Constrained Nonnegative Matrix Factorization with Applications to Music Transcription

by

Daniel Recoskie

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2014

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In this work we explore using nonnegative matrix factorization (NMF) for music transcription, as well as several other applications. NMF is an unsupervised learning method capable of finding a parts-based additive model of data. Since music has an additive property (each time point in a musical piece is composed of a sum of notes) NMF is a natural fit for analysis. NMF is able to exploit this additivity in order to factorize out both the individual notes and the transcription from an audio sample.

In order to improve the performance of NMF we apply different constraints to the model. We consider sparsity as well as piecewise smoothness with aligned breakpoints. We show the novelty of our method on real music data and demonstrate promising results which exceed the current state of the art. Other applications are also considered, such as instrument and speaker separation and handwritten character analysis.

Acknowledgements

I would like to thank Dr. Richard Mann for his time and support.

Table of Contents

List of Tables

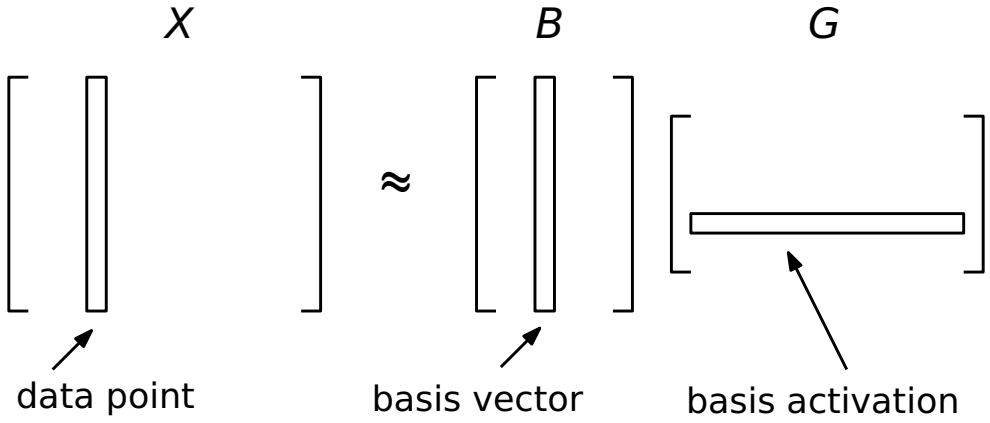
List of Figures

Chapter 1

Introduction

1.1 Matrix factorization

The goal of matrix factorization is to represent a matrix as the product of other matrices. Here, for a data matrix X, we consider the factorization



That is, we wish to approximate X as the product of two smaller matrices. There are many methods of matrix factorization, but here we focus on only one: nonnegative matrix factorization.

Why only consider nonnegative matrices? We will see throughout this thesis that many real signals are composed as a sum of nonnegative parts. Hence, it is natural to analyze these signals using a nonnegative method.

1.2 Motivation from real data

Lee and Seung popularized nonnegative matrix factorization (NMF) as a method for parts-based analysis [29]. The authors consider the problem of facial image analysis. In their experiments they compare three different methods of matrix factorization. Principal component analysis (PCA), vector quantization (VQ), and nonnegative matrix factorization (NMF). The authors run the three methods over a database of faces to find 49 basis vectors which make up the faces as shown in Figure 1.1. VQ finds basis vectors that are whole faces. PCA finds basis vectors that contain both positive and negative parts. NMF finds nonnegative basis vectors. Each of these basis vectors corresponds to a recognizable part of a face (eyes, nose, mouth). This experiment demonstrates the usefulness of NMF when the data is composed of nonnegative, additive parts.

In our work we will be focussing on applying NMF to the analysis of music. Like the face data from Lee and Seung's experiment, audio signals have additive components. We can visualize these components most readily if we consider a time-frequency representation of our data. Figure 1.2 shows the magnitude of the short-time Fourier transform (also known as a spectrogram) of a short musical piece. Note how each data point of the spectrogram is composed of a nonnegative sum of several basis vectors. Each basis vector in the figure, excluding the first noisy vector, corresponds to a single musical note. Each note is itself the sum of a number of harmonic waves which are seen as distinct frequency bands. We will explain how to find these basis vectors later in the section.

We can exploit the additive nature of music by applying NMF to the task of music transcription. For this task, we are given an audio recording of music and we wish to generate the corresponding sheet music. In general, music transcription is difficult. However, identifying single notes played in isolation is relatively easy. Figure 1.3 shows a pressure-time wave of a single piano note. Note that the wave is not a pure sinusoidal wave, but rather a sum of sinusoidal waves (harmonics) whose frequencies are whole number multiples of the fundamental frequency of the note. Figure 1.4 illustrates this summation using the first four harmonics.

We can view the note in Figure 1.5a in terms of its harmonics by considering its Fourier transform. In this view, we can identify the note being played by finding the frequency of the lowest frequency peak. Such a strategy works well for single notes. However, if multiple notes are played together, our task becomes much more difficult. Figure 1.5b shows the magnitude of the Fourier transform of three notes played together. Note how it is not obvious from the Fourier transform how many notes are being played, or which frequency peaks correspond to the three fundamental frequencies.

The additive nature of music means that NMF will be an excellent method for analysis. Figure 1.6 outlines the process. We first take the magnitude of the short-time Fourier transform of our audio data in order to get it into a representation suitable for NMF. Applying NMF on the spectrogram results in the factorization as shown in Figure 1.6. The basis matrix B is composed of the basis vectors of the data. In this case the basis vectors are notes (as well as a vector corresponding to a note onset). Note that several of the notes found in B are never played individually in the music sample. However, NMF is still able to find them. Since each column of B corresponds to a single note, the activation matrix G corresponds to the transcription. Note the correspondence between the underlying sheet music and G.

Now let us suppose we wish to improve the transcription matrix G. We can do so by imposing constraints on it. One constraint we may consider is to impose smoothness on the rows of G. Doing so may remove spurious note activations. However, smoothing over each row entirely will make it difficult to determine note onsets and offsets. Instead, we would like to favour rows of G that are piecewise smooth. See Figure 1.7 for an illustration. Note how the piecewise smooth constraint allows G to have sharp note onsets and offsets.

1.3 NMF in the literature

NMF was introduced by Paatero and Tapper [36] as positive matrix factorization. Paatero improved on their previous work in [35]. The method gained popularity due to work by Lee and Seung [29, 30]. The notion of NMF was seen even earlier under the name 'self modelling curve resolution' [26]. Since then, NMF has been applied to many different applications including: bioinformatics [40], geophysics [41], stock market analysis [18], digit and texture classification [16], and music transcription [38]. See [6, 12, 13, 46] for a selection of reviews of NMF. This thesis will focus on the application of NMF to music transcription.

1.3.1 NMF based music transcription

Smaragdis and Brown [38] introduced the idea of using NMF for music transcription. After Smaragdis and Brown's original paper, there has been continued work in the area. Cont [15] developed a realtime transcriber utilizing sparsity constraints. Virtanen [44, 45] makes use of both sparsity and temporal smoothness constraints. Vincent et al. [43] introduce harmonic constraints on the factorization to improve the learned basis vectors. Bertin et al. [8] use a Bayesian approach to enforce harmonicity and smoothness constraints. Ochiai et al. [32] make use of beat structure to improve their NMF based transcription.

An alternative but similar method to NMF, probabilistic latent component analysis (PLCA), was used for acoustic modelling by Smaragdis and Brown [39]. Their method was extended by Grindlay and Ellis [22] to support multiple instrument transcription. Benetos and Dixon [4] both extended the PLCA method by considering a convolutive probabilistic model. Fuentes et al. [21] also extend the PLCA method to overcome the difficulties of components having time-varying fundamental frequencies and spectral shapes.

Another technique similar to NMF is sparse coding. Abdallah and Plumbley [2] introduce a method of spectral basis decomposition for use in music transcription. Bertin et al. [7] investigate both NMF and k-means singular value decomposition for music transcription. O'Hanlon et al. [33] attempt music transcription using a nonnegative greedy sparse pursuit based method. Lee et al. [28] make use of an exemplar-based sparse representation which eliminates the need for retraining their classifier. For a review of music transcription methods, including those not related to NMF, see [5].

1.4 Outline

In this thesis we will consider various constraints imposed on G and explore their effect on both synthetic and real world applications. Chapter 2 introduces time-frequency analysis. Chapter 3 describes NMF and our proposed constraints. We test our method on synthetic data in Chapter 4 and on real data in Chapter 5. Finally, Chapter 6 contains our concluding remarks.

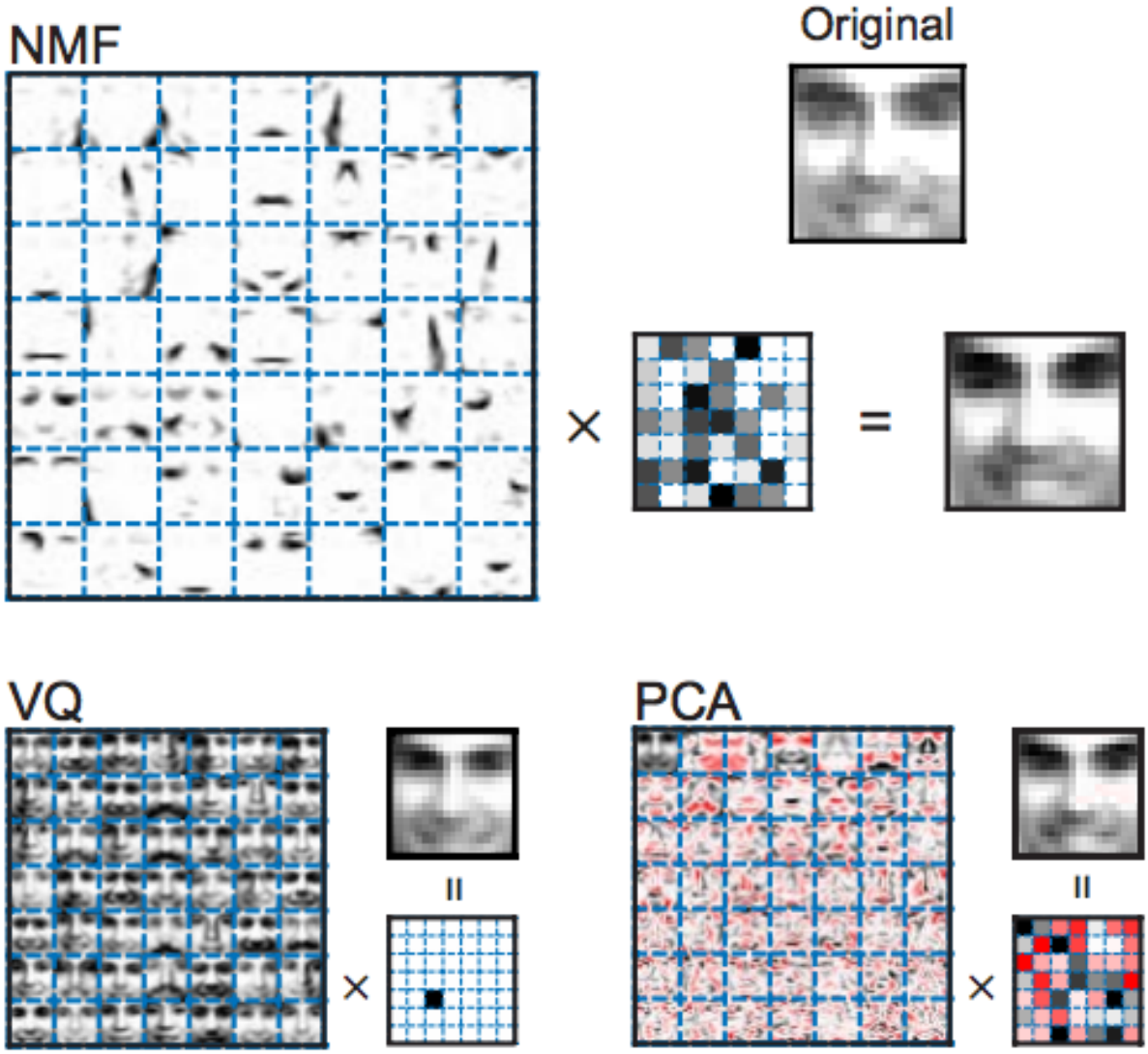


Figure 1.1: Figure from Lee and Seung face analysis [29]. Three methods are compared: Principal component analysis (PCA), vector quantization (VQ), and nonnegative matrix factorization (NMF). Shown in the figure are the 49 basis vectors found by each method as well as the basis weights corresponding to a single face. Note how NMF is able to find parts-based basis vectors.

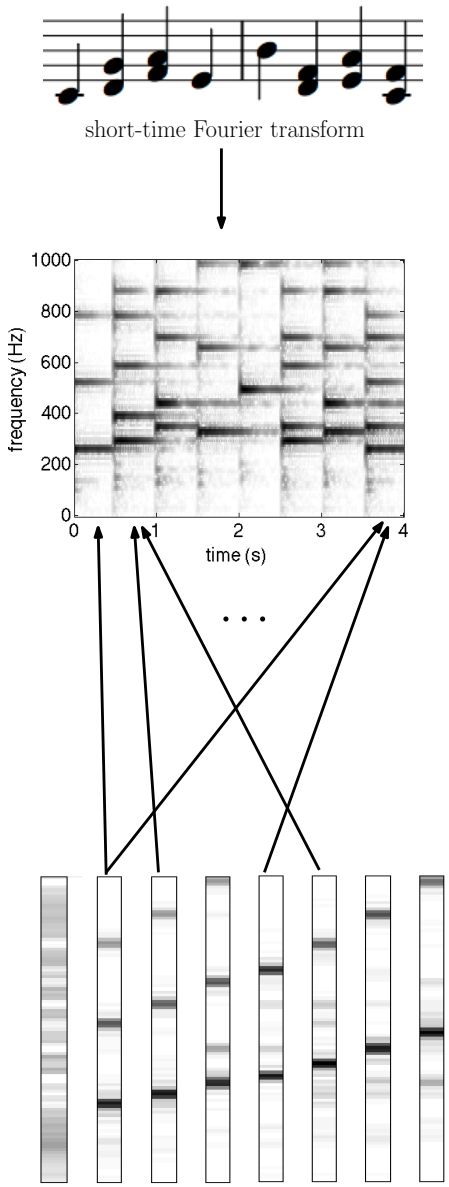


Figure 1.2: The magnitude of the short-time Fourier transform of an audio music sample is taken as the matrix V. NMF is performed to find the basis vectors at the bottom of the image. These vectors are the columns of B. The arrows representing basis activation correspond to the entries of G.

