

© 2012 Kang Kang

ONLINE PARAMETER SELECTION FOR SOURCE SEPARATION  
USING NON-NEGATIVE MATRIX FACTORIZATION

BY

KANG KANG

THESIS

Submitted in partial fulfillment of the requirements  
for the degree of Master of Science in Electrical and Computer Engineering  
in the Graduate College of the  
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Adviser:

Assistant Professor Paris Smaragdis

# ABSTRACT

Blind source separation has been an area of study recently due to the many applications that might benefit from a good blind source separation algorithm. One instance is using blind source separation for audio denoising in cellular phones. In almost all instances, we have very little, if any, information about how background noise is mixed with the speaker's voice in a given cell phone conversation. Current techniques include spectral subtraction and Wiener filtering which are classical DSP techniques to deal with stationary noises. In this document, we aim to present a study on how to use blind source separation algorithms to denoise audio mixtures containing speech and various background noises. We mainly focus on how to implement an online source separation algorithm which can handle non-stationary noises. To address the implementation, we also present a study on how to select the parameters in the separation algorithm in order to deliver the best performance for denoising using a statistical metric we have defined.

*To my parents, for their love and support.*

# ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Paris Smaragdis, for his help and guidance in the completion of my thesis. I would also like to thank members of my lab, Johannes Traa and Minje Kim for their advice and input to my research and coursework. Finally, I give thanks to my parents for supporting me both financially and mentally throughout my graduate career.

# TABLE OF CONTENTS

|                                                                         |      |
|-------------------------------------------------------------------------|------|
| LIST OF TABLES . . . . .                                                | vii  |
| LIST OF FIGURES . . . . .                                               | viii |
| LIST OF ABBREVIATIONS . . . . .                                         | ix   |
| CHAPTER 1 INTRODUCTION . . . . .                                        | 1    |
| CHAPTER 2 BACKGROUND . . . . .                                          | 3    |
| 2.1 Introduction . . . . .                                              | 3    |
| 2.2 Spectral Subtraction . . . . .                                      | 3    |
| 2.3 Wiener Filtering . . . . .                                          | 6    |
| 2.4 LMS (Stochastic Gradient Descent) . . . . .                         | 7    |
| 2.5 Spectrograms . . . . .                                              | 8    |
| 2.6 Principal Component Analysis . . . . .                              | 10   |
| 2.7 PEASS . . . . .                                                     | 11   |
| 2.8 Spectral Features . . . . .                                         | 12   |
| 2.9 Conclusion . . . . .                                                | 13   |
| CHAPTER 3 LITERATURE REVIEW . . . . .                                   | 14   |
| 3.1 Introduction . . . . .                                              | 14   |
| 3.2 Probabilistic Latent Component Analysis . . . . .                   | 14   |
| 3.3 Gaussian Mixture Models . . . . .                                   | 15   |
| 3.4 Expectation Maximization Algorithm . . . . .                        | 17   |
| 3.5 Non-Negative Matrix Factorization . . . . .                         | 21   |
| 3.6 Using NMF for Source Separation . . . . .                           | 25   |
| 3.7 Conclusion . . . . .                                                | 28   |
| CHAPTER 4 PARAMETER SELECTION FOR ONLINE SOURCE<br>SEPARATION . . . . . | 29   |
| 4.1 Problem Formulation . . . . .                                       | 30   |
| 4.2 Noise Parameter Selection . . . . .                                 | 32   |
| 4.3 Results for Noise Parameter Selection . . . . .                     | 33   |
| 4.4 Analysis of Extracted Noise . . . . .                               | 34   |
| 4.5 Conclusion . . . . .                                                | 38   |

|            |                                                               |    |
|------------|---------------------------------------------------------------|----|
| CHAPTER 5  | REAL-TIME SOURCE SEPARATION WITH PA-                          |    |
|            | RAMETER SELECTION . . . . .                                   | 40 |
| 5.1        | Introduction . . . . .                                        | 40 |
| 5.2        | Online Source Separation using NMF Decomposition . . . . .    | 41 |
| 5.3        | Algorithm Performance . . . . .                               | 42 |
| 5.4        | Discussion of Online Parameter Selection Techniques . . . . . | 43 |
| 5.5        | Conclusion . . . . .                                          | 46 |
| CHAPTER 6  | CONCLUSION . . . . .                                          | 48 |
| REFERENCES | . . . . .                                                     | 49 |

# LIST OF TABLES

|     |                                                                |    |
|-----|----------------------------------------------------------------|----|
| 2.1 | Spectral Subtraction . . . . .                                 | 4  |
| 2.2 | Stochastic Gradient Descent and LMS . . . . .                  | 8  |
| 3.1 | Supervised Source Separation . . . . .                         | 26 |
| 3.2 | Semi-supervised Source Separation . . . . .                    | 27 |
| 4.1 | GMM Training Procedure for Noise Parameter Selection . . . . . | 32 |
| 4.2 | Comparison between MFCC GMMs and PEASS OPS . . . . .           | 34 |
| 4.3 | Comparison between MFCC GMMs and PEASS OPS . . . . .           | 38 |
| 5.1 | Online NMF Source Separation . . . . .                         | 42 |
| 5.2 | Online NMF Performance . . . . .                               | 43 |

# LIST OF FIGURES

|     |                                                                  |    |
|-----|------------------------------------------------------------------|----|
| 2.1 | Modern de-noising system . . . . .                               | 5  |
| 2.2 | General Wiener filter for de-noising and signal estimation . . . | 8  |
| 2.3 | Types of windows . . . . .                                       | 10 |
| 2.4 | Filter bank of triangular windows on mel frequency scale . . .   | 12 |
| 2.5 | Warped filter bank on linear frequency axis (Hz) . . . . .       | 13 |
| 3.1 | Output of two processes . . . . .                                | 16 |
| 3.2 | Audio mixture spectrogram . . . . .                              | 23 |
| 3.3 | Factorized components of an audio mixture spectrogram . . .      | 24 |
| 3.4 | Using too few components to decompose mixture . . . . .          | 24 |
| 3.5 | Using too many components to decompose mixture . . . . .         | 25 |
| 4.1 | 30 speech and 8 noise components . . . . .                       | 30 |
| 4.2 | 30 speech and 1 noise component . . . . .                        | 31 |
| 4.3 | Using PCA spectrogram data: 80 PCA dimensions, 10 Gaussians      | 35 |
| 4.4 | Using MFCC spectrogram data with 7 Gaussians . . . . .           | 35 |
| 4.5 | Using MFCC spectrogram data with 3 Gaussians . . . . .           | 36 |
| 4.6 | Overall perceptual score . . . . .                               | 36 |
| 4.7 | Using GMM with PCA-reduced features . . . . .                    | 38 |
| 4.8 | Using GMM with MFCC features . . . . .                           | 39 |
| 5.1 | Online algorithm for NMF source separation . . . . .             | 43 |
| 5.2 | Online parameter selection . . . . .                             | 44 |
| 5.3 | Mixture spectrogram . . . . .                                    | 45 |
| 5.4 | Noise component for real-time denoising . . . . .                | 46 |
| 5.5 | Noise component for real-time denoising . . . . .                | 47 |

# LIST OF ABBREVIATIONS

|      |                                         |
|------|-----------------------------------------|
| PLCA | Probabilistic Latent Component Analysis |
| NMF  | Non-negative Matrix Factorization       |
| PLSI | Probabilistic Latent Semantic Indexing  |
| KL   | Kullback-Leibler Divergence             |
| STFT | Short-Time Fourier Transform            |
| FFT  | Fast Fourier Transform                  |
| DFT  | Discrete Fourier Transform              |
| DTFT | Discrete-Time Fourier Transform         |
| CTFT | Continuous-Time Fourier Transform       |
| GMM  | Gaussian Mixture Model                  |
| PCA  | Principal Component Analysis            |
| ICA  | Independent Component Analysis          |
| LMS  | Least Mean Squares                      |
| SGD  | Stochastic Gradient Descent             |
| SVD  | Singular Value Decomposition            |
| ML   | Maximum Likelihood                      |
| LS   | Least Squares                           |
| IS   | Itakura-Saito                           |
| SNR  | Signal to Noise Ratio                   |
| SDR  | Signal Distortion Ratio                 |

|       |                                                           |
|-------|-----------------------------------------------------------|
| SIR   | Signal Interference Ratio                                 |
| PEASS | Perceptual Evaluation Methods for Audio Source Separation |
| OPS   | Overall Perceptual Score                                  |
| TPS   | Total Perceptual Score                                    |
| IPS   | Intelligible Perceptual Score                             |
| ITU-T | International Telecommunications Union - Technology       |

# CHAPTER 1

## INTRODUCTION

Audio signals have been the primary form of communication for the human race for quite some time. Though there have been inventions such as text messaging and electronic chat, speech remains the predominant form of communication between people. However, audio signals can be easily corrupted by background noise. Whether it is a loud bus passing by or a siren going off in the background, there are numerous sources of interference that might affect the way speech signals are received from one person to another.

There have been several approaches to deal with this problem using standard signal processing tools. A popular approach has been the Wiener filter. [1] discusses a multi-channel approach to eliminate background noise and enhance the speech by using a Wiener filter. Another popular approach is the use of spectral subtraction. Reference [2] provide a method which uses spectral subtraction to suppress noise in speech. Other methods such as Kalman-based approaches are addressed in [3] for audio mixtures, and [4] for single-channel audio mixtures. These methods have been shown to work well for noises which are stationary. However, many of the derivations and algorithms developed using classical DSP techniques have not been able to perform well when the noise is not stationary. This means that noises which have very fast time varying spectral characteristics are not very well estimated or suppressed by the algorithms. This is a problem since there are many noises in real life which are non-stationary. Algorithms based on spectral subtraction deal with this problem by attempting to learn the noise in the spectrogram domain before applying the separation. This approach has its limitations because it utilizes a voice activity detector which needs to be able to determine whether voice is present in a given mixture. The voice activity detector [5] is usually a model-based algorithm which looks at certain patterns in a spectrogram and determines how strongly they correlate to speech by a number of metrics which are determined in the model. In many

practical applications, it is very challenging to have a good voice activity detector.

Recently, data-driven approaches have used machine learning algorithms such as basis decompositions to separate sources straight from the spectral domain; see [6], [7], [8], [9]. These methods attempt to learn a dictionary basis for a given speaker which can represent roughly what the spectrum of the speaker looks like in a mixture. A factorization algorithm, either NMF or ICA, is employed to separate the noise and speech into two different spectrograms. Once the speech spectrogram is obtained, an inverse short-time Fourier transform can be applied to obtain a time series signal which has the background noise removed. The new methods do not require a voice activity detector and do not assume anything about the background noise. Yet, there are a few problems with the source separation algorithms. One issue with source separation is that it requires training data for both the noise and the speaker in a given mixture. Sometimes training data for one or both might be hard to obtain. Recent developments in [10] have been able to skirt this issue by training speech dictionary elements as the denoising occurs. This leads to the question of how many dictionary elements to train for either speech or noise to lead to the best possible performance. One method suggested might be to use SNR, SDR metrics and pick the number of components for speech or noise which give the best SNR, SDR performance. This is a rather costly metric and will not work if any of the original sound sources are missing. For very long signals in time, this method will also be very slow since we need to get the entire mixture separated before analyzing the two separated sources with their original mixtures. In a real-time setting, none of the above methods will work since we have no information about the speech or noise, and the background noise and speech might be changing in each frame of the spectrogram. In this document, we are concerned with removing noise from speech assuming that we have ample training data for speech. This might apply to cell phones and other communication devices which send audio signals from transmitter to receiver. We attempt to solve the problem of online source separation with a way to determine parameters to learn for the noise dictionary. In doing this, we present a new method of determining the quality of the source separation by using statistical models rather than SNR and SDR. We also present a method to incorporate the method of quality estimation into an online source separation algorithm.

# CHAPTER 2

## BACKGROUND

### 2.1 Introduction

In this chapter, we present some of the basic background material which is important to the rest of the thesis. We start by giving a brief introduction of spectral subtraction, which has been the method of denoising in modern day systems. We move on to talk about Wiener filtering, which is another technique to denoise signals, and discuss the LMS using stochastic gradient descent. We present the basics of the spectrogram which is used in the source separation techniques mentioned in this document, followed by a discussion of principal component analysis which is used as a dimensionality reduction in training the Gaussian mixture models. We end with a discussion of the perceptual evaluation toolkit which is currently the state-of-the-art technique to determine audio source separation quality.

### 2.2 Spectral Subtraction

The spectral subtraction technique has been studied widely and is currently the standard for removing noise from speech signals. The basic formulation for spectral subtraction is that we are given a mixture of two signals  $x[n] = s[n] + d[n]$ . We would like to estimate  $s[n]$  based on subtracting the power spectrum of the mixture by the estimated power spectrum of the noise.

Table 2.1: Spectral Subtraction

|        |                                                                                                                                                                                                                                                                                                                                                                                                |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Step 1 | Divide $x[n]$ into $1, \dots, N$ segments of equal length $L$                                                                                                                                                                                                                                                                                                                                  |
| Step 2 | for $i=1:N$<br>Obtain the fft of segment(i) and calculate $ X_i(\omega) ^2$<br>Obtain the phase of segment(i) and store in $\angle X_i(\omega)$<br>Obtain noise measurement $ D(\omega) ^2$<br>Calculate $E\{ S_i(\omega) ^2\}$ via equation (2) and also obtain $ S_i(\omega) $<br>Obtain $S_i(\omega) =  S_i(\omega)  \cdot \angle X_i(\omega)$<br>Obtain $s_i[n]$ via inverse fft operation |
| Step 3 | end for loop                                                                                                                                                                                                                                                                                                                                                                                   |
| Step 4 | obtain $s[n] = [s_1[n] s_2[n] \dots s_N[n]]$                                                                                                                                                                                                                                                                                                                                                   |

$$\begin{aligned}
x[n] &= s[n] + d[n] \\
X_d(\omega) &= S(\omega) + D(\omega) \\
|X(\omega)|^2 &= \dots \\
&|S(\omega)|^2 + |D(\omega)|^2 + S(\omega)D^*(\omega) + S^*(\omega)D(\omega)
\end{aligned}$$

One way to estimate the power spectrum of  $S(\omega)$  is to take the expected value of both sides.

$$\begin{aligned}
E\{|X(\omega)|^2\} &= \dots \\
&E\{|S(\omega)|^2 + |D(\omega)|^2 + S(\omega)D^*(\omega) + S^*(\omega)D(\omega)\} \\
E\{|X(\omega)|^2\} &= E\{|S(\omega)|^2\} + E\{|D(\omega)|^2\} \\
E\{|S(\omega)|^2\} &= \max\{E\{|X(\omega)|^2\} - E\{|D(\omega)|^2\}, 0\} \tag{2.1}
\end{aligned}$$

Equation (2.1) eliminates the issue of having negative values in the power spectrum. Though this might not be mathematically justified, in [11], the authors proposed this method as half-wave rectification. Spectral subtraction can be thought of as a direct operation on the spectrogram of the mixture signal. The algorithm for spectral subtraction is outlined in Table 2.1

The results for spectral subtraction work fairly well when we can accurately learn our noise measurements and the noise spectrum is relatively constant between neighboring frames in the spectrogram. Algorithms utilizing spectral subtraction have been widely used in modern cellphone de-noising systems because they are easy to implement and have fairly low computational com-

plexity. A schematic of a modern cellphone system is in Figure 2.1 [12]. We see that in the de-noising system, there is a signal activity detector (SAD). This detector is required to update the noise template. In a real-time system, a mixture comes in and the SAD attempts to determine if voice activity is present in the mixture. If there is no voice activity, the noise template or spectrogram will be updated and will be used in the succeeding mixture frames for removing noise from speech. One issue with this system would be that if noise changes fairly quickly from frame to frame, we will not be able to fully capture the characteristics of the noise. For example, if we hear a siren in the background as we are talking, the siren will produce a spectrogram that is varying in frequency over time. We might be able to learn parts of the siren as we progress through the noise template updates; however, when we start to speak, the siren frequencies could be elsewhere and we would have no hope of trying to remove the siren frequency at our current frame using only what we have learned in the past frame. One might say that we can average only a few past frames for the noise. Averaging will perform fairly poorly and it will start blurring the frequency contributions for different sounds, and in the end we get a giant block of frequencies to remove instead of a precise measure of the noise spectrum. Using just past frames of the learned noise spectrogram will also be detrimental due to the fact that the past frames might be describing noises that are no longer in the surroundings and will corrupt current estimates of the noise.

