] C.-M. Liu, W.-C. Lee, H.-W. Hsu et al., "High frequency reconstruction for band-limited audio signals," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2003.

[11] P. Smaragdis and B. Raj, "Example-driven bandwidth expansion," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct 2007, pp. 135–138.

[12] D. L. Sun and R. Mazumder, "Non-negative matrix completion for bandwidth extension: A convex optimization approach," in *Proceedings of the IEEE Workshop on Machine Learning for Signal Processing (MLSP)*, 2013.

[13] A. Adler, V. Emiya, M. Jafari, M. Elad, R. Gribonval, and M. Plumbley, "Audio inpainting," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 3, pp. 922–932, March 2012.

[14] A. Wang, "An industrial strength audio search algorithm." in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2003, pp. 7–13.

[15] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, "Query by humming: musical information retrieval in an audio database," in *Proceedings of the third ACM international conference on Multimedia*. ACM, 1995, pp. 231–236.

[16] E. Parzen, "On estimation of a probability density function and mode," *Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[17] V. Raykar, I. Kozintsev, and R. Lienhart, "Position calibration of microphones and loudspeakers in distributed computing platforms," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 1, pp. 70–83, Jan 2005.

[18] I. Himawan, I. McCowan, and S. Sridharan, "Clustered blind beamforming from ad-hoc microphone arrays. audio," *ieeeaslp*, vol. 19, no. 4, pp. 661–676, 2011.

[19] R. Zelinski, "A microphone array with adaptive post-filtering for noise reduction in reverberant rooms," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1988, pp. 2578–2581.

[20] J.-M. Valin, F. Michaud, J. Rouat, and D. Létourneau, "Robust sound source localization using a microphone array on a mobile robot," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2, 2003, pp. 1228–1233.

[21] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

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

[23] T. O. Virtanen, "Monaural sound source separation by non-negative matrix factorization with temporal continuity and sparseness criteria," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, 2007.

[24] N. Mohammadiha, J. Taghia, and A. Leijon, "Single channel speech enhancement using Bayesian NMF with recursive temporal updates of prior distributions," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2012.

[25] 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, 2009.

[26] M. Kim, J. Yoo, K. Kang, and S. Choi, "Nonnegative matrix partial co-factorization for spectral and temporal drum source separation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1192–1204, 2011.

[27] P. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Singing-voice separation from monaural recordings using deep recurrent neural networks," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2014.

[28] N. Mohammadiha, "Speech enhancement using nonnegative matrix factorization and hidden markov models," Ph.D. dissertation, KTH Royal Institute of Technology, 2013.

[29] Z. Duan, G. J. Mysore, and P. Smaragdis, "Online PLCA for real-time semi-supervised source separation," in *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, 2012, pp. 34–41.

[30] T. Hofmann, "Probablistic latent semantic indexing," in *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 1999.

[31] T. Hofmann, "Probablistic latent semantic analysis," in *Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI)*, 1999.

[32] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[33] A. Popescul, L. H. Ungar, D. M. Pennock, and S. Lawrence, "Probabilistic models for unified collaborative and content-based recommendation in sparse-data environment," in *Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2001.

[34] L. Cao and L. Fei-Fei, "Spatially coherent latent topic model for concurrent object segmentation and classification," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2007.

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

[36] P. Smaragdis, M. Shashanka, and B. Raj, "A sparse non-parametric approach for single channel separation of known sounds," in *Advances in Neural Information Processing Systems (NIPS)*, Vancouver, BC, Canada, 2009.

[37] J. Yagnik, D. Strelow, D. A. Ross, and R. Lin, "The power of comparative reasoning," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011, pp. 2431–2438.

[38] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik, "Fast, accurate detection of 100,000 object classes on a single machine," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[39] P. Indyk and R. Motwani, "Approximate nearest neighbor – towards removing the curse of dimensionality," in *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, 1998, pp. 604–613.

[40] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, NY, USA, 2002.

[41] M. Kim and P. Smaragdis, "Mixtures of local dictionaries for unsupervised speech enhancement," *IEEE Signal Processing Letters*, vol. 22, no. 3, pp. 293–297, March 2015.