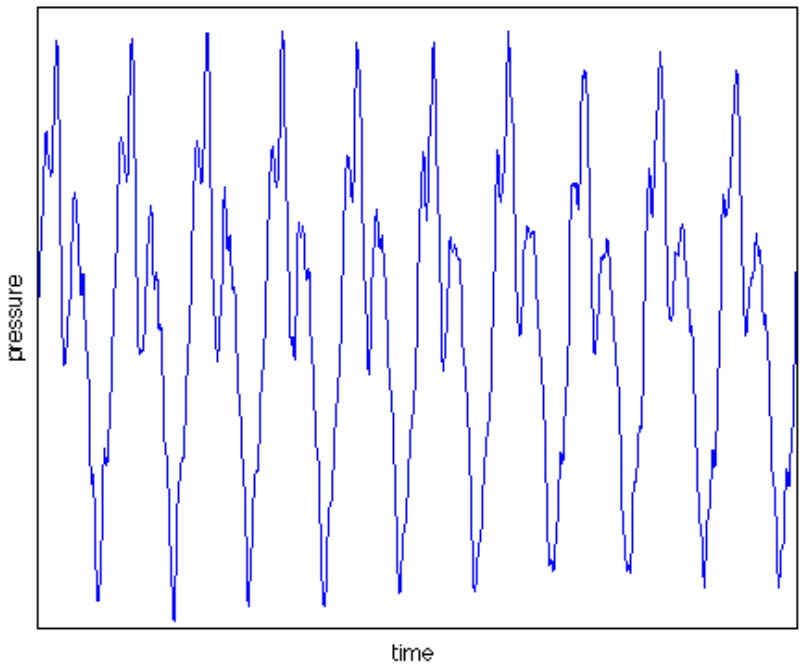


Figure 1.3: Pressure-time wave corresponding to a single piano note.

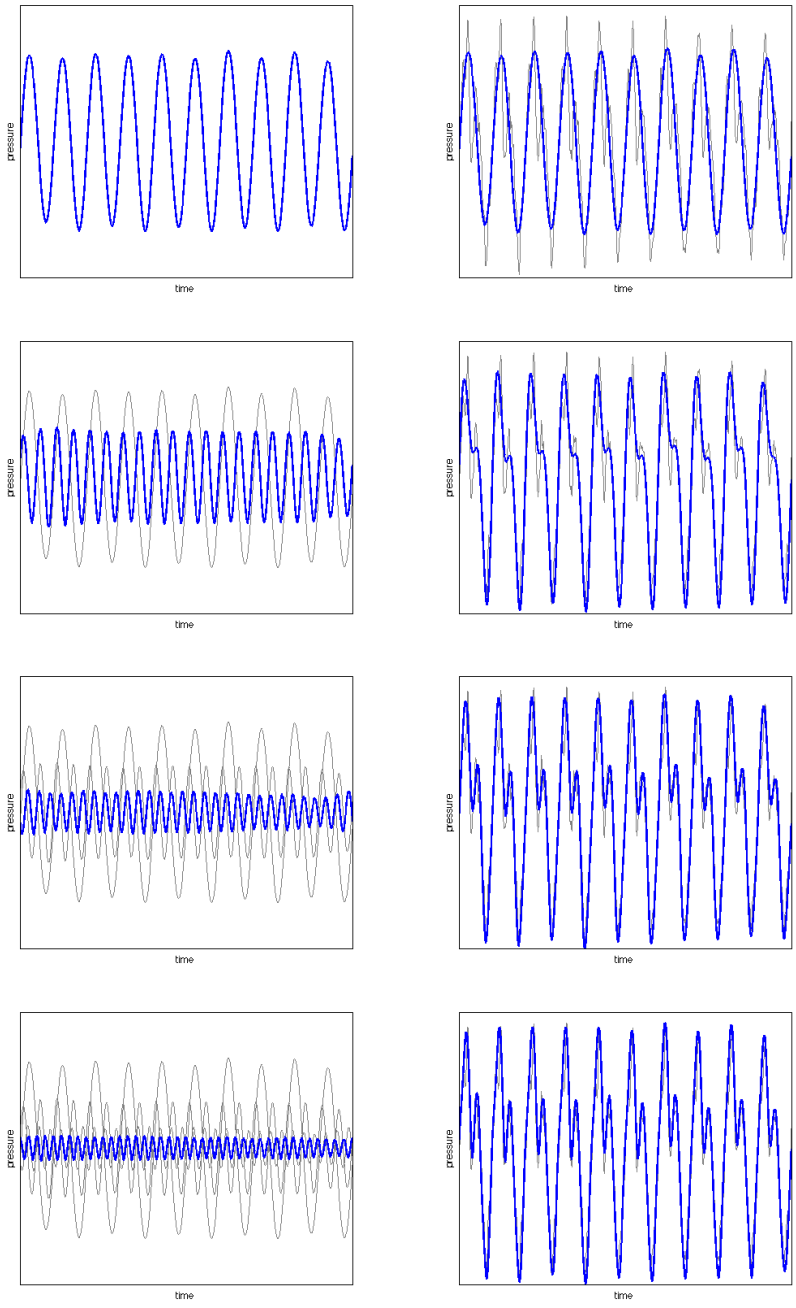
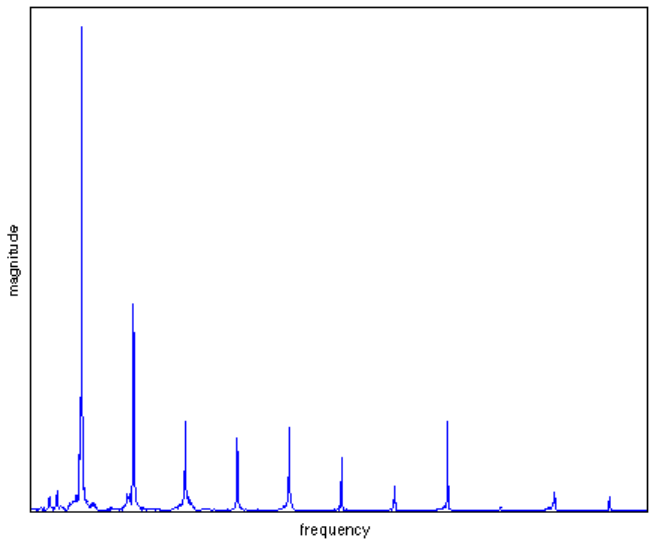
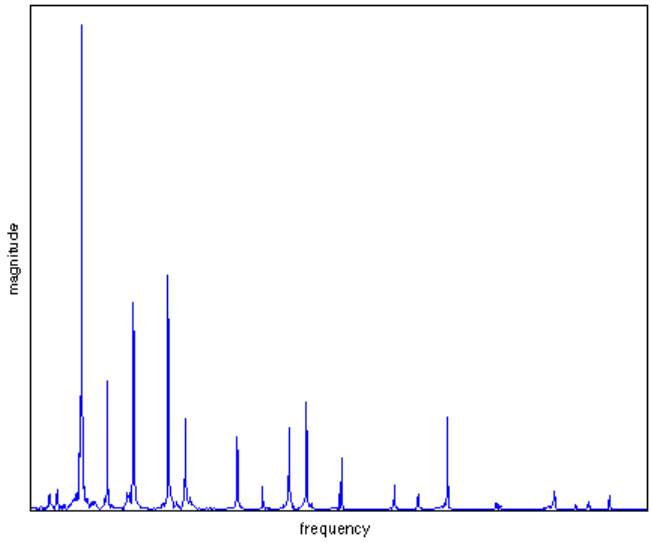


Figure 1.4: The summation of the first four harmonics of a piano note. The plots on the left show each individual harmonic. The plots on the right show the sum of the harmonics.



(a)



(b)

Figure 1.5: Magnitude of the Fourier transform of (a) the note in Figure 1.3 and (b) three notes played together.

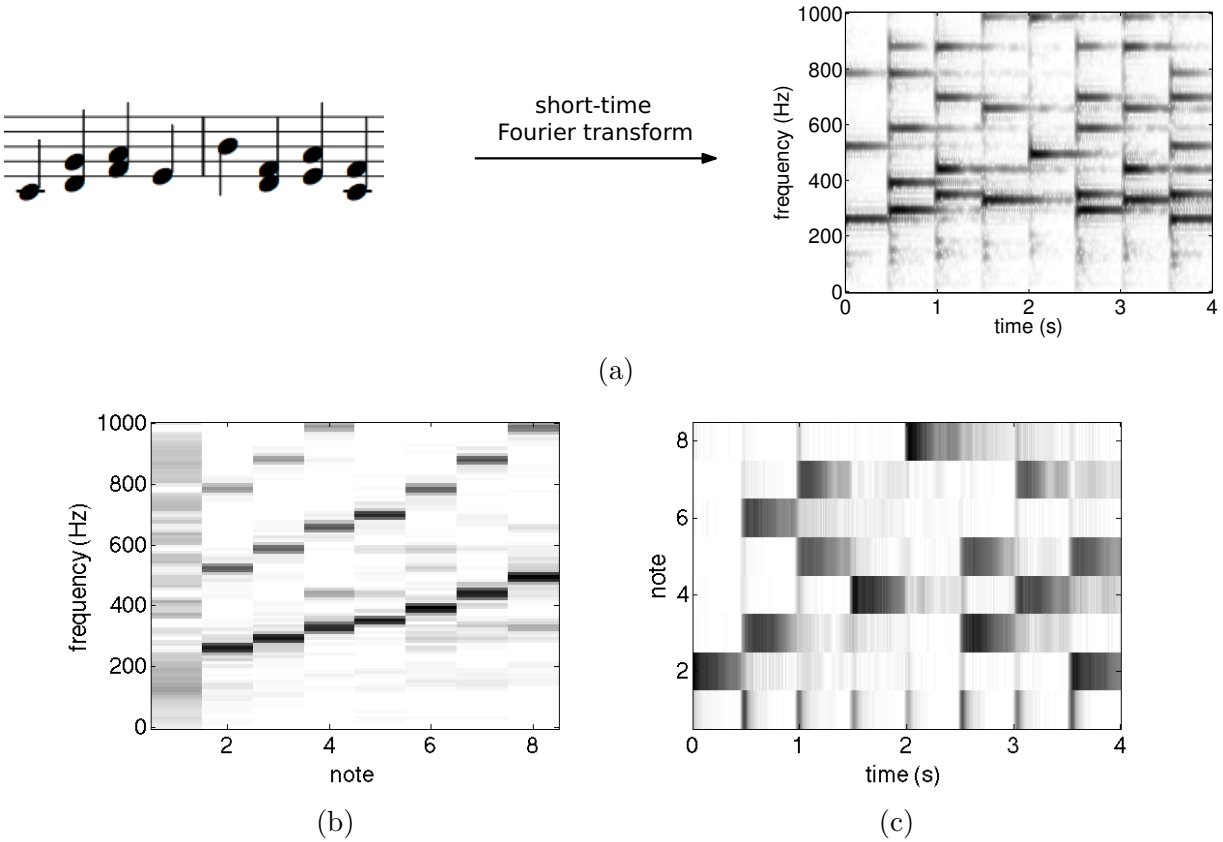


Figure 1.6: The short-time Fourier transform of an audio signal is taken to obtain the matrix X in (a). NMF produces (b) note matrix B and (c) note activation matrix G.

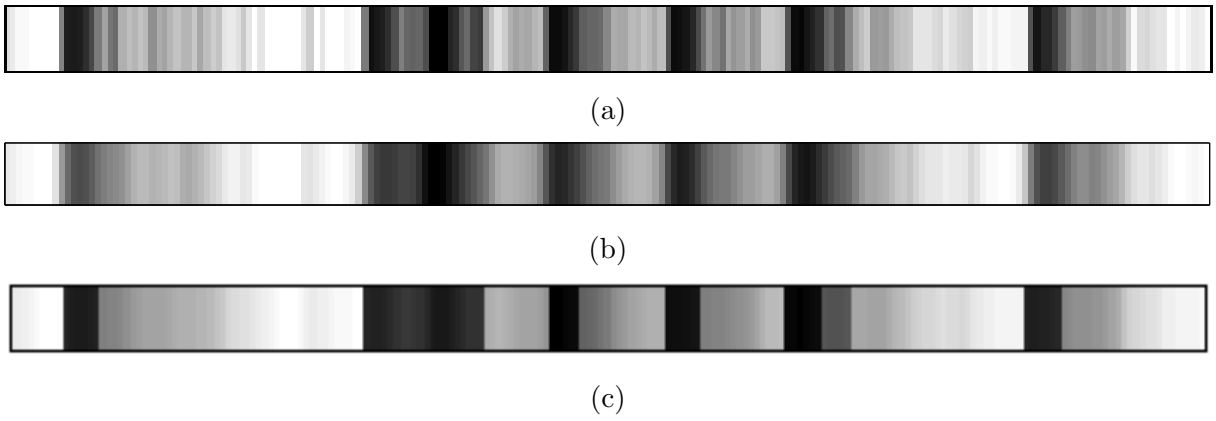


Figure 1.7: A row of G found by (a) standard NMF, (b) smoothed NMF, and (c) piecewise smooth NMF.

Chapter 2

Time-frequency Analysis

In dealing with audio we must transform our data in order to make use of NMF. Consider the audio signal in Figure 2.1a represented by a pressure time wave. From this representation it is difficult to discern what kind of source is responsible for the wave. Suppose we take the Fourier transform of the data as shown in Figure 2.1b. Here we see the structure of a musical note, that is, a strong fundamental frequency as well as overtones.