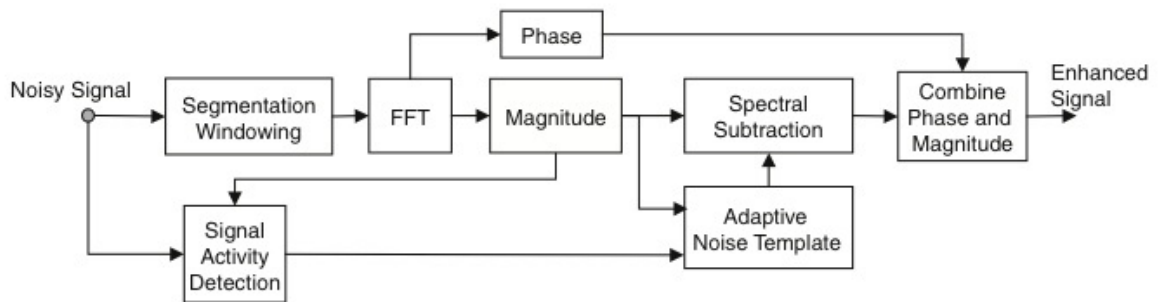


Figure 2.1: Modern de-noising system

## 2.3 Wiener Filtering

Wiener filters are widely used in signal processing applications for filtering and prediction. We will focus on the filtering aspect as it relates more closely to the application in the report. In Wiener filtering, a desired signal  $d[n]$  is passed through an known channel with impulse response  $g[n]$ . Gaussian white noise is also added at the output of the unknown channel. The signal at the receiving end is defined by  $x[n] = g[n] * s[n] + v[n]$  and is passed through the Wiener filter  $h[n]$ . The output from  $h[n]$  is then compared to  $d[n]$  and the resulting error  $e[n] = h[n] * x[n] - d[n]$  is used to update the filter  $h[n]$ , see Figure 2.2.

In the Wiener filter formulation, we would like to minimize the mean squared error defined by

$$\mathcal{E} = E\{\|d[n] - y[n]\|^2\} \quad (2.2)$$

We can find  $h[n]$  which minimizes the expression in (1) by

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial h[l]} &= E\{x[n-l]e[n]\} = 0 \\ \Rightarrow \sum_{m=-\infty}^{\infty} h[m]R_{xx}[l-m] &= R_{xd}[l] \end{aligned}$$

where

$$\begin{aligned} R_{xx}[\tau] &= E\{x[n]x[n+\tau]\} \\ R_{xd}[\tau] &= E\{x[n]d[n+\tau]\} \end{aligned}$$

The Wiener filter makes the assumption that we know the signal properties  $R_{xx}$  and  $R_{xd}$ . From this we obtain the frequency domain expression for the IIR wiener filter which minimizes the mean square error defined by (2.1).

$$H(e^{j\omega}) = \frac{S_{xd}(\omega)}{S_{xx}(\omega)}$$

Taking the above equation even further, we obtain

$$H(e^{j\omega}) = \frac{G^*(e^{j\omega})S_{dd}(\omega)}{\|G(e^{j\omega})\|^2 S_{dd}(\omega) + S_{vv}(\omega)}$$

What these equations imply is that we can use the Wiener filter to obtain a desired signal  $d[n]$  subject to a known channel  $g[n]$  and noise  $v[n]$  just from the received noisy signal defined by  $y[n]$  given that we know the channel and the signal's autocorrelation and cross correlation. However, these equations only provide insight into the Wiener filter problem and are not useful in practical applications when we do not know our desired signal power spectrum and channel impulse response. In many cases, all we are given is the received signal  $y[n]$  and we do not know our  $d[n]$  or  $g[n]$ . These cases require us to look into algorithms such as the LMS which is an application of stochastic gradient descent where we try and obtain the desired signal using only approximate  $R_{xx}$  and  $R_{xd}$ .

## 2.4 LMS (Stochastic Gradient Descent)

Stochastic gradient descent is a general optimization algorithm which attempts to iteratively minimize a cost function by descending in the direction of maximum change of the cost function. In this case, the cost function is defined by (2.1) and taking the negative of the gradient of the cost function we obtain the direction of maximum decrease.

$$-\nabla_h \mathcal{E}(h) = -(2R_{xx}h - 2p)$$

where the variables  $x$ ,  $h$ , and  $p$  are defined as

$$\begin{aligned} x &= [x[n]x[n-1] \dots x[n-(N-1)]]^T \\ h &= [h[n]h[n-1] \dots h[n-(N-1)]]^T \\ p &= [R_{xd}[n]R_{xd}[n-1] \dots R_{xd}[n-(N-1)]]^T \end{aligned}$$

The LMS algorithm is a special case of the stochastic gradient descent where we make approximations that  $R_{xx} \approx x[n]x[n]^T$  and  $R_{xd} \approx x[n]d[n]$ . It has been shown that this is a suitable approximation for convergence to an optimal  $h$  which minimizes (2.1) as long as the learning rate  $\eta$  satisfies (2.2).

Table 2.2 outlines the algorithm for both stochastic gradient descent and LMS. The difference between these two algorithms is in step 3, where (a) corresponds to the general stochastic gradient descent algorithm and (b)

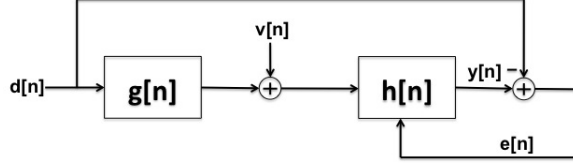


Figure 2.2: General Wiener filter for de-noising and signal estimation

Table 2.2: Stochastic Gradient Descent and LMS

|           |                                                                      |
|-----------|----------------------------------------------------------------------|
| Step 1    | Initialize $h^0$                                                     |
| Step 2    | Update $h^{(n+1)} = h^{(n)} - \eta \nabla_h \mathcal{E}(h)$          |
| Step 3(a) | Update via $h^{(n+1)} = h^{(n)} + \eta(2R_{xx}h - 2p)$               |
| Step 3(b) | Update via $h^{(n+1)} = h^{(n)} + 2\eta x[n](d[n] - x[n]^T h^{(i)})$ |

corresponds to LMS.

### 2.4.1 Application to Audio De-noising

From the previous formulation of the LMS algorithm, we can see that if we were to obtain a noisy measurement of speech, we can estimate the covariance matrix of the noise during the part where speech is not present and design a Wiener filter which will be able to remove the noise via a LMS algorithm. However, there are a few problems with this approach. The first is that noise is assumed to be stationary, and in the case of Gaussian, wide sense stationary. If the noise is not stationary or wide sense stationary, the Wiener filter will not be able to remove such noise.

## 2.5 Spectrograms

The spectrogram is a short-time Fourier transform of a discrete time signal  $x[n]$  which can provide details in the time-frequency domain of the signal's behavior. The short-time Fourier equation can be found in many DSP books today [13], [14]:

$$STFT\{x[n]\}(m, \omega) = X(m, \omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n} \quad (2.3)$$

where  $w[n]$  represents a window function that can be a rectangular window, Hamming window, or Bartlett window. The following are equations for the windows are provided below.

Rectangular Window  $w[n] = 1$  if  $0 \leq n \leq N$

Hamming Window:  $w[n] = 0.54 - 0.46\cos(2\pi\frac{n}{N})$  if  $0 \leq n \leq N$

Bartlett Window:  $w[n] = \begin{cases} \frac{2n}{N} & \text{if } 0 \leq n \leq \frac{N}{2} \\ 2 - \frac{2n}{N} & \text{if } \frac{N}{2} \leq n \leq N \end{cases}$

The rectangular window is just a truncation of the signal. It is easy to implement and can resynthesize the STFT given any hop size. It also does not require any additional storage during implementation. However, the rectangular window has sharp cut-offs and therefore introduces ringing effects in the resynthesis step. The Hamming window provides an alternative to the rectangular window by tapering off sharp edges which are present in the rectangular window. Therefore, it does not introduce artifacts in the resynthesis of the STFT; however, providing a perfect resynthesis is not trivial when using the Hamming window. It was proven in [15] that using a quarter or half window length overlap provided perfect resynthesis from the STFT back to the time domain. These are known today as the overlap-and-add methods (OLA). The Bartlett window is similar to the triangular window, but the  $n = 0$  and  $n = N$  points are always 0 for a length  $N + 1$  Bartlett window. Figure 2.3 shows the typical shape of these windows when  $N = 255$ .

### 2.5.1 Calculating the STFT

Once we determine the window to use for the STFT, we need to determine the window size and the hop size. In most cases, increasing the hop size leads to more frames in the STFT, but becomes very expensive in computation. A general rule of thumb is to use constant hop sizes of  $\frac{N}{4}$ ,  $\frac{N}{2}$ ,  $\frac{3N}{4}$ . To calculate the STFT, we simply calculate each frame via equation (2.3). This is best explained by taking the first  $N$  samples of the signal and multiplying it by the window  $w[n]$ . We take the FFT of this result and this becomes the first frame of the STFT. The next frame is calculated by sliding the window over by the hop size and multiplying it by the window  $w[n]$  and taking the FFT of the result. We do this until we have gone through the entire signal. When

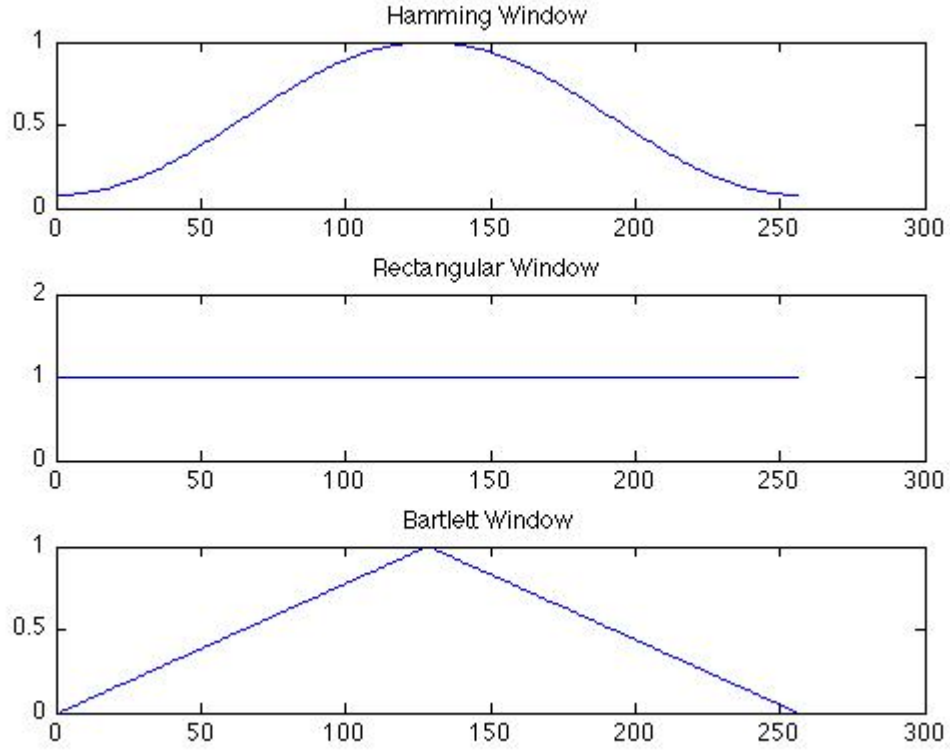


Figure 2.3: Types of windows

we get to the last frame, we zero-pad the signal  $x[n]$  until we can obtain a length  $N$  segment, and we apply the same method of taking the product with the window and taking the FFT.

## 2.6 Principal Component Analysis

Principal component analysis (PCA) is a mathematical tool used in machine learning as a means to project a set of data in a high-dimensional space to a low-dimensional space. The first ever use of PCA is discussed in [16]. The main idea behind PCA is to find a set of principal components which projects the original data onto a subspace which has its variance maximized. Consider a set of data  $X = \{x_1, x_2, \dots, x_N\}$ . Each  $x_i \in \mathbb{R}^M$ . We wish to project  $X$  onto a low-dimensional subspace. As an example, let this low-dimensional subspace have dimension 1. The projected data can be written as  $u_1^T x_n$ . The mean of the projected data is defined as  $u_1^T \bar{x}$  where  $\bar{x} = \frac{1}{N} \sum_{n=1}^N x_n$ . After

some math manipulations, we obtain the variance of the data as:

$$V = \frac{1}{N} \sum_{n=1}^N (x_n - \bar{x})(x_n - \bar{x})^T$$

This equation is the same in [17]. In order to maximize the projected variance defined by  $u_1^T V u_1$ , we have to add the constraint  $u_1^T u_1 = 1$  to enforce that  $\|u_1\|$  does not go to infinity. We eliminate this constraint and make the problem into a Lagrange multiplier problem by rewriting the cost function as:

$$J(u_1) = u_1^T V u_1 + \lambda_1 (1 - u_1^T u_1)$$

Taking the derivative of the cost function with respect to  $u_1$ , we obtain:

$$V u_1 = \lambda_1 u_1$$

This is the famous eigenvalue equation and we see that if we set the vector  $u_1$  to be the eigenvector corresponding to the largest eigenvalue, we have successfully maximized the variance of the projected data onto a 1-dimensional subspace. We make similar arguments for 2, 3, up to  $D < M$  dimensions.