[42] D. L. Donoho and V. Stodden, "When does nonnegative matrix facotrization give a correct decomposition into parts?" in *Advances in Neural Information Processing Systems (NIPS)*, vol. 16. MIT Press, 2004.

[43] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[44] M. Kim and P. Smaragdis, "Manifold preserving hierarchical topic models for quantization and approximation," in *Proceedings of the International Conference on Machine Learning (ICML)*, Atlanta, Georgia, 2013.

[45] D. L. Sun and G. J. Mysore, "Universal speech models for speaker independent single channel source separation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, 2013.

[46] A. Hurmalainen, R. Saeidi, and T. Virtanen, "Group sparsity for speaker identity discrimination in factorisation-based speech recognition," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, 2012.

[47] A. Lefèvre, F. Bach, and C. Févotte, "Itakura-Saito non-negative matrix factorization with group sparsity," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011.

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

[49] S. T. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[50] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[51] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, N. L. Dahlgren, and V. Zue, "TIMIT acoustic-phonetic continuous speech corpus," *Linguistic Data Consortium, Philadelphia*, 1993.

[52] M. Kim, P. Smaragdis, and G. J. Mysore, "Efficient manifold preserving audio source separation using locality sensitive hashing," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2015.

[53] P. Smaragdis, B. Raj, and M. Shashanka, "Supervised and semi-supervised separation of sounds from single-channel mixtures," in *Proceedings of the International Conference on Independent Component Analysis and Blind Signal Separation (ICA)*, London, UK, 2007.

[54] J. Rohlicek, W. Russell, S. Roukos, and H. Gish, "Continuous hidden Markov modeling for speaker-independent word spotting," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1989.

[55] R. Rose and D. Paul, "A hidden Markov model based keyword recognition system," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1990.

[56] M. Weintraub, "Keyword-spotting using sri's decipher large-vocabulary speech-recognition system," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1993.

[57] D. James and S. Young, "A fast lattice-based approach to vocabulary independent wordspotting," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1994.

[58] T. Ezzat and T. Poggio, "Discriminative word spotting using ordered spectro-temporal patch features," in *ISCA Tutorial and Research Workshops on Statistical And Perceptual Audition (SAPA)*, Brisbane, Australia, 2008.

[59] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2014, pp. 4087–4091.

[60] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[61] P. Smaragdis and M. Kim, "Non-negative matrix factorization for irregularly-spaced transforms," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, "Oct." 2013.

[62] F. Auger and P. Flandrin, "Improving the readability of time-frequency and time-scale representations by the reassignment method," *IEEE Transactions on Signal Processing*, vol. 43, no. 5, pp. 1068–1089, May 1995.

[63] D. C. Brabham, "Crowdsourcing as a model for problem solving: An introduction and cases," *Convergence: The International Journal of Research into New Media Technologies*, vol. 14, no. 1, pp. 75–90, 2008.

[64] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, "Learning from crowds," *Journal of Machine Learning Research*, vol. 11, pp. 1297–1322, Aug. 2010.

[65] N. J. Bryan, P. Smaragdis, and G. J. Mysore, "Clustering and synchronizing multicamera video via landmark cross-correlation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Kyoto, Japan, 2012.

[66] P. Shrestha, M. Barbieri, H. Weda, and D. Sekulovski, "Synchronization of multiple camera videos using audio-visual features," *IEEE Transactions on Multimedia*, vol. 12, no. 1, pp. 79–92, 2010.

[67] J. Yoo, M. Kim, K. Kang, and S. Choi, "Nonnegative matrix partial co-factorization for drum source separation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Dallas, Texas, USA, 2010.

[68] M. Kim, J. Yoo, K. Kang, and S. Choi, "Blind rhythmic source separation: Nonnegativity and repeatability," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Dallas, Texas, USA, 2010.

[69] P. Leveau, S. Maller, J. Burred, and X. Jaureguiberry, "Convolutive common audio signal extraction," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, 2011, pp. 165–168.

[70] M. Kim and P. Smaragdis, "Collaborative audio enhancement using probabilistic latent component sharing," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, 2013.