2.1 Fourier transform

The Fourier transform allows us to view spatial or time signals in terms of frequency. It is often useful, as shown in Figure 2.1, to work in the frequency domain in order to discern structure in data. The Fourier transform of a continuous signal, s, is defined as

$$S(\xi) = \int_{-\infty}^{\infty} s(t)e^{-2\pi it\xi}dt \tag{2.1}$$

We can view the Fourier transform as a change of basis from the standard basis to a basis composed of sines and cosines of various frequencies. We can recover the original signal from its transform by using the inverse Fourier transform

$$s(t) = \int_{-\infty}^{\infty} S(\xi)e^{2\pi it\xi}d\xi \tag{2.2}$$

For many applications, the signal being considered will not be continuous. Rather the signal will be a discrete time series $s_0, s_1, \ldots, s_{N-1}$. As such, we will make use of the

discrete Fourier transform defined below

$$S_k = \sum_{n=0}^{N-1} s_n e^{-2\pi ikn/N} \tag{2.3}$$

The inverse discrete Fourier transform is thus

$$s_n = \frac{1}{N} \sum_{k=0}^{N-1} S_k e^{2\pi ikn/N} \tag{2.4}$$

Though in most applications our signal will be real, the Fourier transform is complex. Instead of dealing with complex data we often only consider the magnitude the Fourier transform (as in Figure 2.1b). In doing so we unfortunately lose temporal information. Consider the two signals Figure 2.2. The first is a low frequency sine wave followed by a high frequency sine wave. The second signal is the opposite. Note how the plots of the magnitude of the Fourier transforms are identical. Ideally, we would like to work with only the magnitude of the Fourier transform, but still keep temporal information. We will explore methods for such a task in the next section.

2.2 Time-frequency representations

As noted in the previous section, considering only the magnitude of the Fourier transform, though convenient, does not allow us to use any temporal information. One solution to this problem is to take Fourier transforms of many *windows* of the data. A window is a function that localizes the signal to a short time interval. See Figure 2.3 for an example of the effects of two types of windows on a sine function. We can use windows to adapt the Fourier transform into a time-frequency representation.

2.2.1 Short-time Fourier transform

The short-time Fourier transform (also known as the windowed Fourier transform or Gabor transform) allows us to represent a signal in terms of both frequency and time. The continuous version is defined as

$$S(\tau, \xi) = \int_{-\infty}^{\infty} s(t) w(t - \tau) e^{-2\pi it\xi} dt \tag{2.5}$$

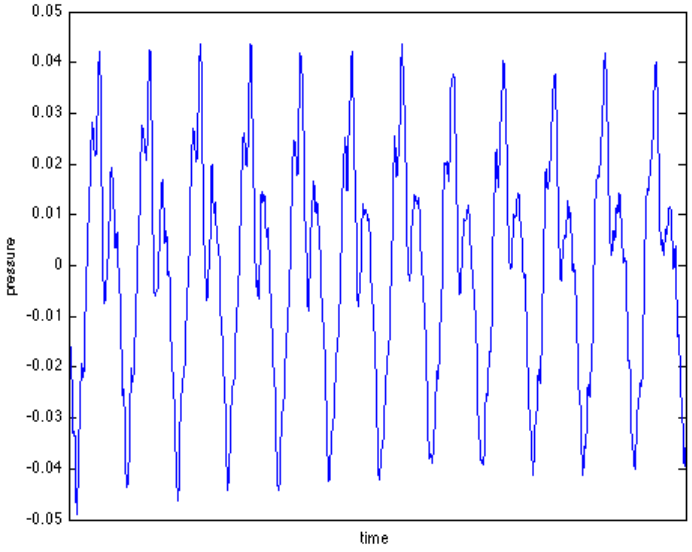
where w is the window function. As opposed to the regular Fourier transform, here S is a function of both frequency and time.

As before, we generally deal with discrete data. The discrete short-time Fourier transform is defined as

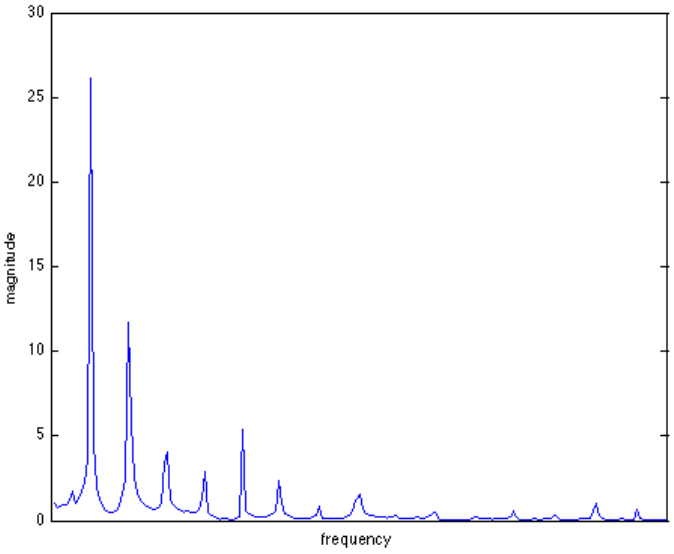
$$S_{k,\xi} = \sum_{n=-\infty}^{\infty} s_n w_{n-k} e^{-2\pi i \xi n} \tag{2.6}$$

By taking the magnitude of S we now have a nonnegative matrix representation of a signal which can be processed by NMF. Let us return to our two signals from Figure 2.2. The magnitudes of the short-time Fourier transforms of both signals are shown in Figure 2.4. We can determine from the transforms that the first signal starts with a low frequency wave and changes to a high frequency wave (and vice versa for the second signal).

Using the magnitude of the short-time Fourier transform and NMF for music transcription was proposed by Smaragdis and Brown [38]. Since then many researchers have worked to improve transcription accuracies [5]. The goal of our work is to explore the effects of constraints on NMF. We will show in Chapter 5 that our constraints allow us to achieve state of the art performance in the task of music transcription.



(a)



(b)

Figure 2.1: (a) Example of an audio signal. (b) Magnitude of the Fourier transform of the audio signal in (a).

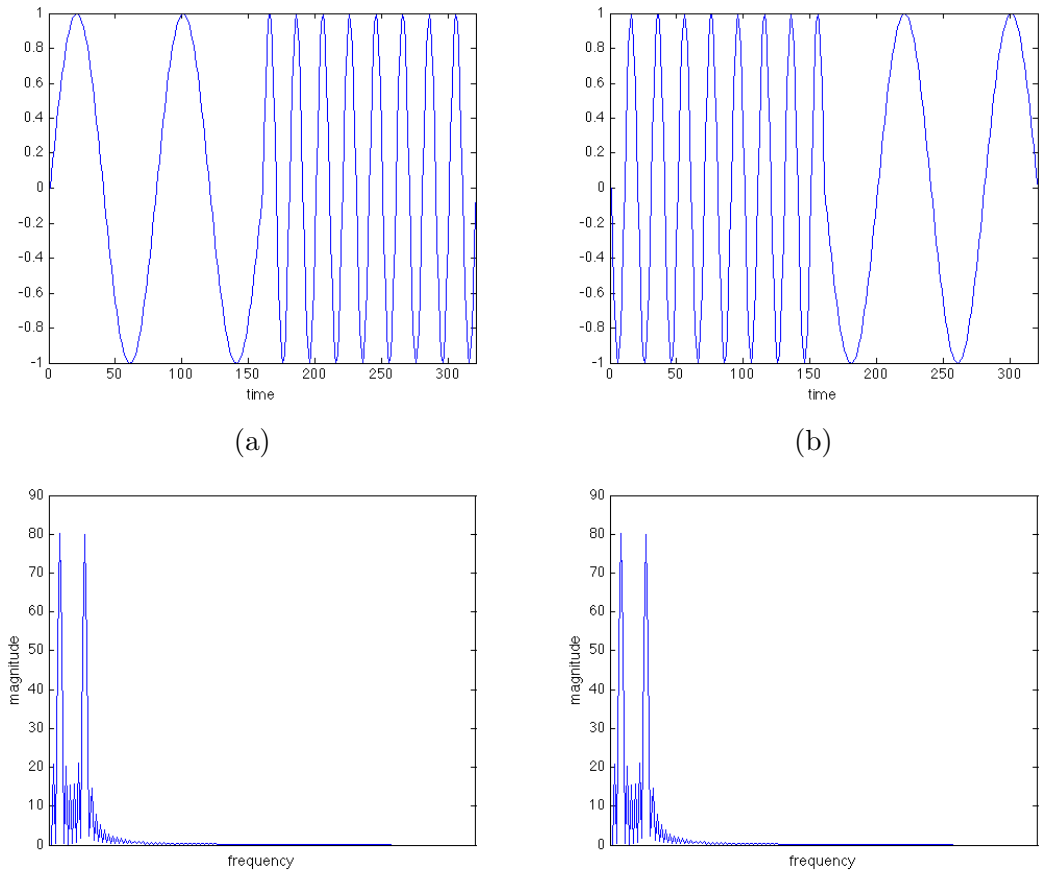
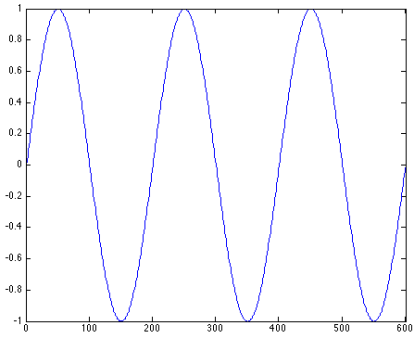
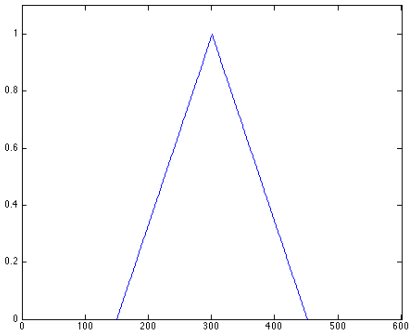
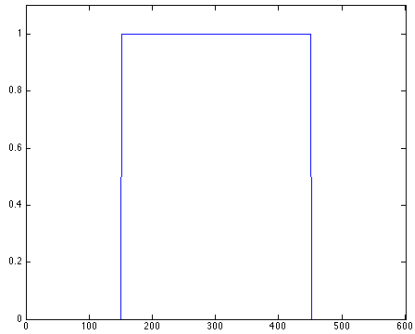


Figure 2.2: (a) A low frequency wave followed by a high frequency wave. (b) A signal opposite to (a). (c,d) Magnitudes of the Fourier transform of (a,b) respectively.

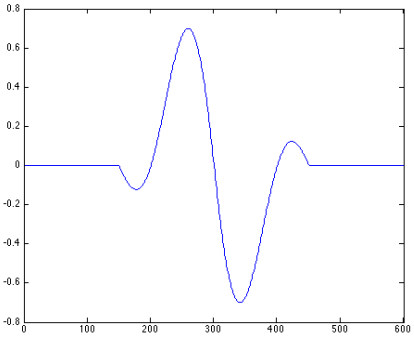


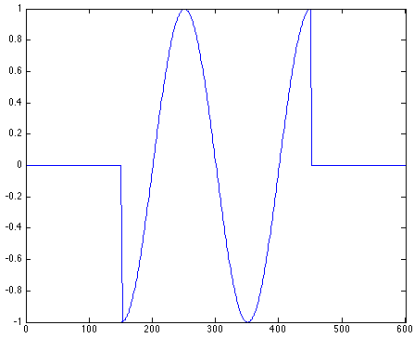
(a)





(b) (c)





(d) (e)

Figure 2.3: (a) Sine wave. (b,c) Triangular and rectangular window. (d,e) Windowed sine wave using a triangular and rectangular window respectively.

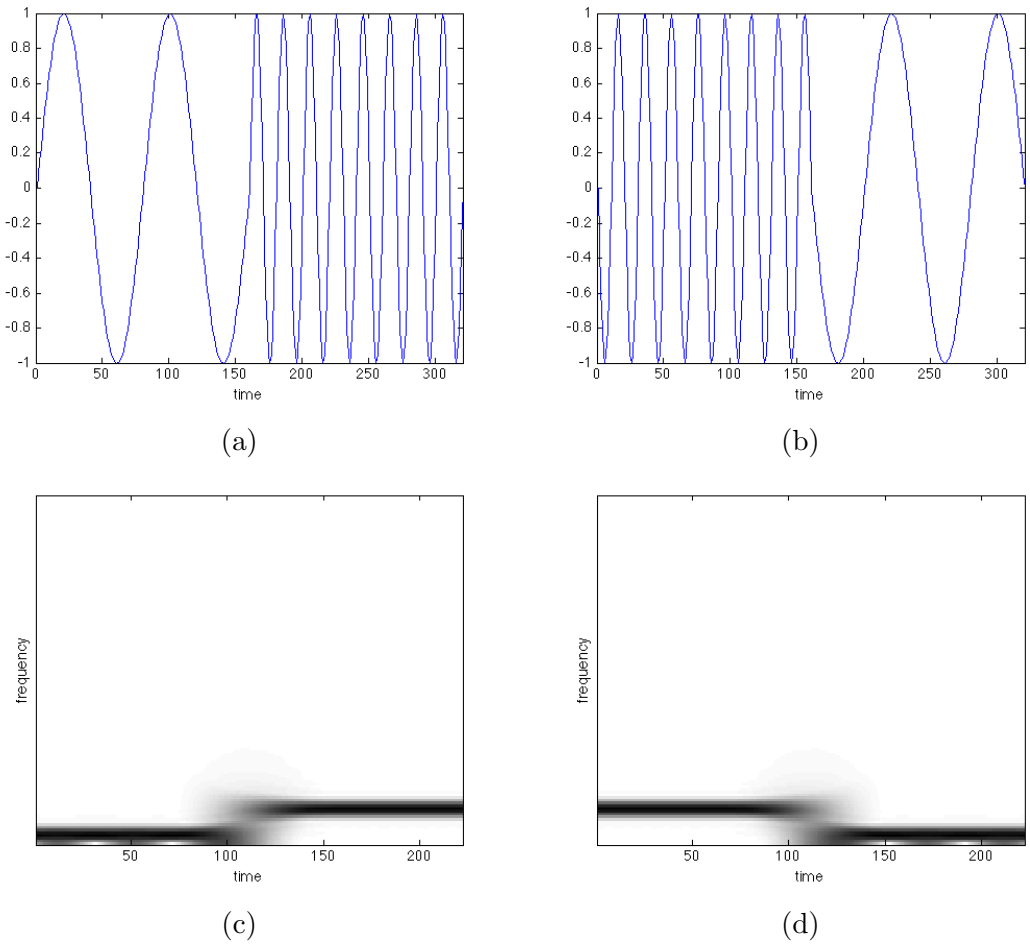


Figure 2.4: (a) A low frequency wave followed by a high frequency wave. (b) A signal opposite to (a). (c,d) Magnitudes of the short-time Fourier transform of (a,b) respectively.