## 2.7 PEASS

In the source separation literature, there are many algorithms developed to separate sources using online, offline, supervised, semi-supervised techniques. As more algorithms come out, there is an increasing demand for evaluation of how these algorithms perform. In [18], the authors provide a subjective and also objective assessment of the quality of the separated mixture to the original sources. They created a toolkit known as PEASS, which has been the standard to evaluate source separation since the paper's publication. The toolkit requires the original audio signal  $x[n]$ , the separated source estimate  $\hat{x}[n]$ , and the noise signal  $v[n]$  and calculates several metrics, both subjective and objective, to evaluate the performance of the source separation. Common approaches in evaluating source separation algorithms use signal-to-distortion ratio as the benchmark in obtaining an objective measure of the source separation. However, PEASS goes a step further and provides

subjective perceptual scores. The perceptual scores are calculated based on a model which was calibrated using 20 test listeners and 80 different sounds. The model is essentially a non-linear mapping of the SDR value to scores defined as OPS, TPS, and IPS. Each score is on a scale of 100 and subjectively based on the MUSHRA protocol [19]. This protocol is similar to the ITU standard which is used to evaluate noise suppression in audio mixtures [20]. The algorithm has been widely accepted and is used as a metric in some of the experiments performed in Chapters 3 and 4.

## 2.8 Spectral Features

To represent audio data in a way that is more amenable to analysis, features must be extracted from the raw audio waveforms. The most basic feature representation is the linear-frequency, log-magnitude short-time Fourier transform (spectrogram). The analysis window used was 1,024 samples long with a hop size of 512 samples. Each window produces a frame in the spectrogram that can be treated as a data point in a 513-dimensional feature space. Each dimension in this space represents the energy in a small band of frequencies in the signal. Low-pitched sounds will have most of the energy at low dimensions whereas high-pitched sounds will have more energy in the higher dimensions. We call these *lin\_spectra* features.

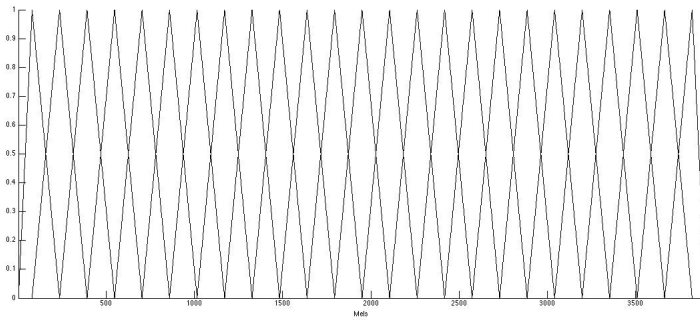


Figure 2.4: Filter bank of triangular windows on mel frequency scale

To represent the raw spectrogram features in a more perceptually meaningful way, each frame is passed through a mel filter bank (Figure 2.4 and 2.5). This has the effect of emphasizing the lower frequencies, reflecting the human ears non-linear behavior. In particular, 25 triangular windows are

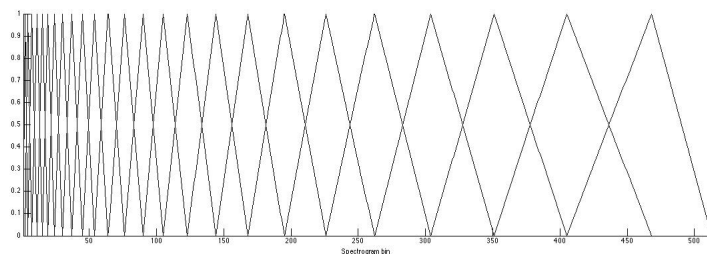


Figure 2.5: Warped filter bank on linear frequency axis (Hz)

evenly spaced out along the mel scale and then warped to the Hz scale. [21] discuss mel scale feature extraction, an effective method for defining audio features more closely aligned with how humans perceive sound.

## 2.9 Conclusion

In this chapter, we presented the necessary background information to understand many of the methods which are used in this document. More information can be found by going into the literature cited in each of the previous sections. In the next section, we will provide a literature review in the area of source separation and give some mathematical formulation on some of the methods used in this thesis.

# CHAPTER 3

## LITERATURE REVIEW

### 3.1 Introduction

In this chapter we give a literature review of the past work in the area of online source separation and non-negative matrix factorization. We also discuss the Gaussian mixture models which are used in this document. We provide the necessary derivations in order to obtain the update equations to learn a mixture model. We start with probabilistic latent component analysis (PLCA) which has been used for online source separation to separate speaker and background noise. PLCA uses a statistical model with EM update equations to estimate probabilistic distributions which describe the speaker's dictionary basis. This is very similar to probabilistic latent semantic indexing (PLSI) which uses the same process to cluster together documents with similar words and word meanings. Hofmann, [22] discusses the derivation details of PLSI. We will finish with a discussion of non-negative matrix factorization and how it has been used in the past for both supervised and semi-supervised source separation.

### 3.2 Probabilistic Latent Component Analysis

Probabilistic latent component analysis, or PLCA, is a method of source separation which is similar to NMF methods of source separation. PLCA uses a statistical model to explain the problem whereas NMF is simply an optimization problem. In PLCA, we use spectrograms which have each of its time frames normalized to sum to one. This makes the spectrogram with  $T$  time frames a collection of  $T$  probability distributions. The speaker and noise dictionaries come from a set of probability distributions which represent the

bases which make up the speaker and noise spectral characteristics. This can be denoted by  $P(f|z)$  where  $z \in \mathcal{S}_1 \cup \mathcal{S}_2$ . The activations, denoted by  $P_t(z)$ , are the coefficients for each of the dictionary components. The entire normalized magnitude mixture spectrogram can be represented by the following [10]:

$$P_t(f) \approx \sum_{z \in \mathcal{S}_1 \cup \mathcal{S}_2} P(f|z)P_t(z) \text{ for } t = 1, \dots, T$$

The distributions are non-negative since they represent magnitude spectrograms. Therefore, the estimation for the dictionary elements and activations belonging to  $\mathcal{S}_1$  and  $\mathcal{S}_2$  which represent the magnitude spectrogram can be formulated as a maximum likelihood problem which can be solved using the EM algorithm.

### 3.3 Gaussian Mixture Models

Gaussian mixture models, or GMMs, are generated on a set of  $k$ -dimensional data defined by  $X = [x_1, x_2, \dots, x_n]$ . Once generated, the Gaussian mixture model is a statistical model which contains a number of multi-variate Gaussian distributions defined by

$$f_i(x) = (2\pi)^{\frac{k}{2}} |\Sigma_i|^{-\frac{1}{2}} e^{-\frac{1}{2}(x - \mu_i)' \Sigma_i^{-1} (x - \mu_i)}, i \in (1, 2, \dots, m)$$

where  $\Sigma$  is a positive-definite covariance matrix and  $m$  is the number of multi-variate Gaussians in the GMM. The mixture model is often used to model clusters in a large set of data. The term Gaussian means that this mixture model is best suited to handle clusters which can be best described by Gaussians. GMMs are often used for classification of data sets. For example, consider that we monitor the output of two independent processes which have Gaussian-like output behavior in Figure 3.1. We also generated a gaussian mixture model from the data which contains two 1-dimensional Gaussians to describe the data. The ground truth for the two processes were Gaussians with  $\mu_1 = 0$ ,  $\mu_2 = 10$ , and unit variances. The Gaussian mixture model estimates of the means were  $\mu_1 = 0.0087$  and  $\mu_2 = 9.9973$  with variances  $\sigma_1 = 0.942$  and  $\sigma_2 = 0.8909$ . From the plot, we can see that the Gaussian

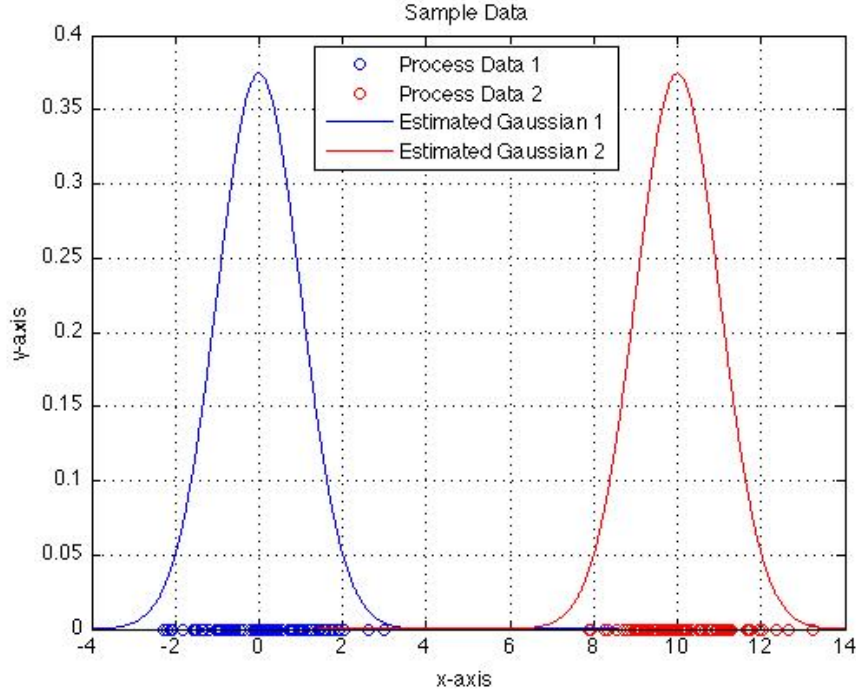


Figure 3.1: Output of two processes

mixture model correctly models the two processes with respect to the means and the variances. The priors for the two Gaussians are both  $\pi = 0.5$ . This prior indicates that there is equal weighting between the two Gaussians. The priors will change if we have more data for one process versus another. This example demonstrates how Gaussian mixture models are used to statistically model processes. However, we can also use the gaussian mixture model to make predictions about incoming data. The model can be used to define a likelihood function which provides us insight into how a set of data fits into a particular process.

$$p(x|\theta) = \sum_{k=1}^M \pi_k N(x|\mu_k, \Sigma_k)$$

Overall, the Gaussian mixture model is very useful in audio; however, its parameters need to be determined. This motivates our next section which discusses a popular method called Expectation Maximization (EM), which estimates the parameters of the mixture model when we are provided the data.

## 3.4 Expectation Maximization Algorithm

### 3.4.1 Bayesian Inference

We are presented with some data  $\{x_1, x_2, \dots, x_N\}$ . We wish to maximize the likelihood of the given data using a distribution with parameter  $\theta$ . We can formulate the problem into

$$\hat{\theta} = \arg \max_{\theta} p(x|\theta)$$

However, we make an assumption that the random variables in  $x$  are i.i.d., which means we can write the above equation as

$$\hat{\theta} = \arg \max_{\theta} \prod_n p(x_n|\theta)$$

So we are posed with an optimization problem. It is the normal convention to take the log of both sides and maximize the log likelihood function. This is valid because log is a monotonically increasing function. Our actual optimization problem is

$$\hat{\theta} = \arg \max_{\theta} \sum_n \log(p(x_n|\theta))$$

If we are given that the above function is a Gaussian and that the data came from this distribution, we can successfully estimate the parameters of that data by just taking the derivative of the likelihood function with respect to  $\mu$  and  $\sigma$  to obtain the parameters of the pdf. The problem described above will reduce to an inference problem with more data and more pdfs. However, if we are given just the data and we would like to get both labels and parameters from the data, we would need to use a mixture model.

### 3.4.2 Latent Variables

A latent variable is used in estimation of a mixture model. Consider a set of data points which came from some unknown random processes. We can model the data by assuming that they came from  $K$  random processes, with each draw determining independent probabilities of  $x_n$  data. So now our goal

is to estimate

$$p(x_n|\theta, \pi)$$

for each  $n$ , where  $\theta$  are the parameters of the distribution and  $\pi$  are the priors. In the case of a Gaussian distribution, they are

$$p(x_n|\mu, \sigma^2, \pi) = \sum_m p(x_n, z_n = m|\mu_m, \sigma_m^2, \pi_m)$$

where we can consider  $p(x_n, z_n = m|\mu_m, \sigma_m^2, \pi_m)$  as the probability that the  $n^{th}$  data point came from the  $m^{th}$  distribution, given the parameters and the latent variable priors.

### 3.4.3 EM Algorithm

The EM algorithm can be derived using the latent variables as well as the inference idea discussed previous. Consider the simple solution provided below.

- 1: Initial parameters  $\theta$  and  $\pi$
- 2: Compute probabilities of each hidden variable given current parameters
- 3: Compute new parameters for each model

Let us observe the following effect of introducing latent variables to our formulation. We consider Gaussians for now to make the effect obvious.

$$p(x_n|\mu, \sigma^2, \pi) = \sum_m p(x_n, z_n = m|\mu_m, \sigma_m^2, \pi_m)$$

Note that this is merely an application of the law of total probability. Also note,  $p(x_n, z_n = m|\mu_m, \sigma_m^2, \pi_m)$  can be rewritten as

$$p(x_n, z_n = m|\mu, \sigma^2, \pi) = p(x_n, z_n = m|\mu_m, \sigma_m^2, \pi_m) = p(x_n|\mu_m, \sigma_m^2)p(z_n = m|\pi_m)$$

So our goal becomes

$$\hat{\theta} = \arg \max_{\theta} \log\left(\sum_z p(\bar{x}, \bar{z}|\theta)\right) \quad (3.1)$$

Note:  $\bar{z} = \{z_1, z_2, \dots, z_K\}$  and  $\bar{x} = \{x_1, x_2, \dots, x_N\}$ . So there are  $K$  latent variables (distributions) and  $n$  observations. Our goal here is to maximize the log likelihood of the data and latent variables given the parameters of the distribution. Since we do not know the distribution parameters, we will employ the EM algorithm to obtain an iterative technique to solve for the parameters.