[71] M. Kim and P. Smaragdis, "Efficient neighborhood-based topic modeling for collaborative audio enhancement on massive crowdsourced recordings," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[72] C. Cotton and D. P. Ellis, "Audio fingerprinting to identify multiple videos of an event," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, March 2010.

[73] A. Asaei, N. Mohammadiha, M. J. Taghizadeh, S. Doclo, and H. Bourlard, "On application of non-negative matrix factorization for ad hoc microphone array calibration from incomplete noisy distances," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2015.

[74] M. Shashanka, "Latent variable framework for modeling and separating single channel acoustic sources," Ph.D. dissertation, Boston University, Aug. 2007.

[75] C. Ding, T. Li, and W. Peng, "On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing," *Computational Statistics and Data Analysis*, vol. 52, pp. 3913–3927, 2008.

[76] Z. Duan, G. J. Mysore, and P. Smaragdis, "Speech enhancement by online non-negative spectrogram decomposition in non-stationary noise environments," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Portland, OR, 2012.

[77] M. Kim and P. Smaragdis, "Bitwise neural networks," in *International Conference on Machine Learning (ICML) Workshop on Resource-Efficient Machine Learning*, Jul 2015.

[78] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[79] G. Cybenko, "Approximations by superpositions of sigmoidal functions," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[80] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[81] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[82] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2013.

[83] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. M, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[84] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "An experimental study on speech enhancement based on deep neural networks," *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 65–68, 2014.

[85] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, Jan. 2007.

[86] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969 – 978, 2009.

[87] E. Fiesler, A. Choudry, and H. J. Caulfield, "Weight discretization paradigm for optical neural networks," in *The Hague'90, 12-16 April*. International Society for Optics and Photonics, 1990, pp. 164–173.

[88] K. Hwang and W. Sung, "Fixed-point feedforward deep neural network design using weights +1, 0, and -1," in *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, Oct 2014.

[89] M. Courbariaux, Y. Bengio, and J.-P. David, "Low precision arithmetic for deep learning," *arXiv preprint arXiv:1412.7024*, 2014.

[90] L. Pitt and L. G. Valiant, "Computational limitations on learning from examples," *Journal of the Association for Computing Machinery*, vol. 35, pp. 965–984, 1988.

[91] M. Golea, M. Marchand, and T. R. Hancock, "On learning μ-perceptron networks with binary weights." in *Advances in Neural Information Processing Systems (NIPS)*, 1992, pp. 591–598.

[92] D. Soudry, I. Hubara, and R. Meir, "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights," in *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[93] W. S. McCulloch and W. H. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[94] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[95] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, Mar 1982.

[96] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 1753–1760.

[97] P. Clarkson and P. J. Moreno, "On the use of support vector machines for phonetic classification," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1999, pp. 585–588.

[98] H. Lee, P. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 1096–1104.

[99] D. Liu, P. Smaragdis, and M. Kim, "Experiments on deep learning for speech denoising," in *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech)*, Sep 2014.

[100] P. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep learning for monaural speech separation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2014.

[101] P. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Joint optimization of masks and deep recurrent neural networks for monaural source separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 12, pp. 2136–2147, Dec 2015.

[102] Y. Wang and D. L. Wang, "Towards scaling up classification-based speech separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 7, pp. 1381–1390, July 2013.

[103] A. Narayanan and D. L. Wang, "Ideal ratio mask estimation using deep neural networks for robust speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2013, pp. 7092–7096.

[104] D. S. Williamson, Y. Wang, and D. L. Wang, "A two-stage approach for improving the perceptual quality of separated speech," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2014, pp. 7084–7088.

[105] D. S. Williamson, Y. Wang, and D. L. Wang, "Reconstruction techniques for improving the perceptual quality of binary masked speech," *Journal of the Acoustical Society of America*, vol. 136, pp. 892–902, 2014.

[106] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2015.

[107] Ö. Yılmaz and S. Rickard, "Blind separation of speech mixtures via time-frequency masking," *IEEE Transactions on Signal Processing*, vol. 52, no. 7, pp. 1830–1847, 2004.