Chapter 3

Nonnegative Matrix Factorization

3.1 Background

The goal of nonnegative matrix factorization (NMF) is to approximate a matrix $X \in \mathbb{R}^{d \times n}$ as a product of two matrices $B \in \mathbb{R}^{d \times r}$ and $G \in \mathbb{R}^{r \times n}$. Here, n is the number of data points, d is the dimension of each data point, and r is the number of basis vectors used to approximate the data. In other words, we wish to find matrices B and G such that

$$X \approx BG$$

where we evaluate the closeness of BG to X by some cost function.

One question the reader may have is why we choose a nonnegative model. The answer lies in the type of data we see in the real world. Consider the audio domain. Here, sounds overlap and add together. Also, we do not typically observe any subtractive circumstances (that is, sounds cancelling out). Hence a nonnegative additive model, such as NMF, seems to be appropriate.

Let us consider the task of music transcription as mentioned in the introduction. Suppose we take an audio sample of piano music and apply NMF. See Figure 1.2 for an illustration. We begin by taking the magnitude short-time Fourier transform of the signal in order to get a nonnegative data matrix. We then find the matrices B and G. The basis vectors making up the columns of B is shown in the bottom of Figure 1.2. The activation arrows represent G. The i^{th} column of G tells us what basis vectors are activated in the i^{th} column of X. In our transcription problem we only concern ourselves with whether a

basis vector is activated at a given time. However, the entries of G are real numbers and thus hold more information than just basis activation.

Other methods of matrix factorizations are not suitable for such a task. Let us repeat the experiment by Lee and Seung [29] on music data. We consider two other methods of factorization: vector quantization (VQ) and principle component analysis (PCA). VQ finds basis vectors that are prototypes for the data points in X. PCA finds basis vectors that contain both positive and negative parts.

Figure 3.1 shows the magnitude of the short time Fourier transform for random piano notes generated from midi software [1]. The music is composed of 12 notes from a single octave. At any point in time there are exactly three notes being played, and each note is played for half a second. Suppose we analyze the data using PCA, VQ, and NMF. Here we choose to break the data into 13 components (one extra component for the noise during a note onset).

Figure 3.2 shows the results of running PCA. Note how it is difficult to discern what each basis vector is representing, especially since they can contain negative values. Recall that the data is the magnitude of a short time Fourier transform and so negative values have no intuitive meaning. However, for a given data point from the data, PCA is able to reconstruct it quite well. Now let us consider VQ as in Figure 3.3. Like PCA, the basis vectors are difficult to interpret. Furthermore, the reconstructed data point has poor accuracy since only one basis vector can be activated at any one time. Finally, let us try NMF. The results are shown in Figure 3.4. Note how each basis vector corresponds to a single note (a fundamental frequency as well as overtones). Also note that for a fixed data point the corresponding column of G shows three notes activated, which is to be expected. Furthermore, the reconstructed data point is quite accurate.

This example shows some motivation for utilizing NMF instead of other methods. We will later show other applications where a parts based nonnegative model is suitable.

3.1.1 Formulation of NMF

Lee and Seung [30] propose using the following cost function[1] to measure the closeness of BG to X

$$D(X||BG) = \sum_{i=1}^{d}\sum_{j=1}^{n}\left(X_{ij}\log\frac{X_{ij}}{(BG)_{ij}} - X_{ij} + (BG)_{ij}\right) \tag{3.1}$$

[1]Note that equation 3.1 reduces to the Kullback-Leibler divergence when $\sum_{i=1}^{d}\sum_{j=1}^{n}X_{ij} = \sum_{i=1}^{d}\sum_{j=1}^{n}(BG)_{ij} = 1$. In this thesis X is scaled so that its maximum element is one.

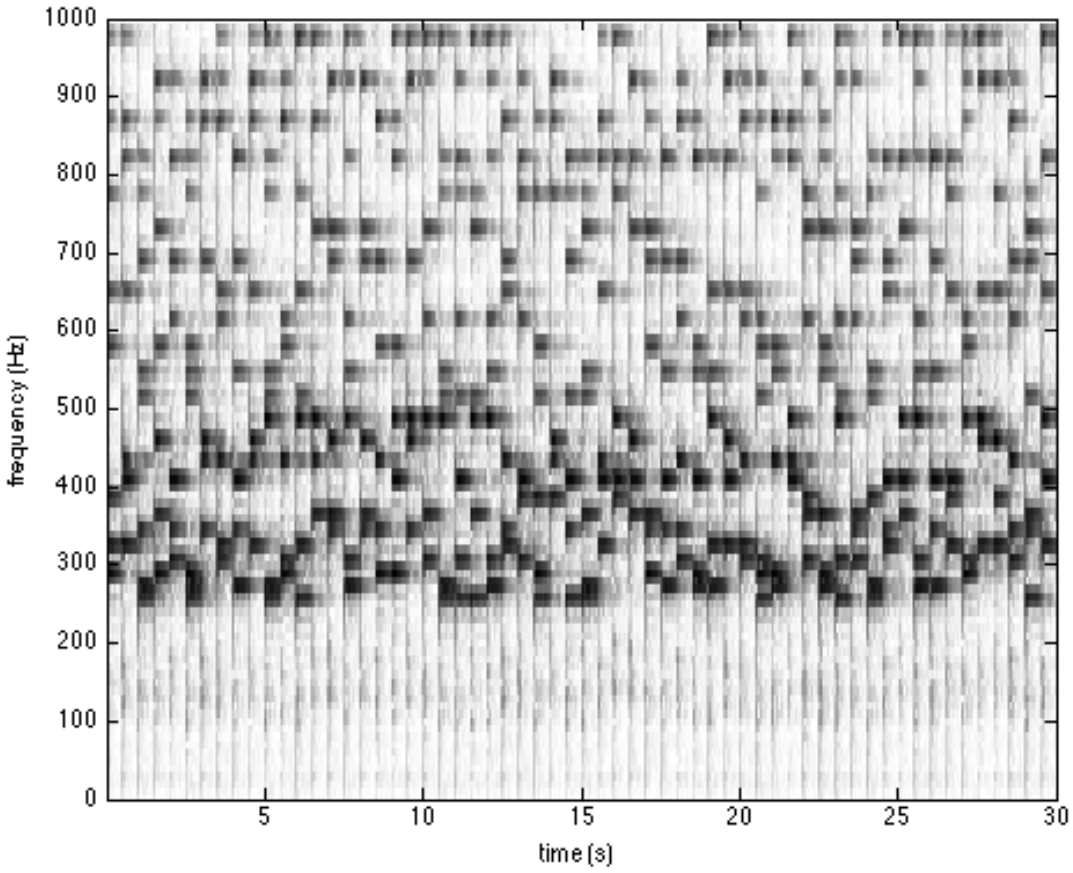


Figure 3.1: Spectrogram of random notes.

where M_{ij} refers to the element of M in the i^{th} row and j^{th} column. We will call this cost function divergence. Though it is possible to use many different cost functions to measure similarity, we choose to use only the above function since the focus of this thesis is on the constraints imposed G. The method discussed in Lee and Seung [30] is an iterative method. There are two update steps:

$$B_{ij} = B_{ij} \frac{\sum_{a=1}^{n} G_{ja} X_{ia} / (BG)_{ia}}{\sum_{b=1}^{n} G_{jb}} \tag{3.2}$$

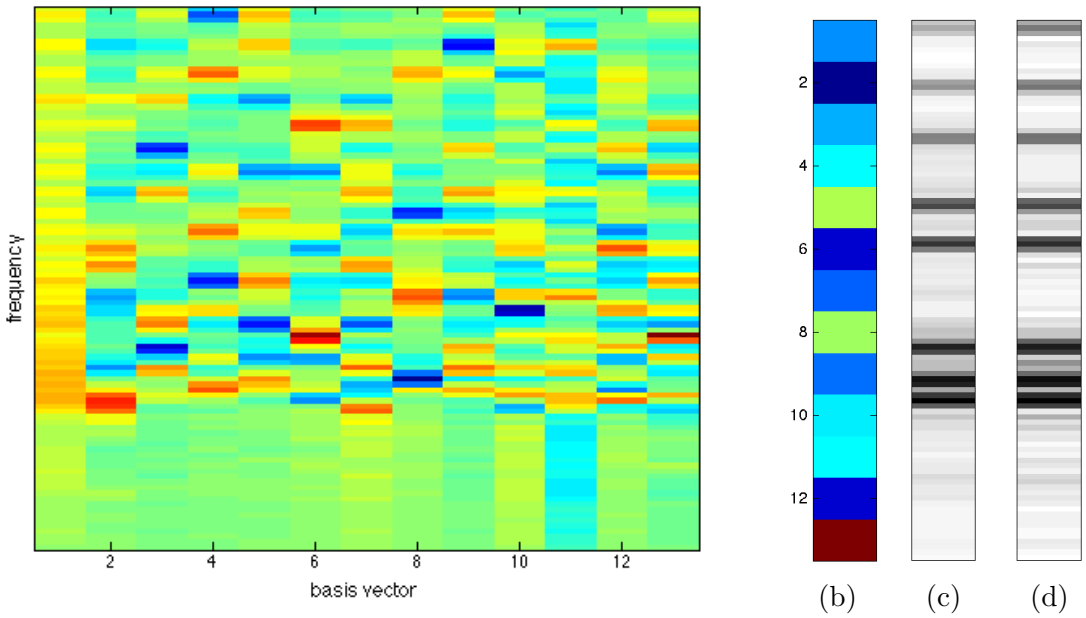


Figure 3.2: PCA method. (a) B matrix, (d) data point in original spectrogram (Figure 3.1) (b) corresponding column of G, (c) reconstructed data point. For (a-b) positive values are shades of red and negative values are shades of blue.

and

$$G_{ij} = G_{ij} \frac{\sum_{a=1}^{d} B_{ai} X_{aj} / (BG)_{aj}}{\sum_{b=1}^{d} B_{bi}} \tag{3.3}$$

Note how the update rules preserve nonnegativity since they are products of nonnegative numbers. This iterative method is similar to gradient descent, but with varying step size. Consider the following additive gradient descent update rule:

$$G_{ij} = G_{ij} + \eta_{ij} \left[\sum_{a=1}^{d} B_{ai} \frac{X_{aj}}{(BG)_{aj}} - \sum_{a=1}^{d} B_{ai} \right] \tag{3.4}$$

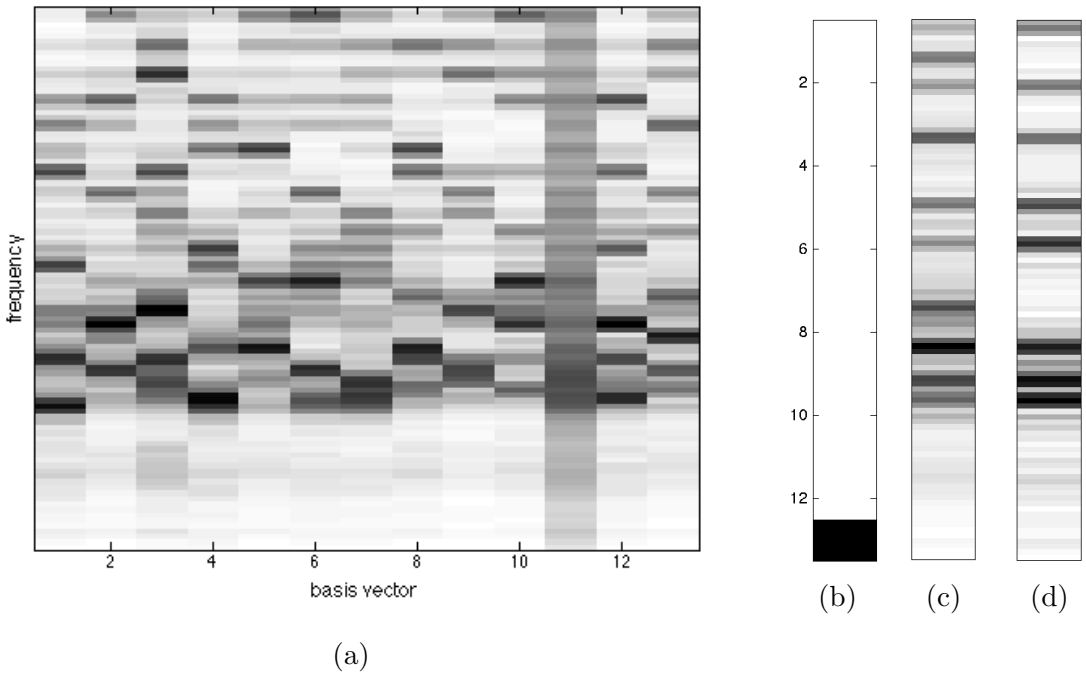


Figure 3.3: VQ method. (a) B matrix, (d) data point in original spectrogram (Figure 3.1) (b) corresponding column of G, (c) reconstructed data point.