### 3.4.4 Massaging the Problem

Equation (3.2) provides us with a slight problem since we have to optimize over a log of sums. This is not a good procedure, so what has been suggested in the literature is that we move the log inside the summation. This is valid but not trivial in solving the optimization problem. A full proof can be found in [23] which revolves around the application of Jensen's inequality due to the fact that the log function is convex. What we end up with are the following E and M steps:

- 1: E-step:  $\mathbb{E}_{\bar{z}|\bar{x}, \theta^{(t)}} \log(p(\bar{x}, \bar{z}|\theta)) = \sum_{\bar{z}} \log(p(\bar{x}, \bar{z}|\theta)) p(\bar{z}|\bar{x}, \theta^{(t)})$
- 2: M-step:  $\theta^{(t+1)} = \arg \max_{\theta} \sum_{\bar{z}} \log(p(\bar{x}, \bar{z}|\theta)) p(\bar{z}|\bar{x}, \theta^{(t)})$

Going through these procedures, we can derive the update equations for the EM algorithm. We restrict ourselves to the gaussian mixture model during this derivation.

$$\mathbb{E}_{\bar{z}, \bar{x}, \theta^{(t)}} [\log p(\dots)] = \sum_{\bar{z}} [\log(p(\bar{x}, \bar{z}|\theta)) p(\bar{z}|\bar{x}, \theta^{(t)})] \quad (3.2)$$

$$= \sum_{z_1} \dots \sum_{z_N} [\sum_n \log(p(x_n|z_n = m, \theta)) p(z_n|\theta)] p(\bar{z}|\bar{x}, \theta^{(t)}) \quad (3.3)$$

$$= \sum_n \sum_{z_1} \dots \sum_{z_N} \log[(p(x_n|z_n = m, \theta)) p(z_n|\theta)] p(\bar{z}|\bar{x}, \theta^{(t)}) \quad (3.4)$$

$$= \sum_n \sum_m \log[(p(x_n|z_n = m, \theta)) p(z_n|\theta)] p(z_n = m|x_n, \theta^{(t)}) \quad (3.5)$$

From (3.2) to 3.3), we simply applied Baye's rule to rewrite the joint probability as a conditional probability with its respective prior. From (3.3) to (3.4), we shift the order of the summation operator. From (3.4) to (3.5), we deduce that each data sample is only generated by one latent distribution and we can sum over a particular  $z$  instead of over all the  $z$ 's. If there

are  $K$  latent variables, then  $m$  ranges from 1 to  $K$ . Let us define the term  $p(z_n = m|x_n, \theta^{(t)})$  by  $\alpha_{nm}$ . This term is the probability that the  $n^{th}$  data point came from component  $m$ . We require to solve for all of these in the E-step. This can be solved by a direct application of Baye's rules again.

$$\alpha_{nm} = \frac{p(x_n|z_n = m, \theta^{(t)})p(z_n = m|\theta^{(t)})}{\sum_k p(x_n|z_n = k, \theta^{(t)})p(z_n = k|\theta^{(k)})}$$

### 3.4.5 Gaussian Mixture Model

We will now derive explicit update equations for a Gaussian mixture model. Referring back to basic probability, the Gaussian pdf can be written as

$$p(x_n|\theta) = p(x_n|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_n - \mu)^2}{2\sigma^2}}$$

Equation (3.5) can then be rewritten in terms of the gaussian pdf as

$$\mathcal{L} = \sum_n \sum_m [\log \frac{1}{\sqrt{2\pi}} + \log \frac{1}{\sigma_m} - \frac{1}{2} \frac{(x_n - \mu_m)^2}{\sigma_m^2} + \log \pi_m] \alpha_{nm}$$

We differentiate the likelihood function with respect to each  $\mu_m$  and  $\sigma_m$ .

$$\frac{\partial \mathcal{L}}{\partial \mu_m} = \sum_n \frac{x_n - \mu_m}{\sigma_m^2} \alpha_{nm} = 0 \quad (3.6)$$

$$\implies \frac{1}{\sum_n \alpha_{nm}} \sum_n x_n \alpha_{nm} = \mu_m \quad (3.7)$$

$$\frac{\partial \mathcal{L}}{\partial \sigma_m} = \sum_n [-\frac{1}{\sigma_m} + \frac{(x_n - \mu_m)^2}{\sigma_m^3}] \alpha_{nm} = 0 \quad (3.8)$$

$$\implies \sigma_m^2 = \frac{1}{\sum_n \alpha_{nm}} \sum_n \alpha_{nm} (x_n - \mu_m)^2 \quad (3.9)$$

Estimation of the priors requires us to add an equality constraint to our initial optimization problem so we do not get infinity as the answer to our future priors. This constraint is

$$\sum_k \pi_k = 1$$

This is normal, as we do require that all the priors of the latent variables sum up to 1 to satisfy a given latent variable probability space and for Baye's and the law of total probability to hold. This equality constraint appears as a Lagrange multiplier when we do our optimization. Applying the basic principles of Lagrangian multipliers, we can arrive at an update solution for  $\pi_m$ .

### 3.5 Non-Negative Matrix Factorization

Non-negative matrix factorization decomposes any non-negative data matrix into a product of two non-negative matrices. In machine learning, these two matrices are known as dictionary and activation. In the machine learning literature, non-negative matrix factorization is formulated by

$$\begin{aligned} X &= WH \\ \text{s.t. } W &\geq 0, H \geq 0 \end{aligned} \tag{3.10}$$

where  $X \in \mathbb{R}^{m \times n}$ ,  $W \in \mathbb{R}^{m \times k}$ ,  $H \in \mathbb{R}^{k \times n}$ . In practical cases, we choose  $k$  to be smaller than  $m$  or  $n$  so we can achieve some type of compression with respect to the original data. We can now define the equality in (3.10) as the following optimization problem. Two metric functions  $D(\cdot)$  have been examined in current literature. The first is the standard Frobenius norm for matrices defined in (3.11). The second measure is similar to the KL-divergence except for a normalization factor which is appended at the end defined by (3.12).

$$\min_{W, H} = D(X, WH) \text{ s.t. } W, H \geq 0 \tag{3.11}$$

$$D(X, Y) = \|X - Y\|^2 = \sum_{ij} (X_{ij} - Y_{ij})^2 \tag{3.12}$$

$$\hat{D}(X, Y) = D(X||Y) = \sum_{ij} (X_{ij} \log \frac{X_{ij}}{Y_{ij}} - X_{ij} + Y_{ij}) \tag{3.13}$$

This problem has been studied by several researchers in the past. The most well-known approach to solving this optimization problem came out of a paper by Lee and Seung in 2001 [24]. Their proposed solution provides a

set of multiplicative update rules for decreasing the cost functions defined in (3.12) and (3.13). These rules have become very popular among researchers today because of their ease of implementation and memory requirements.

### 3.5.1 Divergence NMF Update Equations

$$W^{(n+1)} = W^{(n)} \circ \frac{\frac{X}{W^{(n)}H^{(n)}} H^{(n)T}}{\sum H^{(n)}}$$

$$H^{(n+1)} = H^{(n)} \circ \frac{W^{(n)T} \frac{X}{W^{(n)}H^{(n)}}}{\sum W^{(n)T}}$$

### 3.5.2 Frobenius NMF Update Equations

Shown below is the update equations which can be found in [?] which will decrease the cost function defined by (3.12). The sub-indice  $n$  is the iteration. For most audio spectrograms, it has been found that 30 to 40 iterations will provide a good result.

$$W^{(n+1)} = W^{(n)} \circ \frac{(XH^{(n)T})}{W^{(n)}H^{(n)}H^{(n)T}}$$

$$H^{(n+1)} = H^{(n)} \circ \frac{W^{(n)T}X}{W^{(n)T}W^{(n)}H^{(n)}}$$

The  $\circ$  denotes a Hadamard product. When working with audio spectrograms, it has been empirically shown that the divergence norm produces better results in coming up with a  $W$  and  $H$  to approximate the  $X$  matrix. In order to illustrate the results obtained using NMF, we have provided some examples of artificial and real world audio data decomposed using the NMF procedure.

### 3.5.3 Decompositions using NMF

Figure 3.2 is a magnitude spectrogram of a recording of two different beeps and a snare drum. We can see three distinct patterns on the spectrogram and the two different beeps are characterized by the white lines with distinct bright spots in the prominent frequencies of the beeps. The snare drum is

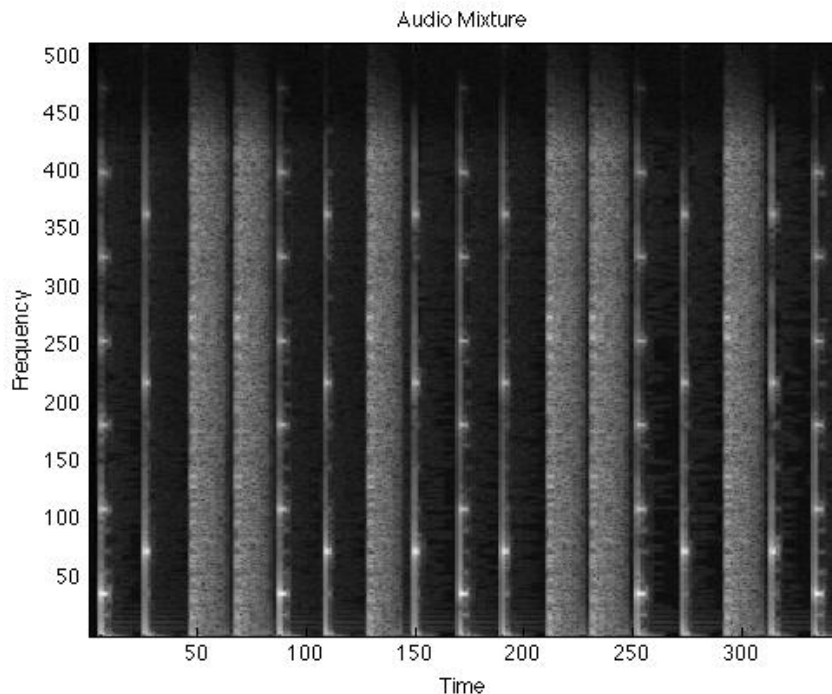


Figure 3.2: Audio mixture spectrogram

characterized by the smear-like pattern in the spectrogram. Since this is the magnitude spectrogram of the mixture, it is non-negative and we can apply NMF using the divergence update equations. We note that we chose  $k = 3$  since we knew ahead of time that the mixture contained three distinct types of sources. Once we do this, we obtain results generated in Figure 3.3. We can see that the mixture has been separated into three distinct components. The first component corresponds to the first beep, the second component corresponds to the snare drum, and the third component corresponds to the next beep. From the activations, we see that only one of the dictionary components is active given any time interval. This is a desired effect as we do not want any confusion regarding distinct types of sounds. However, we cannot always realize this desired effect as Figure 3.4 illustrates; if we chose  $k = 2$ , we see that the two distinct beeps are represented by one feature and the snare drum is represented by the other feature. We can call this effect under-fitting, where we have a smaller than sufficient dictionary to represent the sounds. There is also over-fitting, where we have too many dictionary elements representing the mixture. In the over-fitting case, we start to see dictionary elements which look similar to each other in Figure 3.5.

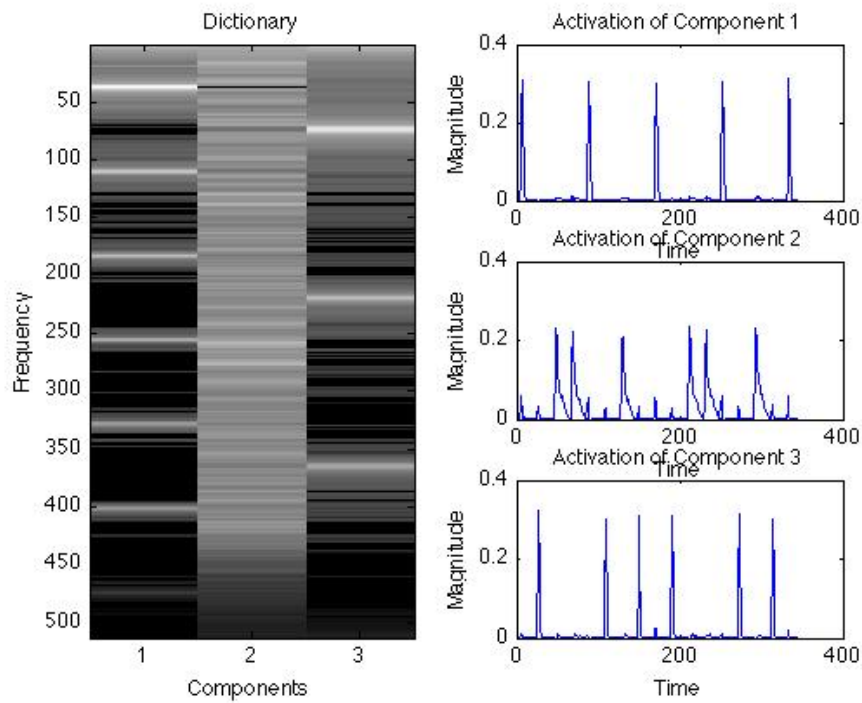


Figure 3.3: Factorized components of an audio mixture spectrogram

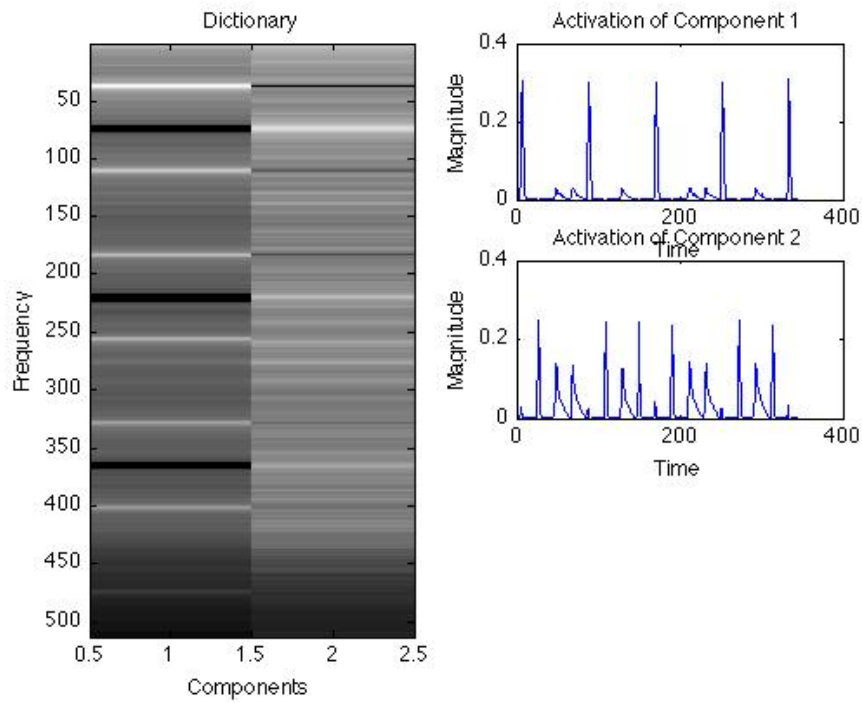


Figure 3.4: Using too few components to decompose mixture

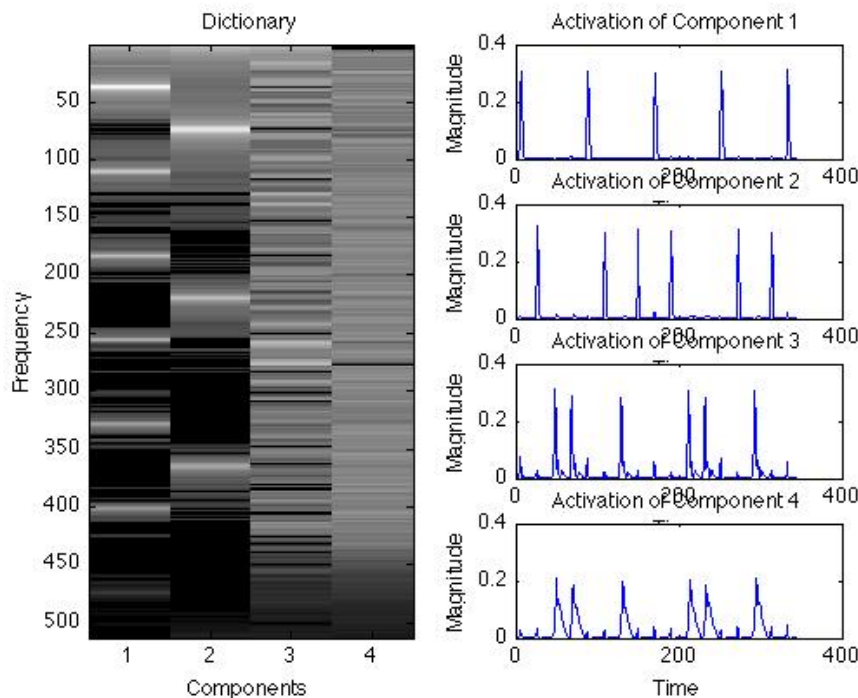


Figure 3.5: Using too many components to decompose mixture

### 3.6 Using NMF for Source Separation

So far, we only looked at the mathematical formulation for NMF and illustrated how NMF can be used for factorizing audio data into distinct and separate components. We will now talk about source separation and discuss the supervised and semi-supervised methods for separation of a mixture composed of a target source and background noise. Again, we will be dealing with magnitude spectrogram data since we are considering the use of NMF. From the previous section, we see that NMF can decompose a mixture of sounds into distinct components. We can take this a step further and isolate the specific sound component by multiplying the dictionaries corresponding to that sound with its respective activations. For example, if we wanted to isolate the first beep we hear from the mixture described in the previous section, we simply take the first column of the dictionary matrix  $W$  and multiply it with the first row of the activation matrix  $H$ . This will give us a matrix containing the magnitude spectrogram of the first beep. The phase information can be appended from the original mixture to the magnitude spectrogram. We can recover this sound by taking the inverse STFT and

obtaining the time-domain signal. In the case of removing background noise from recorded speech, we cannot just apply a direct NMF decomposition to the mixture. The reason is that the NMF decomposition does not give a label to which dictionary elements are speech and which dictionary elements are noise. In order to get around this, we need to learn the speech dictionary components beforehand from a set of clean speaker training samples. We can also learn the noise dictionary if we know ahead of time what background noises will be present. However, this is not necessary, as semi-supervised approaches to noise estimation are possible with the NMF algorithm.