For small η_{ij} this update rule will decrease $D(X||BG)$. If we set

$$\eta_{ij} = \frac{G_{ij}}{\sum_{a=1}^{d} B_{ai}}$$

and substitute it into equation 3.4 we obtain equation 3.3. We can obtain equation 3.2 similarly. Note, however, that η_{ij} is not guaranteed to be small and hence equations 3.2 and 3.3 may not decrease equation 3.1. However, these update rules are in fact proven to be nonincreasing [30]. The proof makes use of an auxiliary function similar to that used in the expectation-maximization algorithm [30]. There are several drawbacks to using this method. First, it is prone to getting stuck in local minima. Second, the correct choice for r, the number of basis vectors, is not clear. Lastly, the method does not exploit sparseness or temporal coherence between columns of X. We will show that our additions help to solve these limitations.

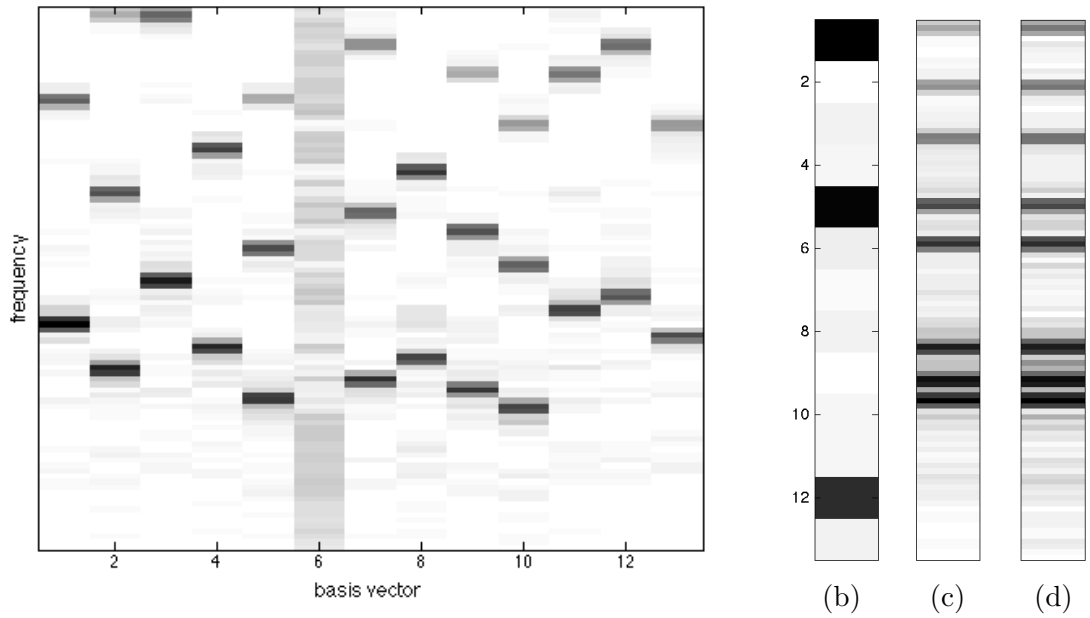


Figure 3.4: NMF method. (a) B matrix, (d) data point in original spectrogram (Figure 3.1) (b) corresponding column of G, (c) reconstructed data point.

3.2 Constrained NMF

Let $F(X, BG) = D(X||BG)$, and define

$$\nabla_M F \triangleq \begin{pmatrix} \frac{\partial F}{\partial M_{11}} & \frac{\partial F}{\partial M_{12}} & \cdots & \frac{\partial F}{\partial M_{1j}} \\ \frac{\partial F}{\partial M_{21}} & \frac{\partial F}{\partial M_{22}} & \cdots & \frac{\partial F}{\partial M_{2j}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F}{\partial M_{i1}} & \frac{\partial F}{\partial M_{i2}} & \cdots & \frac{\partial F}{\partial M_{ij}} \end{pmatrix} \tag{3.5}$$

for some $i \times j$ matrix M. Let $\nabla_M^+ F(X, BG)$ and $\nabla_M^- F(X, BG)$ be the absolute values of the positive and negative parts of $\nabla_M F(X, BG)$ respectively. We can show [45] that the

previous update rules can be written as

$$B_{ij} = B_{ij} \frac{[\nabla_B^- F(X, BG)]_{ij}}{[\nabla_B^+ F(X, BG)]_{ij}} \tag{3.6}$$

and

$$G_{ij} = G_{ij} \frac{[\nabla_G^- F(X, BG)]_{ij}}{[\nabla_G^+ F(X, BG)]_{ij}} \tag{3.7}$$

This form of update rules are often observed to be nonincreasing in practice (though not in general) [8]. It also allows us to easily derive new equations if we modify our cost function.

In our work we only impose constraints on G. Recall that B consists of the underlying basis vectors which compose our data samples. Any constraints on B would constrain the type of data that we can model. Hence, we do not constrain B.

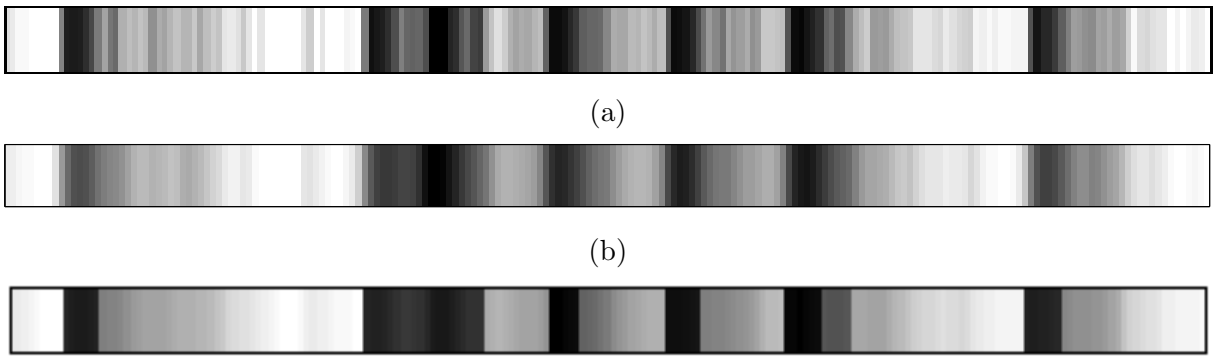
3.2.1 Temporal coherence

In many applications the columns of X are not independent. Take, for example, piano music. In this application X represents time series data. The matrices B and G will represent the notes and note activations respectively. See Figure 1.6 for an example. One way to exploit this temporal coherence is to impose that the rows of G be smooth. This smoothness constraint has been studied in the literature. One such method modifies the cost function to

$$D(X||BG) + \lambda \sum_{i=1}^{r} \sum_{j=2}^{n} (G_{ij} - G_{ij-1})^2 \tag{3.8}$$

where λ controls the weight of the constraint. This constraint attempts to minimize the squared differences between neighbouring elements of G. However, imposing such a smoothness constraint over an entire row of G means that there can be no breakpoints (i.e. sharp changes in the rows of G). We instead propose to constrain the rows of G to be piecewise smooth. Our motivation for such a constraint is inspired from music data where a piecewise smooth model is quite evident when looking at data such as in Figure 3.1. However, we do not limit the scope of this work to just music data.

Consider a row of G found by standard NMF on a music sample as in Figure 3.5a. Imposing smoothness over the entire row will result in Figure 3.5b. This smoothness makes the onset and offset of the notes unclear. If instead we impose our piecewise smoothness constraint (as in Figure 3.5c) we see the onsets and offsets more clearly. Clear onsets and offsets are important for tasks such as music transcription.



(a)

(b)

(c)

Figure 3.5: A row of G found by (a) standard NMF, (b) smoothed NMF, and (c) piecewise smooth NMF.

In order to impose piecewise smoothness, we modify our cost function to be

$$D(X||BG) + \lambda \sum_{i=1}^{r} \sum_{j=2}^{n} \left(1 - e^{-(G_{ij} - G_{ij-1})^2/2\sigma^2}\right) \tag{3.9}$$

See Figure 3.6 for plots of the function $1 - e^{-x^2/\sigma^2}$. Note how small differences between elements of G are treated like quadratic differences, and large differences have an approximately fixed cost of λ. Consider the distribution of differences between neighbouring elements of G. We can view large differences as outliers of the distribution. In this sense our method of assigning a fixed cost to large difference is similar to outlier detection in robust statistics [11].

We have introduced two new parameters to the model, λ and σ. The reader may wonder if doing so will greatly complicate our optimization task, since now we must not only optimize a non-convex cost function but also find suitable parameter values. However, we will show that by fixing λ and iteratively optimizing the cost function for varying values of σ we can achieve good results without much extra work. This method has similarities with the graduated non-convexity algorithm pioneered by Blake and Zisserman in which a non-convex problem is solved by optimizing a sequence simpler functions [9].

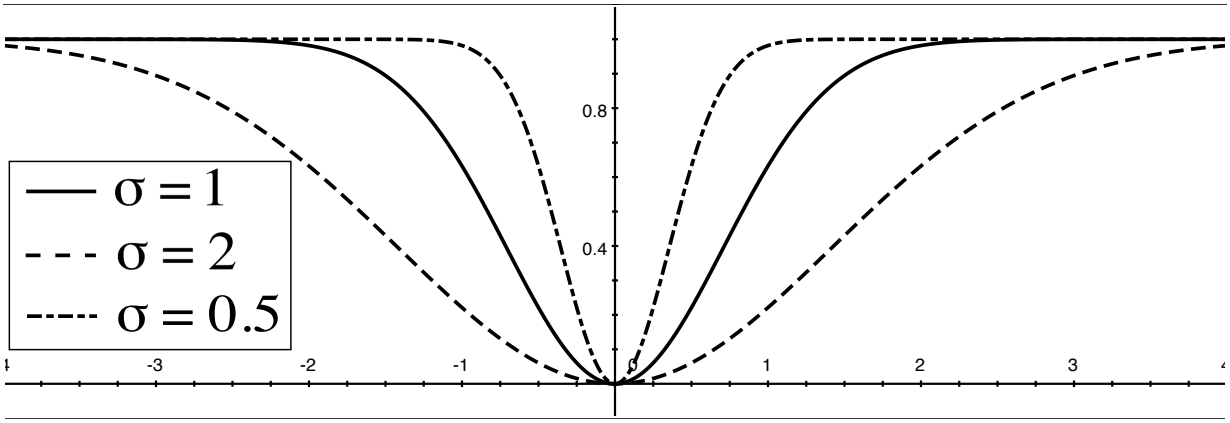


Figure 3.6: Plots of the function $1 - e^{-x^2/\sigma^2}$ for varying σ values.

This new constraint does not involve B, so its update rule stays the same. We now find the gradient of the cost function with respect to G.

$$\nabla_G D(X||BG) = B^\intercal \left(\mathbf{1} - \frac{X}{BG}\right) \tag{3.10}$$

where $\mathbf{1}$ is the all ones matrix the same size as X.

$$\left[\nabla_G \lambda \sum_{i=1}^{r}\sum_{j=2}^{n}\left(1 - e^{-(G_{ij}-G_{ij-1})^2/2\sigma^2}\right)\right]_{ij} =$$

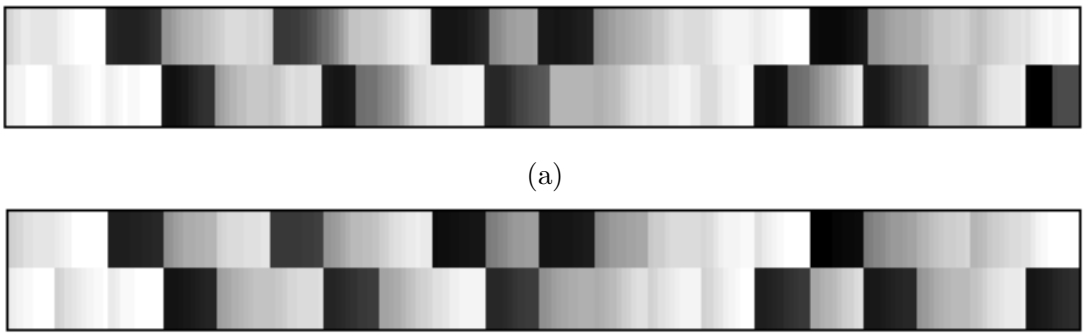
$$\frac{\lambda}{\sigma^2}\left(e^{-(G_{ij}-G_{ij-1})^2/2\sigma^2}(G_{ij} - G_{ij-1}) - e^{-(G_{ij+1}-G_{ij})^2/2\sigma^2}(G_{ij+1} - G_{ij})\right) \tag{3.11}$$

The update rule becomes

$$G_{ij} = G_{ij}\frac{\left[B^\intercal \frac{X}{BG}\right]_{ij} + \frac{\lambda}{\sigma^2}\left(G_{ij-1}\cdot e^{-(G_{ij}-G_{ij-1})^2/2\sigma^2} + G_{ij+1}\cdot e^{-(G_{ij+1}-G_{ij})^2/2\sigma^2}\right)}{[B^\intercal \mathbf{1}]_{ij} + \frac{\lambda}{\sigma^2}G_{ij}\left(e^{-(G_{ij}-G_{ij-1})^2/2\sigma^2} + e^{-(G_{ij+1}-G_{ij})^2/2\sigma^2}\right)} \tag{3.12}$$

A similar update rule was derived by Jia and Qian [23] for the purpose of hyper spectral unmixing. See their work for a proof of convergence.

In many signals (such as music and speech) breakpoints are correlated. For example, consider a chord played on a piano. Each note is struck simultaneously, hence the breakpoints corresponding to the attacks of the notes should be aligned. Consider the two



(a)

(b)

Figure 3.7: Two rows of G found by imposing (a) piecewise smoothness and (b) piecewise smoothness with aligned breakpoints.

rows of G in Figure 3.7a. By favouring aligned breakpoints, we can improve breakpoint detection as seen in Figure 3.7b.