### 3.6.1 Supervised NMF Source Separation

Supervised source separation assumes that we know the noise which will be in the mixture beforehand. In this case, we obtain training data for the noise and speaker. The training data is used to train a set of dictionary elements which will be used later for source separation of a mixture containing the speakers voice corrupted by the known noise. The algorithm is provided in Table 3.1. Though this algorithm might not seem very robust, it can be of

Table 3.1: Supervised Source Separation

| Step | Procedure                                                                         |
|------|-----------------------------------------------------------------------------------|
| 1    | Obtain training data $X_{speech}$ and $X_{noise}$ . (Magnitude Spectrograms)      |
| 2    | Perform NMF decomposition to learn $W_{speech}$ .                                 |
| 3    | Perform NMF decomposition to learn $W_{noise}$ .                                  |
| 4    | Perform partial NMF on $X_{mixture}$ by updating only the $H$ matrix              |
| 5    | Obtain phase information from $X_{mixture}$                                       |
| 6    | After decomposition, gather $S_{speech} = W_{speech}H_{speech}$                   |
| 7    | After decomposition, gather $S_{noise} = W_{noise}H_{noise}$                      |
| 8    | Perform inverse STFT of $S_{speech}$ and $S_{noise}$ to obtain separated sources. |

some use in many environments. For example, if there is engine hum in the cockpit of a truck or aircraft, it is fairly easy to record just the engine hum and learn a noise basis for it. Also, it is not hard to obtain a few seconds of clean speech from a truck driver or pilot. This is just one example of background noise removal via supervised source separation, but there are many more examples of scenarios where we are speaking in an environment where the background is not changing. Supervised source separation should

be able to handle these situations very well.

### 3.6.2 Semi-Supervised Source Separation

In some cases, we might not know the noise we are dealing with or do not have training data for the noise in the mixture. In these cases, it is useful to consider a semi-supervised approach to source separation. "Semi-supervised" means we have training data for either the speaker or noise, and we want to perform separation on a mixture while estimating the dictionary components for the source without training data. Solving this problem is similar to the case of supervised source separation with a slight modification in the update step of the  $W$  matrix. In the supervised source separation procedure, we see that in step 4 in Table 3.1, we estimate only the  $H$  matrix. We can do this because our  $W$  matrix contains the dictionary components for both the speaker and the noise. When we do not have dictionary components for our noise/speech, we simply initialize the noise/speech basis to random values, then use the update equations for NMF in order to reach a suitable dictionary set which minimizes the equation defined by (2.2). The algorithm is outlined in Table 3.2 when we do not have data for the noise. Step 3 is the

Table 3.2: Semi-supervised Source Separation

| Step | Procedure                                                                                        |
|------|--------------------------------------------------------------------------------------------------|
| 1    | Obtain training data $X_{speech}$ and $X_{noise}$ . (Magnitude Spectrograms)                     |
| 2    | Perform NMF decomposition to learn $W_{speech}$ .                                                |
| 3    | Perform partial NMF on $X_{mixture}$ by updating the $W$ matrix, but keeping $W_{speech}$ fixed. |
| 4    | Perform partial NMF on $X_{mixture}$ by updating only the $H$ matrix                             |
| 5    | Obtain phase information from $X_{mixture}$                                                      |
| 6    | After decomposition, gather $S_{speech} = W_{speech}H_{speech}$                                  |
| 7    | After decomposition, gather $S_{noise} = W_{noise}H_{noise}$                                     |
| 8    | Perform inverse STFT of $S_{speech}$ and $S_{noise}$ to obtain separated sources.                |

change made in order to accommodate not having training data for one of the sound sources. Our dictionary matrix  $W$  can be divided into two separate collections of dictionaries representing the speech and the noise. During the

NMF update, we will keep  $W_{speech}$  fixed.

$$W = [W_{speech} W_{noise}] \quad (3.14)$$

In the standard NMF decomposition formulation, there is simply no difference between the order of updating the  $W$  and  $H$  matrices. However, there are some things we need to be cautious of when updating  $W$  and  $H$  in the semi-supervised case. We need to ensure that the columns corresponding to  $W_{speech}$  were not changed by any operations when updating the  $H$  matrix. If we are not careful, then in the  $H$  update step, we will get inaccurate results which do not correspond to the set of speaker/noise bases which we learned using the training data. The reconstruction will not be accurate because it is using another set of dictionary elements instead of the actual dictionary elements which were learned from the training data.

### 3.7 Conclusion

We have provided a literature survey of the recent developments in the area of source separation. We have also provided a literature review of the techniques which have been used in source separation algorithms. In the next chapter, we will discuss the problem of parameter selection for online source separation.

# CHAPTER 4

## PARAMETER SELECTION FOR ONLINE SOURCE SEPARATION

There are many algorithms which have been developed to perform blind source separation. Non-negative matrix factorization (NMF) and principal latent component analysis (PLCA) are among several methods which are exploited in source separation algorithms. Some of these methods are described in [10], [25]. However, in almost all these different algorithms, separation of the mixture occurs only after being provided the entire mixture. Hence, there will have to be some modifications made to these algorithms before they can be used to implement a real-time source separator. Of course on top of the implementation issues, a problem we face is determining how many components we need to keep for speech and noise to achieve the best possible source separation. There has also been debate about what is classified as the best possible separation. Though there have been studies in the past in regards to these questions, the area remains open to research. In regards to the best possible source separation, there is a widely available and used software known as PEASS [18], [26]. Yet, the problem with using the PEASS toolkit for measuring the quality source separation in real-time is that it requires all of the original sound sources to be known. Many of the previous papers on source separation require an estimate or rough number when determining the number of speech and noise components. In many situations, this becomes a guessing game. To illustrate the effects of component selection, we have performed an experiment using the PLCA source separation algorithm on an artificially created mixture of human speech and flute. Figure 4.1 shows an offline source separation algorithm using 30 speech component and 8 noise components. Figure 4.2 shows extracted speech using 30 speech components and 1 noise component. By looking at the spectrograms, we see that using 8 noise components clearly eliminates the flute-like spectral characteristics in the extracted speech. In this chapter, we will present an algorithm which tries to provide a way to determine how many noise components to use given

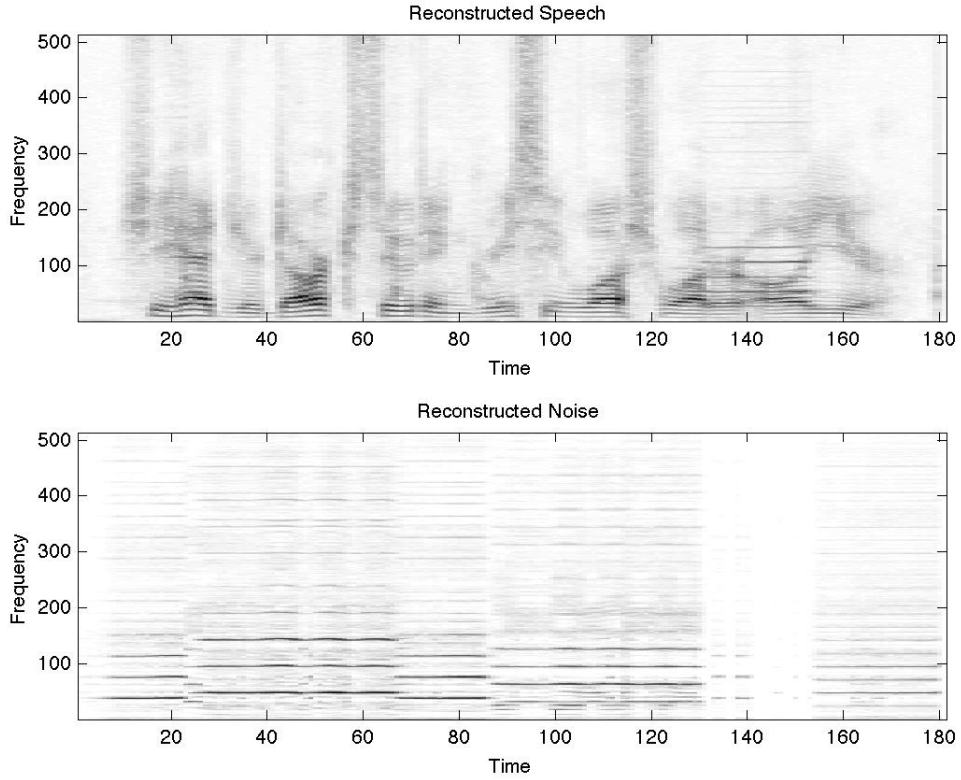


Figure 4.1: 30 speech and 8 noise components

a fixed number of speech basis. We attempt to do this without any prior knowledge of what the original speech and noise signals.

## 4.1 Problem Formulation

The basic premise behind our suggested algorithm stems from the idea of a Gaussian mixture model. In voice recognition systems and voice classification systems, Gaussian mixture models are used to characterize speaker spectral characteristics. Though more modern speech recognition systems use hidden Markov models, we focus on using Gaussian mixture models because they are easy to implement and easy to use. Hidden Markov models are able to capture temporal characteristics of human speech rather than just the spatial characteristics and will be interesting to study in future research.

The problem is to determine the number of dictionary components to learn for both noise and speech in order to achieve the best source separation.

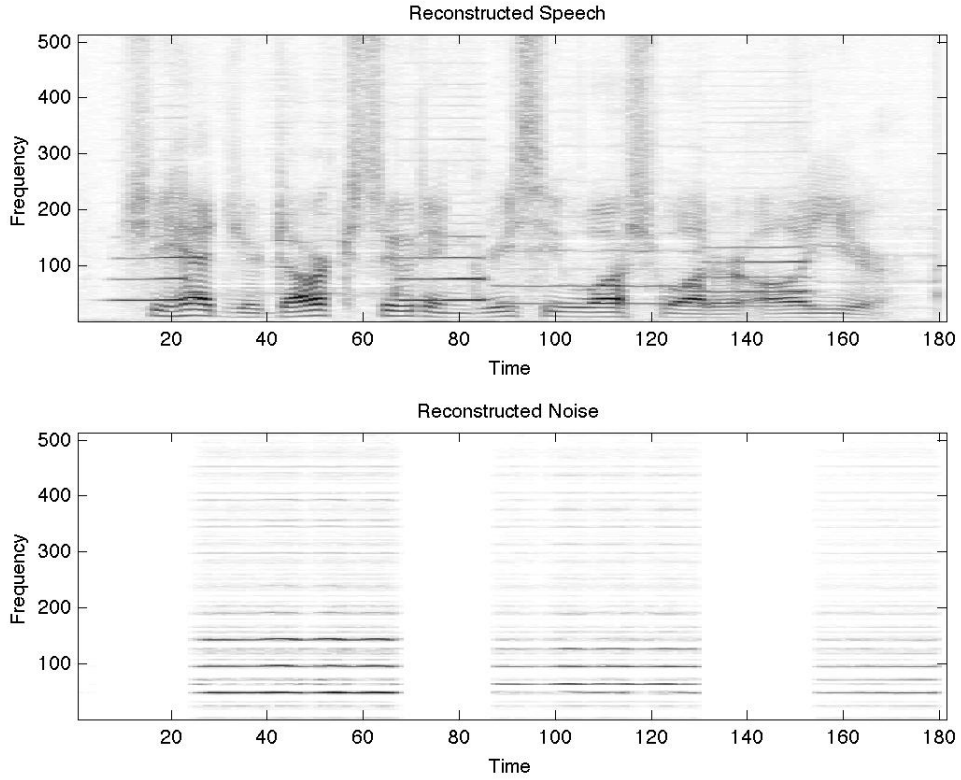


Figure 4.2: 30 speech and 1 noise component

ration. This can be viewed as an optimization problem. Normally, the most commonly used metrics are the signal-to-noise and signal-to-distortion ratio. However, in real-time separation systems, it is impossible to determine these parameters because we do not have the original sounds required to calculate these values. In order to solve this problem, we took a completely different approach by thinking along the lines of classification models. In many source separation algorithms, the primary concern is that the extracted speech should sound like the speaker. In other words, after the source separation, we will compare the spectrogram of the extracted speech to a speech Gaussian mixture model which has been trained on the same speaker. Therefore, we can formulate this in terms of maximizing a likelihood defined by the speech model.

$$\max_{n_c, s_c} P(x(n_c, s_c) | \theta) \quad (4.1)$$

$\theta$  contain the parameters of the Gaussian mixture model,  $x$  is the extracted

speaker spectrogram based on the parameters  $n_c$  and  $s_c$  for number of noise and speech components to keep.  $P(x|\theta)$  gives the likelihood that the extracted speech fits the trained Gaussian mixture model.

## 4.2 Noise Parameter Selection

The solution requires us to train a GMM on the speaker’s voice using clean audio segments. We make an assumption that the input mixture consists of only the speaker we trained on and some varying background noise. In more practical situations, we will want to use training data from a variety of male and female voices to account for variations in spectral characteristics between people, gender, and age. We investigate the use of the principal components of the raw spectrogram and mel-frequency cepstral coefficients (MFCCs) as features in the training. Finally, we analyze the output from using the probabilistic model and compare it to the PEASS toolkit to evaluate the possibility of a substitute which can help us determine the quality of source separation without using the original speaker and noise signals. Table

Table 4.1: GMM Training Procedure for Noise Parameter Selection

| Step | Procedure                                                         |
|------|-------------------------------------------------------------------|
| 1    | Collect clean speech audio segments of speaker                    |
| 2    | Take STFT of the entire audio segment                             |
| 3    | Apply PCA transform to raw spectrogram, (skip 4 and go to step 5) |
| 4    | Apply MFCC transform                                              |
| 5    | Obtain GMM parameters                                             |

4.1 shows the steps to train the GMM. These steps are straightforward and follow the same logic as the derivation of the GMM update equations in the background section. There are also available MATLAB packages which can aid with steps 4 and 5 which are cited in the literature review. Some practical considerations which need to be discussed are the number of Gaussians needed to form the Gaussian mixture model, the necessity of the delta and delta-delta features of the MFCCs, and how many principal components to use if we apply the PCA transformation to our spectrogram data. These questions are answered when we look at the experiments section.

### 4.3 Results for Noise Parameter Selection

Once we generate a Gaussian mixture model to characterize the speaker’s voice, we can evaluate the likelihood that the extracted speech has spectral characteristics similar to the speaker’s voice. We perform the source separation while keeping the number of speech components constant. This is to solely speed up the processing speed of the algorithm. In any practical scenario, it is too expensive to run the search over an entire range of viable speech and noise components. Experiments have shown that the quality of the reconstructed speech largely depends on how many speech components we have learned before performing the separation. We want to put more emphasis on determining the number of noise components, since if we learn a sufficient number of speech dictionaries, we can effectively eliminate the need to learn more speech components. The quality of the separation will then just depend on how many noise components we choose to learn. If this number vastly exceeds the number of speech components, then we have an issue with over-learning the noise dictionary and the quality of the extracted speech decreases as Figures 4.1 and 4.2 illustrate. So this modifies the optimization problem in (4.1) to the following:

$$\max_{n_c} P(x(n_c)|\theta)$$

After running through the experiment, we obtained several figures to illustrate the behavior of using likelihoods to model source separation quality. Figures 4.3, 4.4, and 4.5 show the likelihood score of the GMM model trained on speech to different source separation outputs using different noise components. Figure 4.6 provides the overall perceptual score (OPS) which gives the PEASS tool-kit output of the performance of source separation using differing numbers of noise components. Figure 4.3 shows the GMM model trained on PCA reduced spectrogram data. We see that varying the number of Gaussians in the model does not have an affect on the likelihood outputs. This implies that the data closely resembles a Gaussian and can be described by as few as two independent Gaussian distributions. The same observation is made with the GMMs which were trained on the MFCC features. However, looking at the MFCC features, we see that there is a decrease of the likelihood score when we use more than 8-10 dictionary components for the

noise. By looking at the spectrograms of the extracted speech (Figures 4.2 and 4.3), we see that using just one noise component for source separation gives us an extracted speech which contains a large portion of the flute in the background. This is because a flute playing in the background cannot be properly described by just one dictionary component. By increasing the number of components to learn for the noise, we start to gradually hear less and less of the flute. However, when we increase the number of components to learn to 25 or 30, we start hearing degradations in the extracted speech because the source separation using PLCA starts over-fitting the noise basis. Experiments were performed with several different types of sounds ranging from ambient street noise which is not fast varying to a jackhammer going off in the background. Table 4.2 shows the number of components to keep for noise versus the PEASS-OPS metric.

Table 4.2: Comparison between MFCC GMMs and PEASS OPS

| Noise Type | MFCC GMMs  | PEASS OPS  |
|------------|------------|------------|
| Street     | $n_c = 15$ | $n_c = 24$ |
| Flute      | $n_c = 8$  | $n_c = 10$ |
| Drill      | $n_c = 2$  | $n_c = 7$  |
| Crowd      | $n_c = 12$ | $n_c = 18$ |

From the results obtained in these experiments it seems that the MFCCs can be used as a metric in getting a rough estimate of how many components we would like to use for noise when we are operating without the original speech and background noise.

## 4.4 Analysis of Extracted Noise

We also analyzed the extracted noise in order to see if the added information will help us. During the source separation procedure, it was observed that if we over-fit the noise basis, the extracted noise contains quite a bit of recognizable speech. The original idea for the experiment was to determine how close the extracted speech was to clean speech via the GMM. However, since the GMM contained spectral characteristics of speech, another way

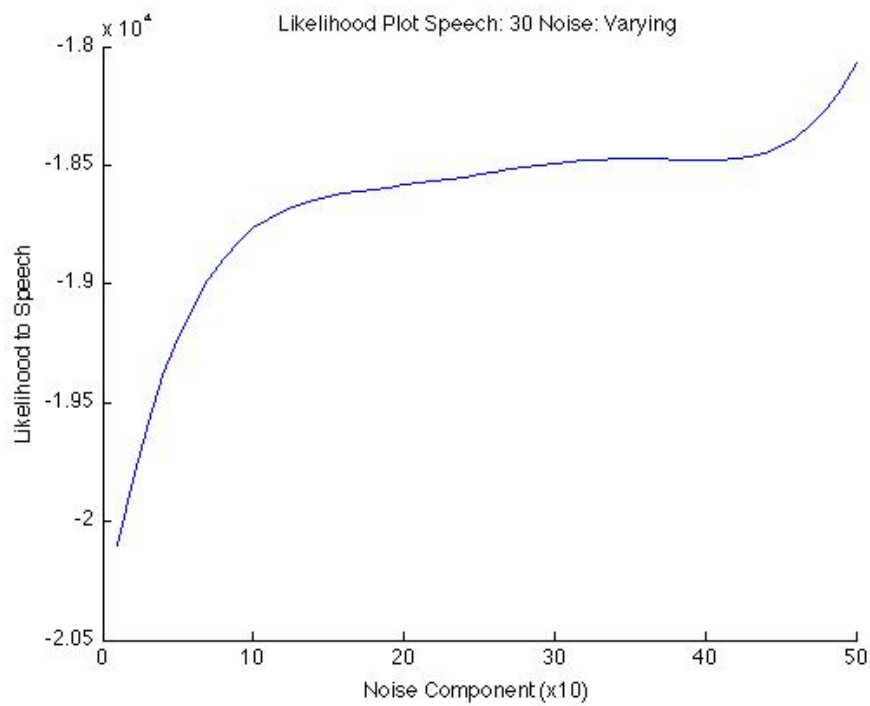


Figure 4.3: Using PCA spectrogram data: 80 PCA dimensions, 10 Gaussians

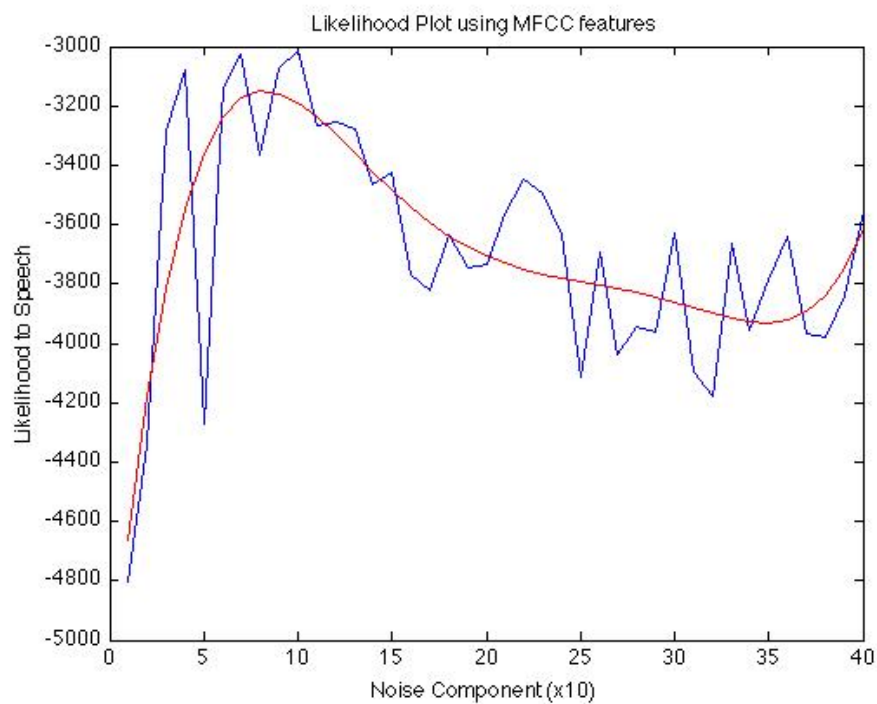


Figure 4.4: Using MFCC spectrogram data with 7 Gaussians

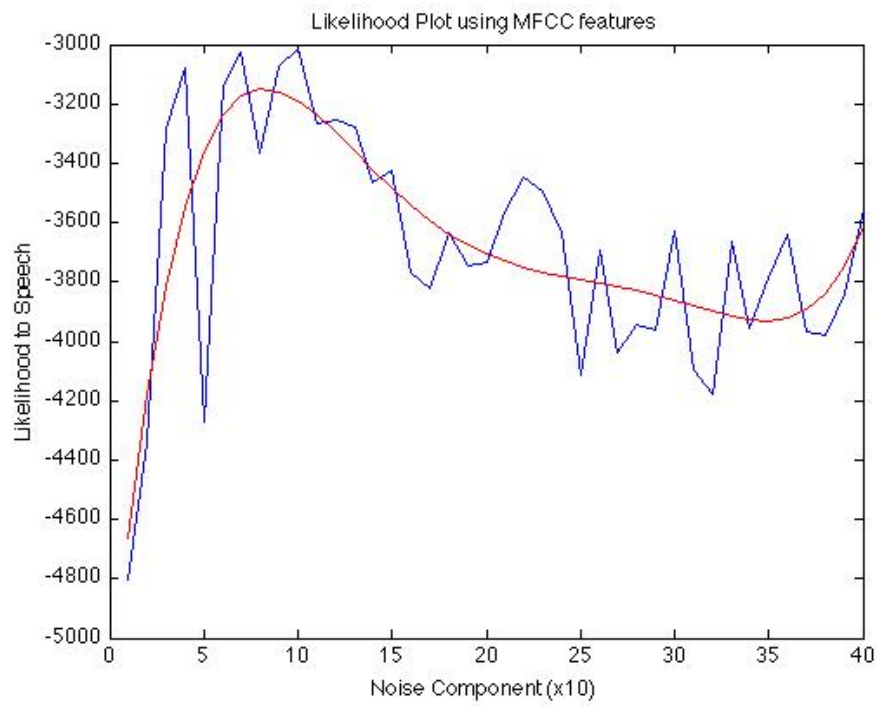


Figure 4.5: Using MFCC spectrogram data with 3 Gaussians

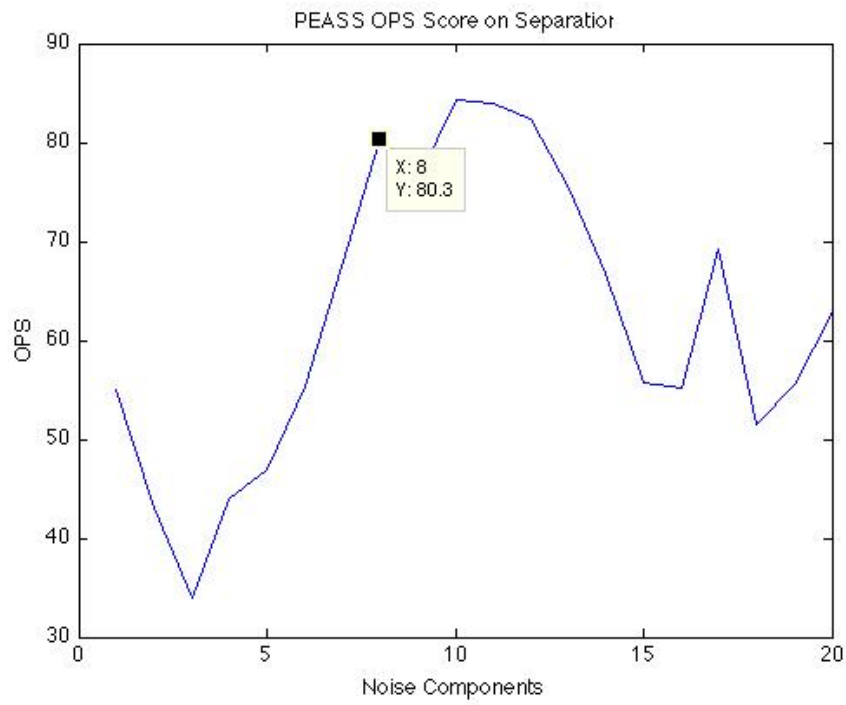


Figure 4.6: Overall perceptual score

to interpret the GMM would be to use it as a pure speech detector. This was done in the past in [27], [28]. We wish to analyze how much speech is contained in the noise. This can be done by using the trained GMM and determining if the spectrogram data had a high relative likelihood. We performed a simple experiment by keeping the speech component constant and varying the number of noise components during the source separation. The noise component varied from 1 dictionary element to 50. We plotted the likelihood value of the extracted noise spectrogram using the PCA reduced GMM and also the MFCC GMM, (see Figures 4.7 and 4.8). The likelihood plots for the GMMs using either the PCA reduced features or the MFCC features agree with our hypothesis that the more noise components we select, the more the extracted noise will contain features similar to speech. Yet, a problem with using the GMM trained on speech as a voice detector in the extracted noise is determining where to draw the threshold when we have too much speech in the extracted noise. Picking the minimum value is not correct in this case because the minimum value will most likely correspond to having just one noise component. We know that this cannot be correct since one component might capture just the main characteristic of the noise but not the entire noise spectrum. Hence, choosing just one noise dictionary to learn might not remove the entire noise signal from the extracted speech spectrum. We instead decided to analyze the rate of change defined by:

$$\frac{dg}{dn_c} = g[n_c + 1] - g[n_c]$$

where  $g[n_c]$  is the likelihood function which is defined by the GMM trained on the MFCC or PCA-reduced features. We set a threshold  $\tau$  by and defined the stop condition as a percentage of desired change. By having this threshold, we can determine the value of  $n_c$  such that when we increase the value of  $n_c$  by 1, the marginal increase in the likelihood increases by  $\tau$ . This is seen in economics as the concept of marginal utility [29]. When we apply the marginal utility to the noise component parameter selection, we believe the value of  $n_c$  chosen becomes the point which does not make the entire extracted noise spectrum seem like speech. At this point, we believe that the noise spectrum has already accurately represented the noise complexity and if we increase the value of  $n_c$  further, we will start to get more and more chunks of speech. Performing a few experiments using this technique

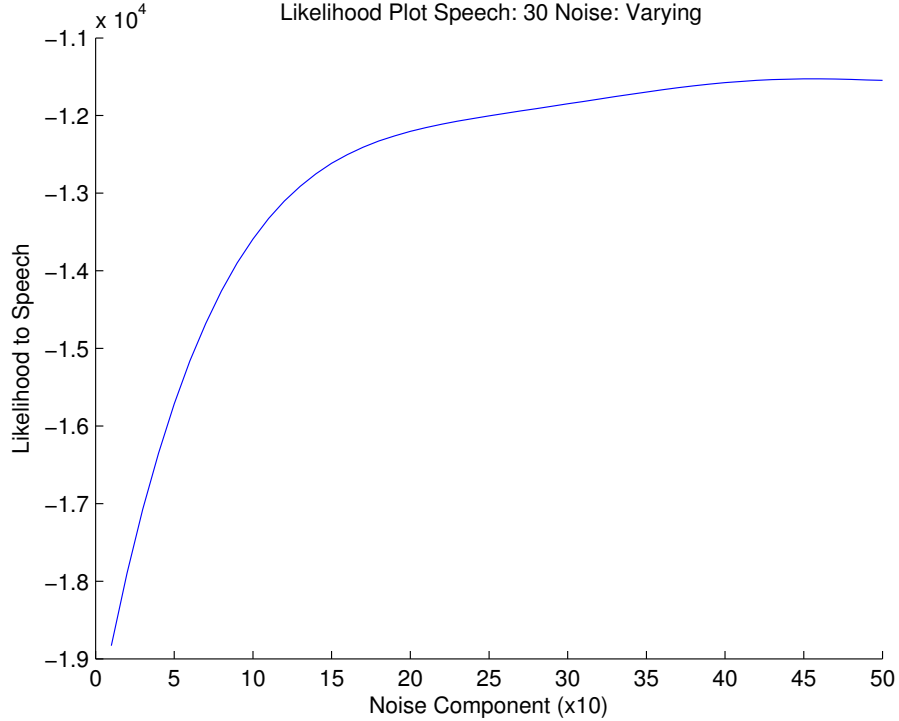


Figure 4.7: Using GMM with PCA-reduced features

on different types of background noise, we obtained the results in Table 4.3.