In order to align breakpoints, we modify our cost function to be

$$\sum_{i=1}^{d}\sum_{j=1}^{n}(X_{ij} - BG_{ij})^2 + \lambda\sum_{j=2}^{n}\left(1 - e^{-\sum_{i=1}^{r}(G_{ij}-G_{ij-1})^2/2\sigma^2}\right) \tag{3.13}$$

As in our last cost function, a single breakpoint is assigned a fixed cost of λ. However, if multiple breakpoints occur at the same time step, then they are assigned a total cost of λ. Hence, the cost function tends to prefer aligned breakpoints. The gradient of the new term with respect to G is

$$\left[\nabla_G\lambda\sum_{j=2}^{n}\left(1 - e^{-\sum_{i=1}^{r}(G_{ij}-G_{ij-1})^2/2\sigma^2}\right)\right]_{ij} =$$

$$\frac{\lambda}{\sigma^2}\left(e^{-\sum_{i=1}^{r}(G_{ij}-G_{ij-1})^2/2\sigma^2}(G_{ij} - G_{ij-1}) - e^{-\sum_{i=1}^{r}(G_{ij+1}-G_{ij})^2/2\sigma^2}(G_{ij+1} - G_{ij})\right) \tag{3.14}$$

The new update rule for G is now

$$G_{ij} = G_{ij}\frac{\left[B^\intercal\frac{X}{BG}\right]_{ij} + \frac{\lambda}{\sigma^2}\left(G_{ij-1}\cdot e^{-\sum_{i=1}^{r}(G_{ij}-G_{ij-1})^2/2\sigma^2} + G_{ij+1}\cdot e^{-\sum_{i=1}^{r}(G_{ij+1}-G_{ij})^2/2\sigma^2}\right)}{[B^\intercal\mathbf{1}]_{ij} + \frac{\lambda}{\sigma^2}G_{ij}\left(e^{-\sum_{i=1}^{r}(G_{ij}-G_{ij-1})^2/2\sigma^2} + e^{-\sum_{i=1}^{r}(G_{ij+1}-G_{ij})^2/2\sigma^2}\right)} \tag{3.15}$$

3.2.2 Sparsity

We will consider two different sparsity constraints on G. The first constraint favours small elements of G to be very close to zero. We can impose this constraint with the following cost term:

$$\lambda \sum_{i=1}^{d} \sum_{j=1}^{n} (1 - e^{-G_{i,j}^2/2\sigma^2}) \tag{3.16}$$

with the gradient with respect to G being

$$\left[\nabla_G \lambda \sum_{i=1}^{d} \sum_{j=1}^{n} (1 - e^{-G_{i,j}^2/2\sigma^2}) \right]_{ij} = \frac{\lambda}{\sigma^2} G_{ij} \, e^{-G_{ij}^2/2\sigma^2} \tag{3.17}$$

The update rule for G is:

$$G_{ij} = G_{ij} \frac{\left[B^\intercal \frac{X}{BG} \right]_{ij}}{\left[B^\intercal \mathbf{1} \right]_{ij} + \frac{\lambda}{\sigma^2} G_{ij} \, e^{-G_{ij}^2/2\sigma^2}} \tag{3.18}$$

The second sparsity constraint favours few nonzero rows of G, which can aid in finding a suitable value for r. The choice of r can greatly influence the results of NMF. Choosing r too small may force the algorithm to combine basis vectors. While choosing r too large may segment correct basis vectors into separate pieces. Consider the following cost term:

$$\lambda \sum_{i=1}^{r} (1 - e^{-(\sum_{j=1}^{n} G_{i,j})^2/2\sigma^2}) \tag{3.19}$$

The idea of this term is to assign a cost of λ to any nonzero row of G. The gradient with respect to G is

$$\left[\nabla_G \lambda \sum_{i=1}^{d} (1 - e^{-(\sum_{j=1}^{n} G_{i,j})^2/2\sigma^2}) \right]_{ij} = \frac{\lambda}{\sigma^2} e^{-(\sum_{j=1}^{n} G_{i,j})^2/2\sigma^2} \sum_{j=1}^{n} G_{ij} \tag{3.20}$$

The update rule for G is:

$$G_{ij} = G_{ij} \frac{\left[B^\intercal \frac{X}{BG} \right]_{ij}}{\left[B^\intercal \mathbf{1} \right]_{ij} + \frac{\lambda}{\sigma^2} e^{-(\sum_{j=1}^{n} G_{i,j})^2/2\sigma^2} \sum_{j=1}^{n} G_{ij}} \tag{3.21}$$

3.3 Choosing λ and σ

In regular NMF we only need to set r, the number of basis vectors. Our method introduces two new parameters per constraint: λ, the weight of the constraint, and σ, which controls the width of the Gaussian constraint function. Choosing correct value for these parameters may seem difficult at first. However, we will show that an approximate estimation of σ is good enough in practice. We will also show that the algorithm is fairly robust to changes in λ. That is, the exact value of λ does not matter so long as it is within a certain range.

We propose estimating σ for each constraint from G using the following:

- piecewise smooth: $\lambda \sum_{i=1}^{r} \sum_{j=2}^{n} \left(1 - e^{-(G_{ij} - G_{ij-1})^2 / 2\sigma^2}\right)$

$$\hat{\sigma} = std(\{G_{ij} - G_{ij-1} \mid 1 \leq i \leq d, 1 \leq j \leq n\}) \tag{3.22}$$

- aligned breakpoints: $\lambda \sum_{j=2}^{n} \left(1 - e^{-\sum_{i=1}^{r} (G_{ij} - G_{ij-1})^2 / 2\sigma^2}\right)$

$$\hat{\sigma} = std\left(\left\{\sum_{j=2}^{n} (G_{ij} - G_{ij-1}) \mid 1 \leq i \leq d\right\}\right) \tag{3.23}$$

- element-wise sparsity: $\lambda \sum_{i=1}^{d} \sum_{j=1}^{n} (1 - e^{-G_{i,j}^2 / 2\sigma^2})$

$$\hat{\sigma} = std(\{G_{ij} \mid 1 \leq i \leq d, 1 \leq j \leq n\}) \tag{3.24}$$

- row-wise sparsity: $\lambda \sum_{i=1}^{r} (1 - e^{-(\sum_{j=1}^{n} G_{i,j})^2 / 2\sigma^2})$

$$\hat{\sigma} = std\left(\left\{\sum_{j=1}^{n} G_{ij} \mid 1 \leq i \leq d\right\}\right) \tag{3.25}$$

where $std(S)$ is the standard deviation of the elements of the set S.

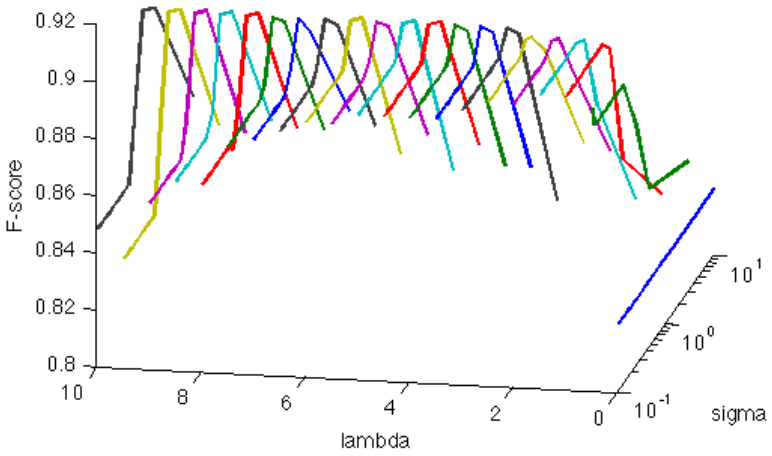
Once we have an estimate for σ, we choose a σ' which is much larger than it, say 10σ. We iterate NMF using σ' until the change to the cost function is small. We then decrease σ' and optimize again. We repeat this until σ' is relatively small, say $\sigma' = \sigma/10$. Choosing a relatively large σ' means our cost function curve is very wide, see Figure 3.6. A wide function means that only very large values to the function will be treated as outliers. As we

decrease σ', the function decreases in width and smaller values are treated as outliers. We propose that this process of decreasing σ' means that an exact choice of σ is not important. Hence, estimating σ as shown above is sufficient to get good results in practice.

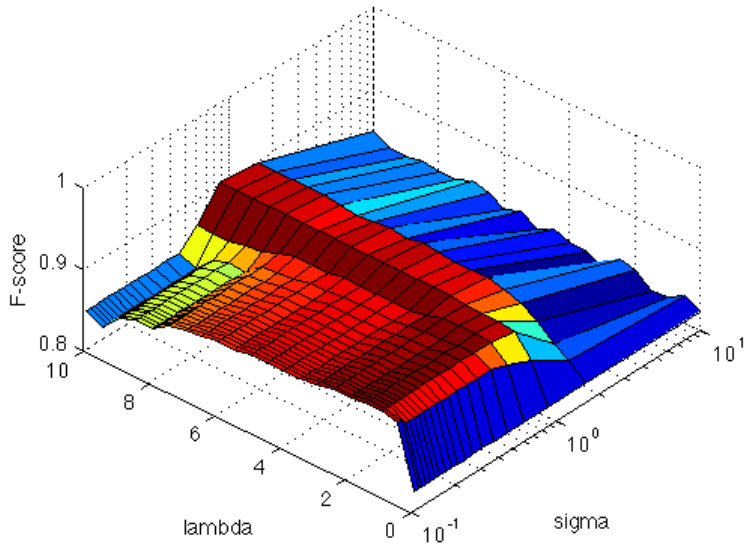
We will illustrate this point with an experiment. We plot the music transcription performance (F-score) on a piece of randomly generated synthetic piano music as we vary λ and σ. We discuss our method of music transcription in Chapter 5. In this experiment we apply the piece wise smoothness constraint. In our first test we let λ range from 0 to 20 and σ range from 10 to 0.1. In our second test we hold $\lambda = 1$ constant. Again, we let σ range to 0.1, but we run several trials with varying initial values. See Figure 3.8 and 3.9 for the results.

Let us consider the results of the first test. Each line in Figure 3.8a represents a fixed value of λ. We have included a surface plot of the data in Figure 3.8 in order to help with visualization. Observe that the F-score for $\lambda = 0$ is constant as is to be expected since σ has no effect. Also, the transcription performance with $\lambda > 0$ does not improve over the base case of $\lambda = 0$ for large values of σ. However, once σ gets sufficiently small performance improves. For small λ the F-score remains more or less constant as σ is decreased. However, as λ increases, decreasing σ negatively affects the F-score.

In Figure 3.9 we hold $\lambda = 1$ constant and vary the starting values of σ. As can be seen, starting from a large value of σ reduces the likelihood that the algorithm will get stuck in a relatively poor local minimum. Hence, we conclude that in general we should start with a very large value of σ and minimize our cost function for smaller and smaller values of σ. Furthermore, the exact values of σ at each iteration is not important so long as λ is chosen appropriately.



(a)



(b)

Figure 3.8: F-score (higher is better) for varying λ and σ values. (a) line plots and (b) surface plot.

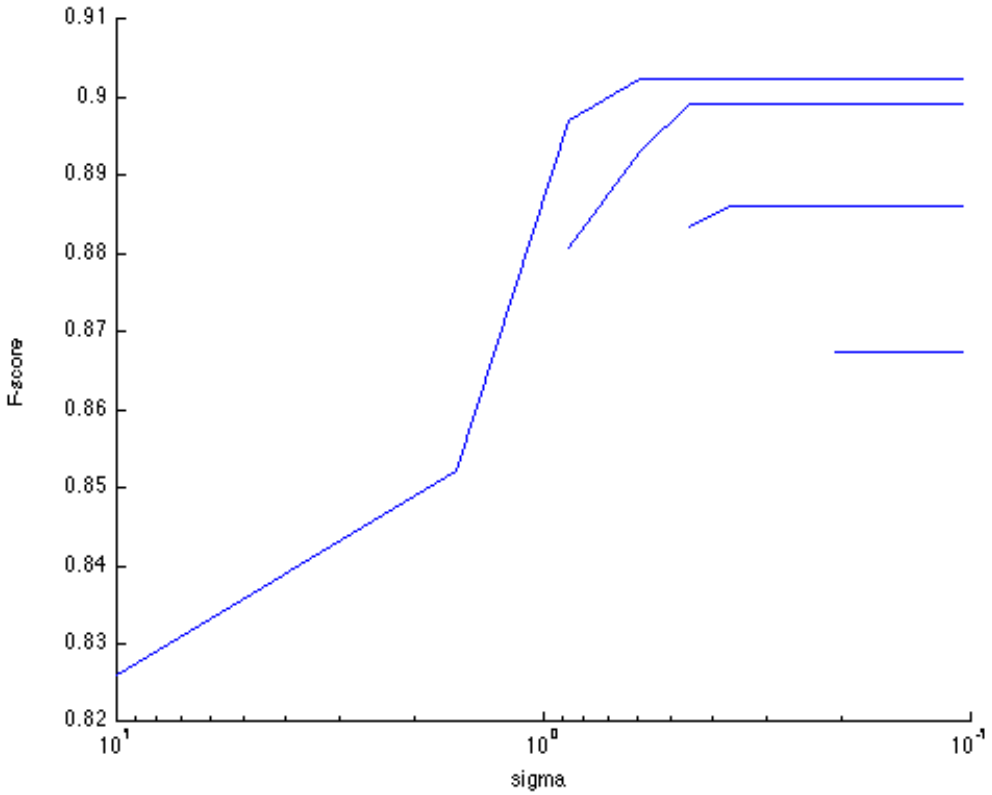


Figure 3.9: F-score (higher is better) for varying initial σ values.