Table 4.3: Comparison between MFCC GMMs and PEASS OPS

| Noise Type | MFCC GMMs  | PEASS OPS  |
|------------|------------|------------|
| Street     | $n_c = 27$ | $n_c = 24$ |
| Flute      | $n_c = 15$ | $n_c = 10$ |
| Drill      | $n_c = 9$  | $n_c = 7$  |
| Crowd      | $n_c = 26$ | $n_c = 18$ |

## 4.5 Conclusion

In this chapter, we presented a technique which can be used to determine the quality of the source separation by using our proposed statistical model algorithm. Our results indicate that our proposed algorithm has similar performance as the PEASS toolkit when applied to synthetic test experiments. We believe that there is still room for exploration by using more sophisticated statistical models such as HMMs. We will save this for future discussions.

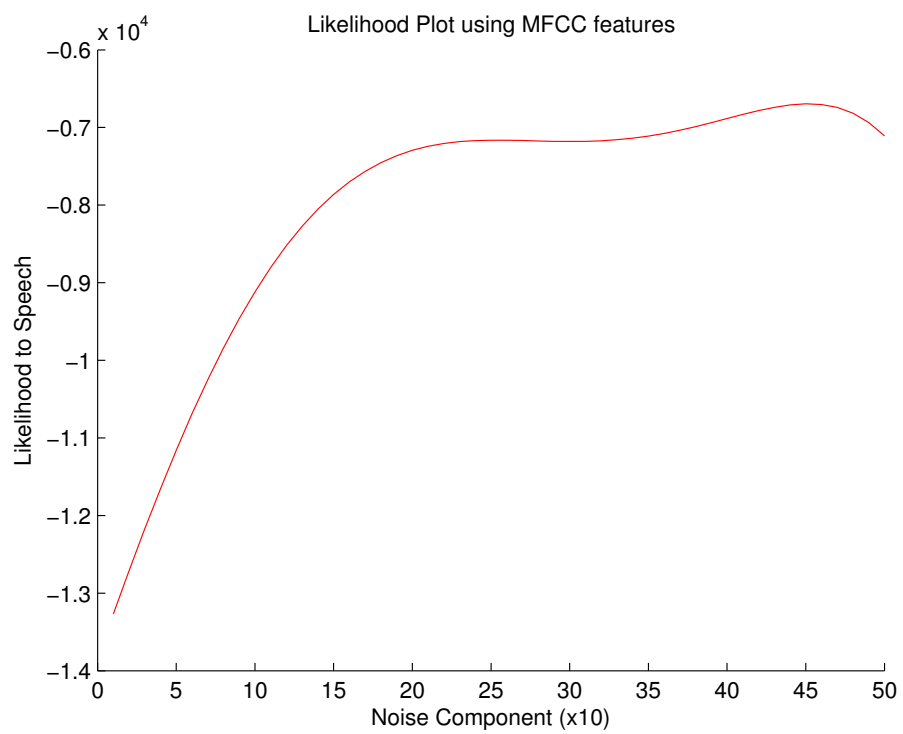


Figure 4.8: Using GMM with MFCC features

# CHAPTER 5

## REAL-TIME SOURCE SEPARATION WITH PARAMETER SELECTION

### 5.1 Introduction

In this chapter, we discuss the problem of separating sources in a real-time setting. More specifically, we discuss a way to implement the parameter selection scheme as a previous step to determine the number of noise components to estimate during each separation step. So far, we have reviewed the current techniques available for source separation in Chapter 1. We have discussed several methods which are used in this thesis for source separation in Chapter 2. However, in all of these algorithms, the mixture signal defined by  $s[n]$  was measured, the short-time Fourier transform was calculated, and we applied matrix factorization techniques on the entire mixture matrix  $X$ . This is not practical in a DSP chip which might be implemented in the real world. There are a few algorithms which look at how to deal with denoising in the real-time setting; however the limitation of these algorithms is that they require us to fix the number of speech and noise components to learn and cannot adapt to noises which vary in complexity over the range of a person talking. For example, if we have background noise which is fairly constant, then it can be described by just one dictionary component, but if the background noise changes to something more complex such as an orchestra concert, we are stuck with describing the concert using just one component, which will result in poor performance from a separation standpoint. There is also the issue of having too many components and over-fitting the noise bases. If we over-fit the noise bases, we observe that parts of speech get removed and the resulting separated speech will be missing higher harmonics, which make it displeasing to the ears. We attempt to address this problem in this chapter. We will first talk about how to do frame-by-frame separation using NMF, then we will describe how to fit the concept defined in Chapter

3 as an addendum to the real-time source separation using NMF.

$$\begin{aligned} W(t, f) &\in \mathbb{R}^{m \times f} \\ X &\in \mathbb{R}^{m \times n} \end{aligned}$$

Let us define  $W(t, f) \in \mathbb{R}^{m \times f}$  as a buffer matrix which contains the  $f$  most recent frames we obtained from our mixture  $X$  and  $t \in \{1, 2, \dots, f\}$ . In this chapter, we assume that the mixture is finite in time; however, the algorithm presented here is easily generalized to mixtures which are infinite. We will start by presenting an algorithm we developed using NMF to do online source separation. The idea behind this algorithm is very similar to that behind the algorithm outlined in [10]. Note that we use the term “online NMF” here, because the update equations used in this implementation are the NMF update equations instead of the PLCA update equations. These have been stated as numerically identical in [10]; therefore, we used the NMF update equations for their simplicity and speed. However, the separation algorithm requires us to normalize the columns of the input matrix  $X$  before doing the updates of the NMF.

## 5.2 Online Source Separation using NMF Decomposition

Starting with the spectrogram, we have the running buffer  $W(t, f)$  which will contain  $f$  most recent frames of the STFT from the input mixture signal  $X$ . We perform separation on this buffer using the NMF source separation technique described in Chapter 2. Once we have performed this operation, we obtain a reconstructed speech signal and a reconstructed noise signal for the current buffer. The last frame of the current buffer is the most recent input frame that has been denoised. We can take the inverse STFT of this frame and output a sound signal of the denoised output. We make a slight modification and replace the current frame estimate with an average of the previous  $f$  buffer frames. We do this because the NMF algorithm is a modified gradient descent algorithm on the cost functions defined by equations (2.3) and (2.4). It is unclear if each specific column (frame of the

spectrogram) is accurately represented with respect to the columns of the actual data,  $X$ . We attempt to solve this uncertainty by generating multiple estimates for the current frame as it moves through the buffer. For example, consider Figure 5.1. The top row represents the denoised buffer at time instant  $t = 3$ , and the second row represents the denoised buffer at time instant  $t = 4$ . We can see that the last frame when  $t = 3$  is the second to last frame at  $t = 4$ . This means that each frame which goes through the denoising algorithm is actually reconstructed  $f$  times due to the buffer size. We can use this information to construct a much better estimate of the current frame given a delay of  $f$  frames. This algorithm is outlined in Table 5.1.

Table 5.1: Online NMF Source Separation

| Step | Procedure                                                                                                                                                                                                    |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1    | Obtain dictionary for $W_{speaker}$ using training data for clean speech.                                                                                                                                    |
| 2    | Initialize $W(t, f)$ buffer and also create an averaging buffer $\mathcal{B} \in \mathbb{R}^{m \times f}$ .                                                                                                  |
| 3    | while not at end of mixture spectrogram $X$<br>$W(t, f) = [W(t-1, f) \ X(:, i)]$<br>$i = i+1$<br>Separate buffer $W(t, f)$ into $X_{speaker}$ and $X_{noise}$<br>Load $W(t = i, f)$ into $\mathcal{B}(:, i)$ |
| 4    | if( $i == f$ )<br>$\hat{X}_{speaker}(:, i - f + 1) = average(\mathcal{B})$                                                                                                                                   |
| 5    | Reconstruct $x_{speaker} = ISTFT(\hat{X})$                                                                                                                                                                   |

The average operation in step 4 is the average across the columns of  $\mathcal{B}$ . Also, we note that  $X$  here denotes the normalized spectrogram of the mixture. This can be obtained when we take a segment of the input signal, then we take the fast Fourier transform once we have acquired enough samples of the input signal. We then normalize the fast Fourier transform by the column sum.

### 5.3 Algorithm Performance

We ran the algorithm separation and compared it with results of using the semi-supervised offline NMF separation algorithm.

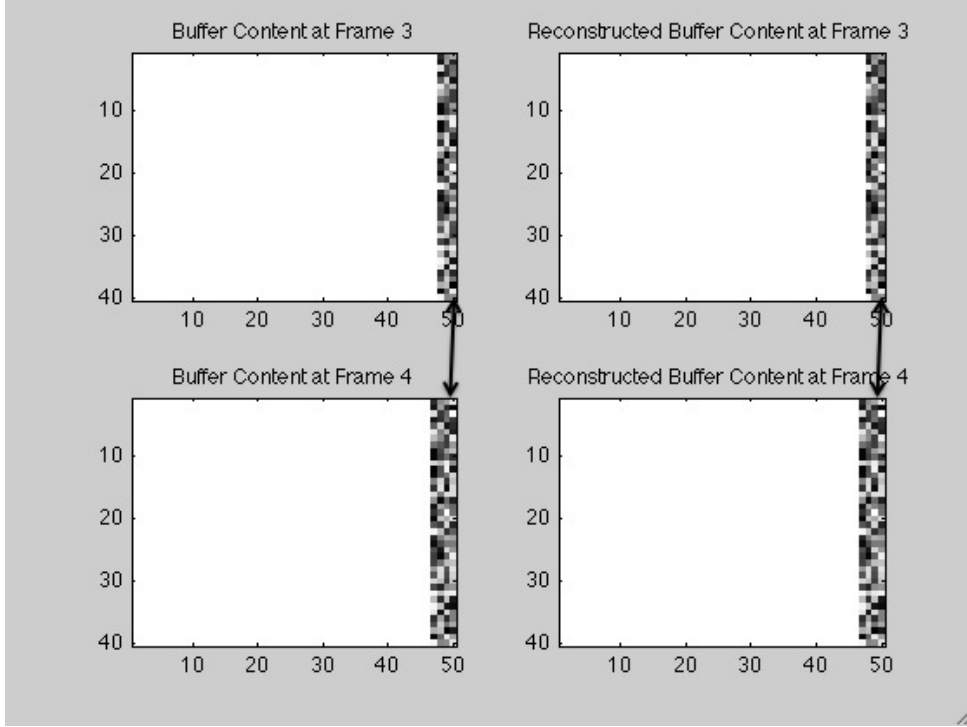


Figure 5.1: Online algorithm for NMF source separation

Table 5.2: Online NMF Performance

| Noise Type | SDR (Offline) | SDR (Online) | SAR (Offline) | SAR (Online) |
|------------|---------------|--------------|---------------|--------------|
| Street     | 13.5 dB       | 12.7 dB      | 13.5 dB       | 12.7 dB      |
| Flute      | 14.6 dB       | 14.4 dB      | 14.5 dB       | 13.7 dB      |
| Drill      | 17 dB         | 16.5 dB      | 17 dB         | 15.2 dB      |
| Crowd      | 18 dB         | 16.6 dB      | 18.3 dB       | 14.2 dB      |

The performance, summarized in Table 5.2, indicates that separation using the online NMF algorithm achieves similar performance as the semi-supervised NMF source separation algorithm. The mixtures were generated after normalizing both the speaker and noise to have unit energy.

## 5.4 Discussion of Online Parameter Selection Techniques

In order to select the number of noise components in a real-time source separation algorithm, we need a method of feedback to tell us how the algorithm is performing at any given instant in time. Given that the noise profile can

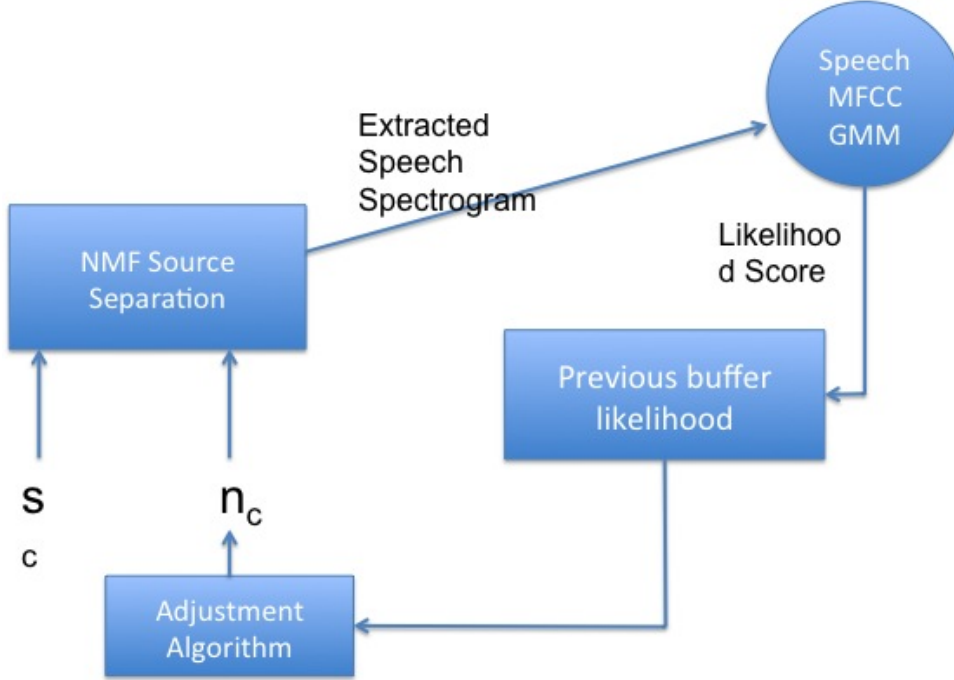


Figure 5.2: Online parameter selection

change in the real-time setting, this method of feedback should be able to adapt to any given noise regardless of complexity. Since we already have the online separation algorithm, we can use the GMM likelihood model as the method of feedback. We can describe our algorithm in a two-stage procedure. The first stage is the evaluation of the buffer and the second stage is the online denoising based on the output from the first stage. We will now explain the first and second stages with Figure 5.2.

The first stage is the box labeled previous buffer likelihood. We compare the previous buffer likelihood score with the current buffer likelihood score. The likelihood score is computed by using the posterior likelihood for the speech MFCC GMM. If we determine that the previous buffer likelihood score is higher than the current buffer likelihood score, we stay with our current value to denoise the updated buffer. If the previous buffer likelihood score is lower than the current buffer likelihood score, we increase the noise component by a fixed number  $\eta$ . Normally  $\eta = 1$ , but we can be more aggressive in our search by increasing  $\eta > 1$ . Increasing  $\eta$  will help in the scenario where the noise profile changes very fast and the algorithm needs to adapt to the fast-changing noise profiles. For stationary or very slow-changing noise profiles,  $\eta$  being large will not necessarily help us in obtaining

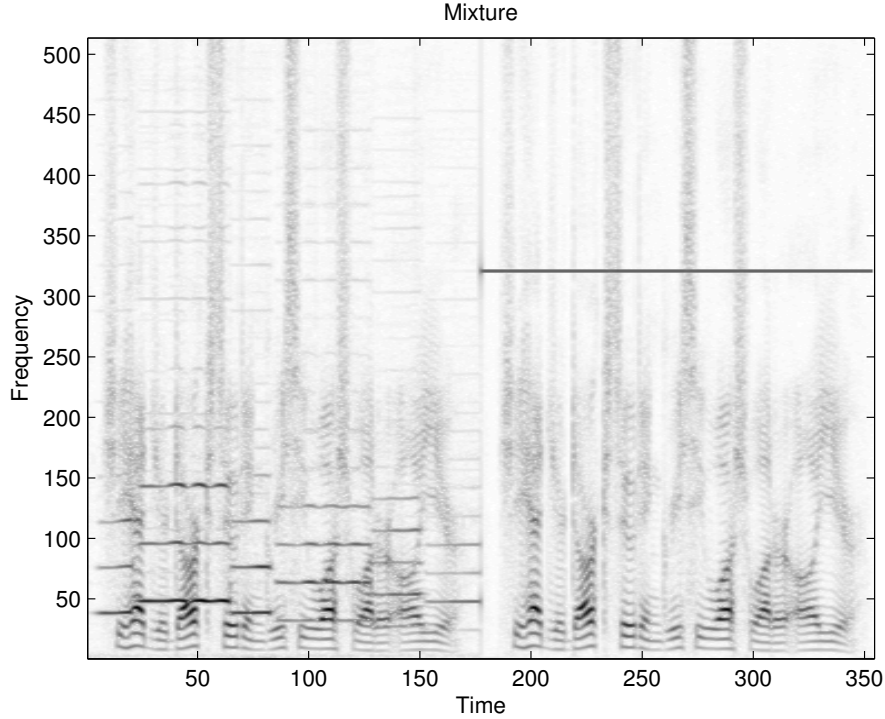


Figure 5.3: Mixture spectrogram

the best performing algorithm. To illustrate an example of this algorithm in action, we have created a long mixture by concatenating noises on top of a speaker speaking the same sentence. The spectrogram for the mixture is provided in Figure 5.3. In this mixture, we see that the noise profile changes from the flute to a single sine tone. We attempt to denoise this using the proposed algorithm with the first and second stages determining how many noise components we need to keep. Our buffer size is 50 and we start at an initial value for  $n_c = 1$ . Our step size parameter is  $\tau = 1$ . We obtained the plot in Figure 5.4 for the noise components which were selected based on a GMM trained on the MFCC components using 4 Gaussian mixtures.

This plot shows us that the algorithm made an attempt to roughly estimate how many parameters to use for the noise component. The extracted spectrograms for both speech and noise are shown in Figure 5.5. We see that the algorithm took on average about 30 frames to determine if it needed to adjust the number of noise components to learn. The main observation is that the noise components were affected by the noise complexity. We see that when the algorithm detected a sine tone in the mixture, it reduced the

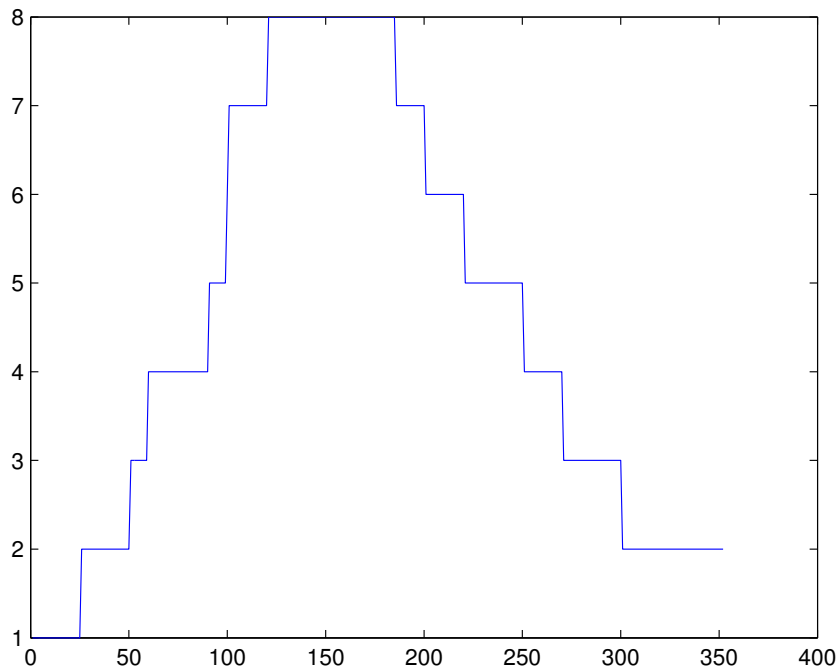


Figure 5.4: Noise component for real-time denoising

noise component to 1. After the noise profile changed to a flute the noise component increased to 8. We believe that there are more efficient ways to perform the switching strategy when the algorithm detects a change in the noise, but for now, it seems that this algorithm can roughly estimate the number of noise components to use in an online setting.

## 5.5 Conclusion

In this chapter, we showed that we can roughly estimate the number of noise components to use in a real-time setting. Our algorithm provides feedback on how well the source separation is doing whenever a new frame comes in. So far, we have analyzed mixtures which are artificially made, but we hope to implement this on a DSP processor and have it run real-time for future studies. We also hope to see how well this algorithm performs if we train our models using a different voice than what is in the mixture. We think this is of interest because in some applications, there may be many speakers and

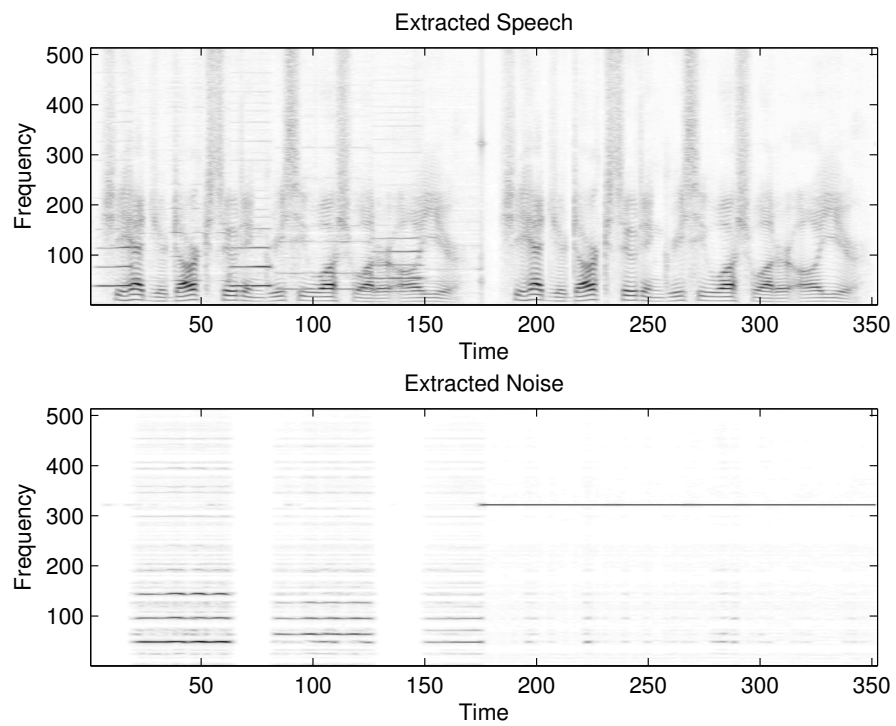


Figure 5.5: Noise component for real-time denoising

we will not have time to acquire training samples for each speaker.

# CHAPTER 6

## CONCLUSION

We have presented a new method by which to determine the parameter selection problem in blind source separation. We have presented an online algorithm to separate sources which uses our method of parameter selection. We demonstrated that this algorithm works well when presented with synthetic audio mixtures. The next step would be to try and implement the algorithm in C code and see how it performs in real-time conditions. We believe the work presented in this document is a first step to solving the problem of selecting the number of parameters in given source separation algorithms. A direction of future research would be to perform the experiments in this thesis using a hidden Markov model instead of a Gaussian mixture model. We believe that the hidden Markov model should be more robust in detecting speech in the extracted noise mixture by recognizing the temporal characteristics of the speaker. Another interesting direction to pursue is the optimization procedure in determining the size of the jump when we need to increase or decrease the number of noise components to use in the algorithm described in the previous chapter. Since we are interested in phone applications, we would ultimately hope to program a DSP processor on a phone and see how the algorithm performs in limited processing power and memory settings.

## REFERENCES

- [1] J. Meyer and K. Simmer, “Multi-channel speech enhancement in a car environment using wiener filtering and spectral subtraction,” in *Proc. ICASSP*, 1997, pp. 1167–1170.
- [2] S. Boll, “Suppression of acoustic noise in speech using spectral subtraction,” in *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 27, no. 2, 1979, pp. 113–120.
- [3] K. Paliwal and A. Basu, “A speech enhancement method based on kalman filtering,” in *ICASSP*, vol. 12, 1987, pp. 177–180.
- [4] M. Gabrea, “A single microphone kalman filter-based noise canceller,” in *IEEE Signal Processing Letters*, 1999.
- [5] J. Sohn, N. S. Kim, and W. Sung, “A statistical model-based voice activity detection,” in *IEEE Signal Processing Letters*, vol. 6, no. 1, 1999, pp. 1–3.
- [6] P. Smaragdis, “Convolutive speech bases and their application to supervised speech separation,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 1, pp. 1–12, 2007.
- [7] T. 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.
- [8] B. Raj, M. Shashanka, and P. Smaragdis, “Latent dirichlet decomposition for single channel speaker separation,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 5, 2006, pp. 14–19.
- [9] M. N. Schmidt and M. Morup, “Nonnegative matrix factor 2-d deconvolution for blind single channel source separation,” in *ICA2006*, 2006.
- [10] Z. Duan, G. J. Mysore, and P. Smaragdis, “Online plca for real-time semi-supervised source separation,” in *International Conference on Latent Variable Analysis/Independent Component Analysis*, 2012, pp. 34–41.

- [11] J. S. Lim and A. V. Oppenheim, “Enhancement and bandwidth compression of noisy speech,” in *Proceedings of the IEEE*, vol. 67, no. 12, 1979, pp. 1586–1604.
- [12] S. V. Vaseghi, *Advanced Digital Signal Processing and Noise Reduction*, 4th ed. John Wiley and Sons, 2008.
- [13] A. V. Oppenheim and R. W. Schaffer, *Digital Signal Processing*. Prentice-Hall, 1975.
- [14] J. G. Proakis and D. K. Manolakis, *Digital Signal Processing*. Prentice-Hall, 2006.
- [15] J. B. Allen and L. R. Rabiner, “A unified approach to short-time fourier analysis and synthesis,” in *Proceedings of the IEEE*, vol. 65, no. 11, 1977, pp. 1558–1564.
- [16] K. Pearson, “On lines and planes of closest fit to systems of points in space,” *Philosophical Magazine*, vol. 2, pp. 559–572, 1901.
- [17] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
- [18] V. Emiya, E. Vincent, N. Harlander, and V. Hohmann, “Subjective and objective quality assessment of audio source separation,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 7, pp. 2046–2057, 2011.
- [19] V. Emiya, E. Vincent, N. Harlander, and V. Hohmann, “Multi-criteria subjective and objective evaluation of audio source separation,” in *AES 38th International Conference*, 2010.
- [20] I. T. U. T. (ITU-T), “Itu-t recommendation p.835: Subjective test methodology for evaluating speech communication systems that include noise suppression algorithm,” 2003.
- [21] S. S. Stevens, J. Volkman, and E. B. Newman, “A scale for the measurement of the psychological magnitude pitch,” *Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 208–208, 1937.
- [22] T. Hofmann, “Probabilistic latent semantic indexing,” in *Proceedings of the 22nd Annual International SIGIR Conference on Research and Development in Information Retrieval*, 1999, pp. 50–57.
- [23] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*. New York: John Wiley and Sons, 1996.
- [24] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *NIPS*, 2000, pp. 556–562.

- [25] A. Cichocki, R. Zdunek, and S.-I. Amari, “New algorithms for non-negative matrix factorization in applications to blind source separation,” in *ICASSP*, vol. 5, 2006, pp. 14–19.
- [26] E. Vincent, “Improved perceptual metrics for the evaluation of audio source separation,” in *10th Int. Conf. on Latent Variable Analysis and Signal Separation*, 2012, pp. 430–437.
- [27] H. Hall, “A gaussian mixture model spectral representation for speech recognition,” Ph.D. dissertation, Cambridge University Engineering Department, 2003.
- [28] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, “Speaker verification using adapted gaussian mixture models,” *Digital Signal Processing*, vol. 10, no. 19-41, 2000.
- [29] L. Rittenberg and T. Tregarthen, *Principles of Microeconomics*. Flat World Knowledge, 2009.