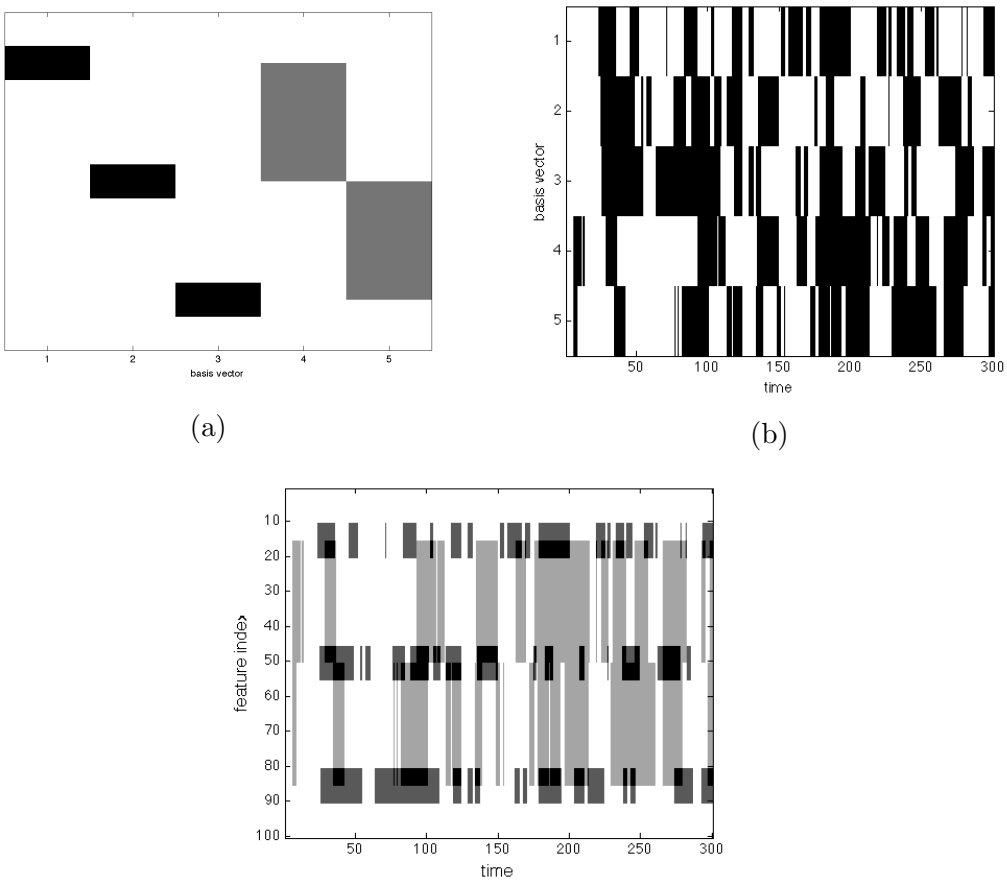
Chapter 4

Synthetic Experiments

4.1 Temporal coherence

In the next experiment we will test the effects of the piecewise smoothness constraint on synthetic data. We begin with five basis vectors (normalized to have unit sum) as in Figure 4.1a. These vectors represent generic data and so we label their dimension as 'feature index'. In the case of music transcription, the feature index corresponds to frequency. To generate our data we use a Markov process. Each basis vector is either in an 'active' or 'inactive' state. The probability of a basis vector staying in its current state is 0.9 and its probability of swapping states is 0.1. Using this Markov process we generate a G matrix as in Figure 4.1b and use it to create the data matrix in Figure 4.1c. We run NMF with varying λ values with different amounts of white Gaussian noise added to the data. See Figure 4.2 for an example of the noisy data. The signal to noise ratios (SNR) were varied from 50 dB to 20 dB. See Figures 4.3 and 4.4 for the results. The plots show the error (squared Frobenius norm) of $G - \hat{G}$, where $\hat{G}$ is our estimated G matrix.

We see that for very little noise (SNR = 50 dB) the constraint does little to improve the error. However, once the noise becomes high enough (SNR $\leq$ 40 dB) using the piecewise smooth constraint improves error. Also of note is that using too large a λ value can lead to poor results as is seen for SNR $\leq$ 30 dB.



(a) (b) (c)

Figure 4.1: Synthetic rectangle data. (a) five basis vectors (B matrix), (b) generated G matrix, (c) synthetic data created from BG.

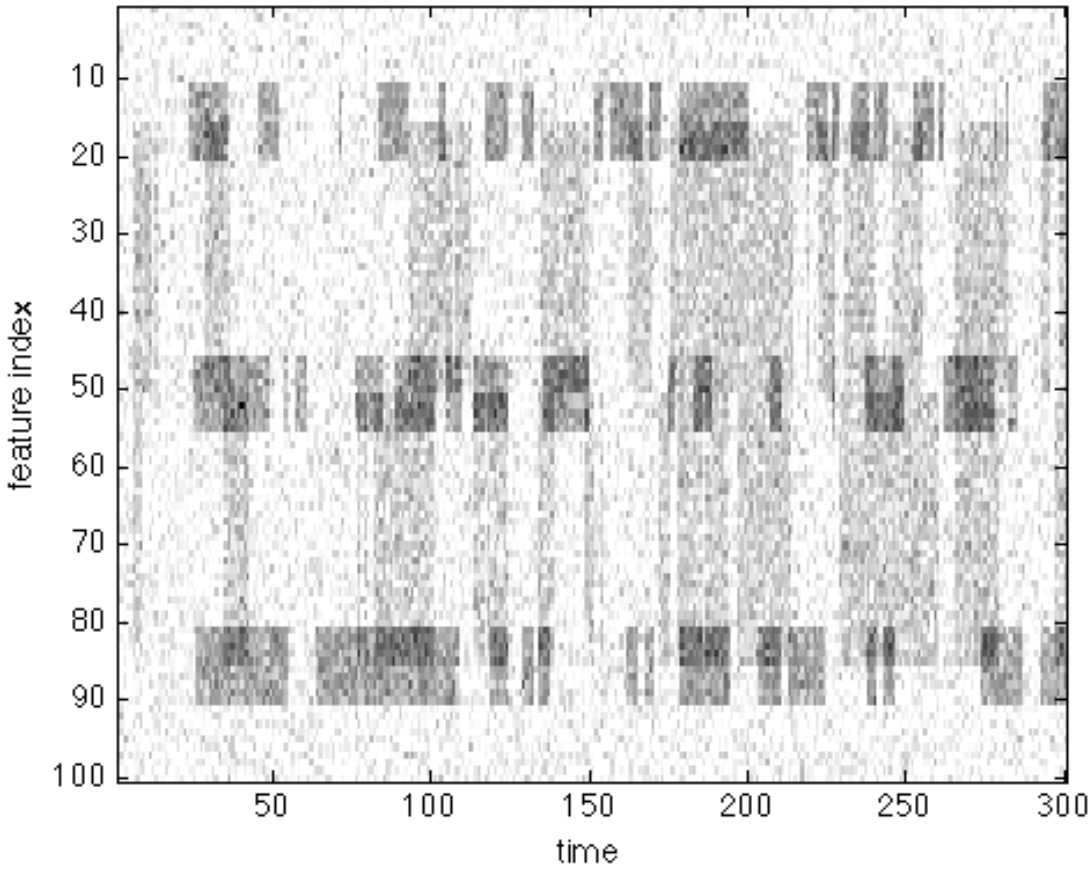


Figure 4.2: White Gaussian noise was added to data from Figure 4.1 with signal to noise ratio 20 dB.

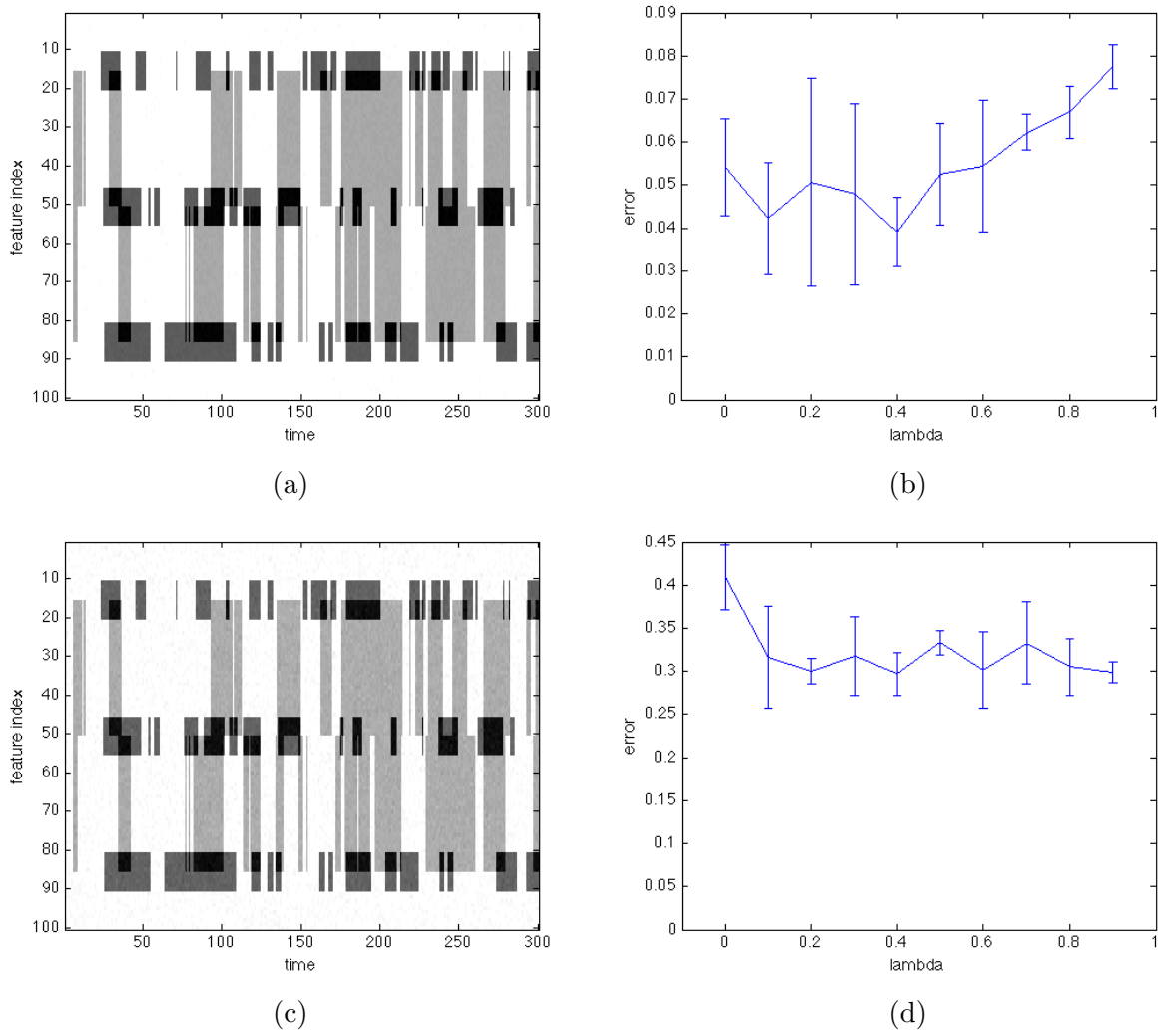


Figure 4.3: White Gaussian noise was added to data with signal to noise ratio (a) 50 dB and (c) 40 dB. Error rates with 95% confidence intervals averaged over ten trials versus λ values are shown in (b) and (d).

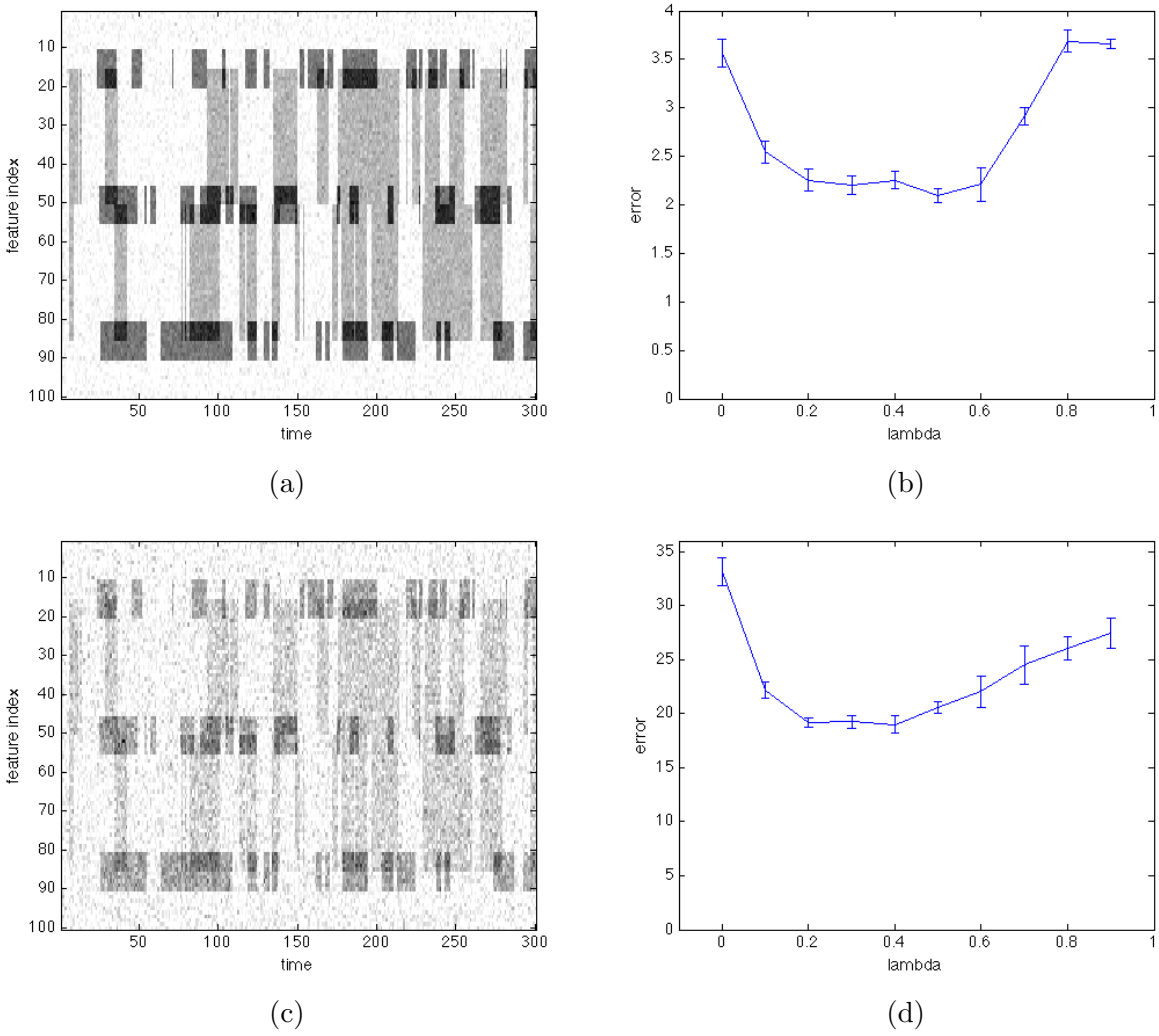


Figure 4.4: White Gaussian noise was added to data with signal to noise ratio (a) 30 dB and (c) 20 dB. Error rates with 95% confidence intervals averaged over ten trials versus λ values are shown in (b) and (d).

4.2 Aligned breakpoints

Our next experiment tests the effects of aligning breakpoints. We use ten basis vectors as shown in Figure 4.5a. We modify our generation process from our previous experiments to favour aligned breakpoints. That is, a basis has a greater chance of being activated if another basis vector is activated at the same time. The same is true for deactivation. We use this process to generate G. We then set $X = BG'$ where G' has every third row shifted by two time steps. Figure 4.5 shows the generated data. We run NMF with our piecewise smooth constraint with aligned breakpoints. We do not learn B in these experiments (i.e. we set it equal to the ground truth) in order to speed convergence to a good minimum. We measure error as the squared Frobenius norm of $G - \hat{G}$, where $\hat{G}$ is our estimated G matrix. The results are shown in Figures 4.6 and 4.7.

As we see from the results, the piecewise smoothness constraint with aligned breakpoints not only is robust to noise, but also improves the error for the case of little noise. This is the case because it is aligning the rows of G that were shifted prior to the generation of X. Though not included in the figures, NMF with the piecewise smoothness constraint (no aligned breakpoints) was also run on the data. However, the error of all $\lambda > 0$ was greater than the error for $\lambda = 0$.

4.3 Sparsity

The swimmer dataset [17] is composed of images of a synthetic swimmer with varying positions of limbs. The swimmer is composed of a constant torso and four limbs. Each limb can be in one of four possible positions. Hence there are $4^4 = 256$ possible swimmers. See Figure 4.8 for example images.

Suppose we wish to apply non-constrained NMF to the dataset. We set $r = 16$ and see the piecewise breakdown in Figure 4.9. Note how the swimmer's body is incorporated into each piece. The reader may wonder why we did not set $r = 17$, since there are 16 arm variations and one body position. Though it is possible for NMF to pick out the swimmer's body individually, we can see that the data can be represented by 16 pieces. Hence, setting $r = 17$ can result in NMF finding and extraneous 17th piece. See Figure 4.10.

Now let us suppose we do not know the underlying model of the swimmer dataset. That is, we do not know the swimmer has four limbs each with four positions. How do we go about choosing r? We can make use of our row sparseness constraint in order to make a guess at the correct r. For this experiment we will initially set $r = 50$ and $\lambda = 10$. After

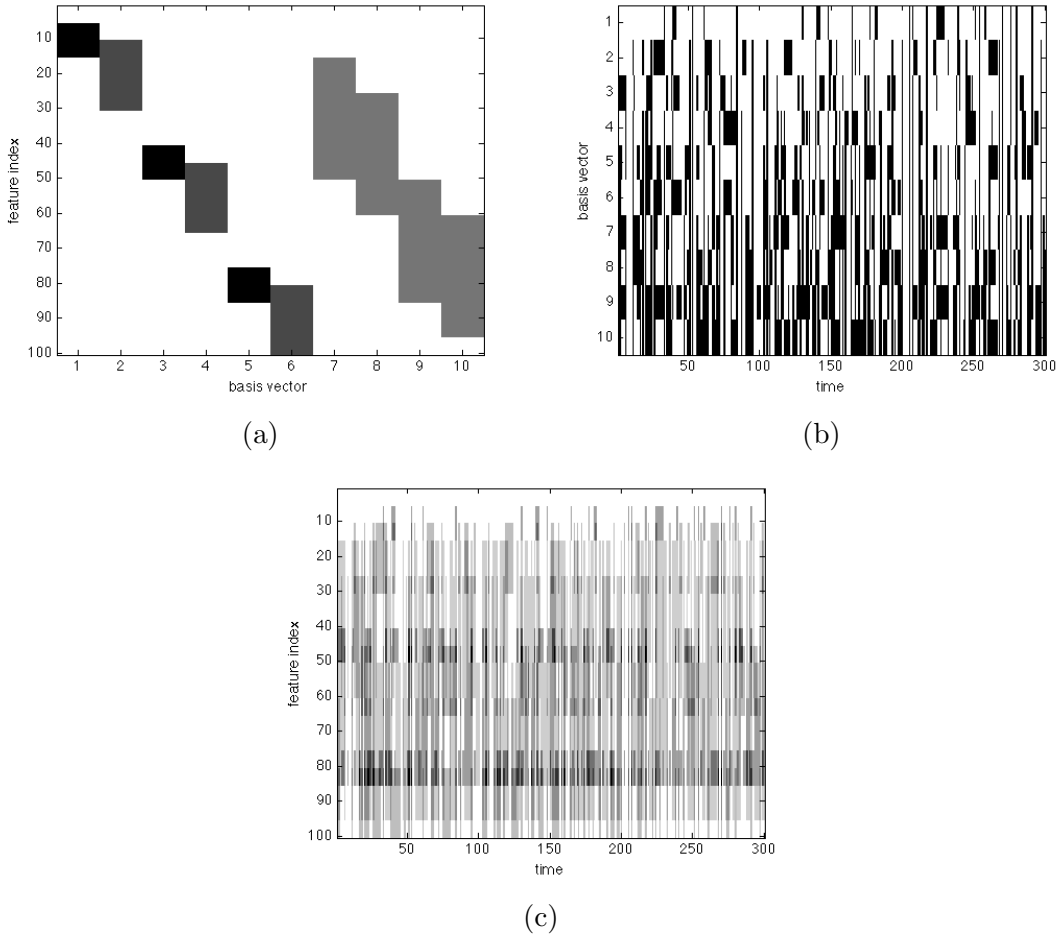


Figure 4.5: Synthetic rectangle data. (a) ten basis vectors (B matrix), (b) generated G matrix, (c) synthetic data created from BG.

we run row-wise sparse NMF, we keep only the basis vectors corresponding to the nonzero rows of G. See Figure 4.11 for the results. Note how the algorithm successfully finds the 16 pieces of the swimmer. As opposed to the non-constrained NMF case, we see the body piece attached only to the top left limb.

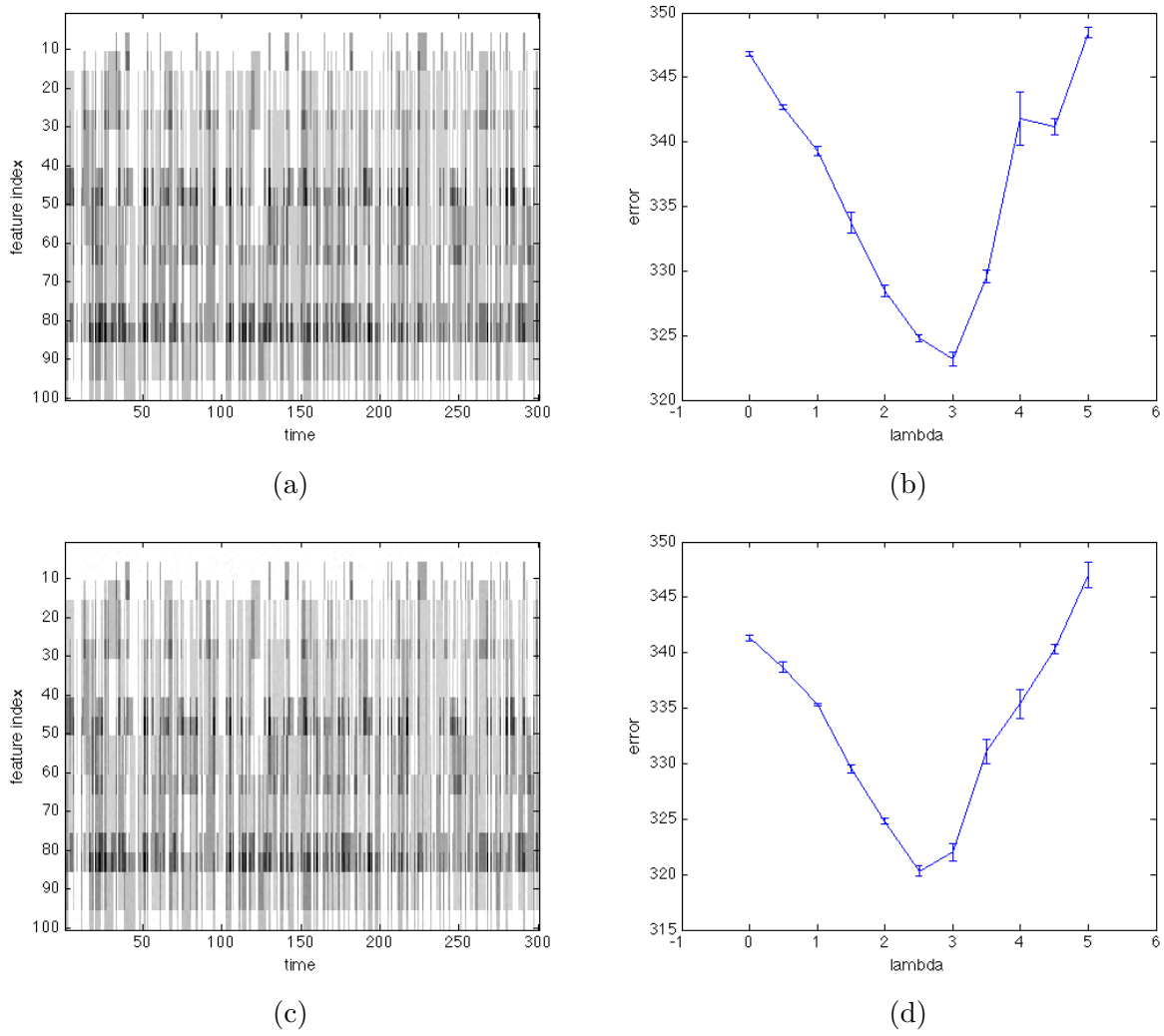


Figure 4.6: White Gaussian noise was added to data with signal to noise ratio (a) 50 dB and (c) 40 dB. Error rates with 95% confidence intervals averaged over ten trials versus λ values are shown in (b) and (d).

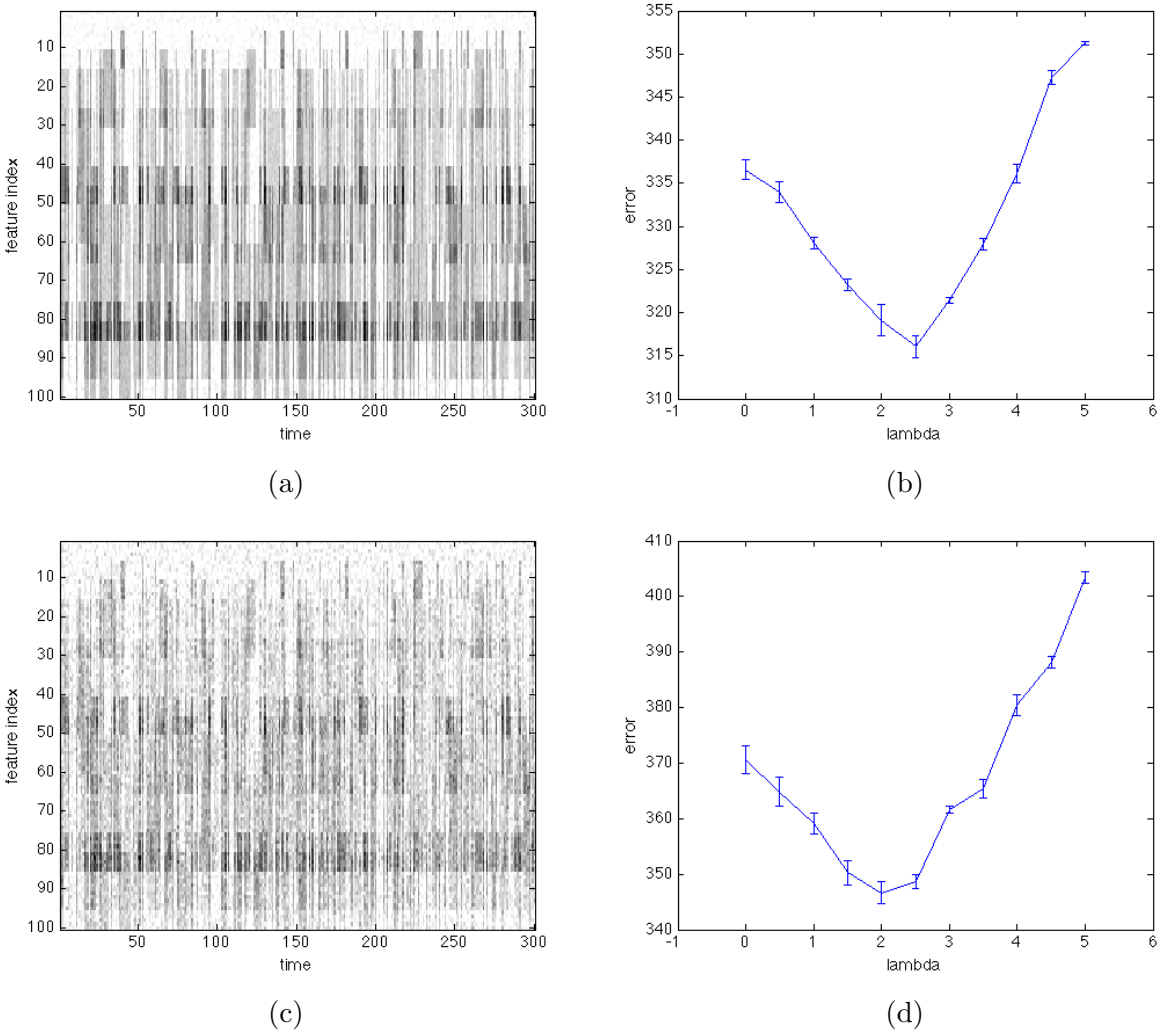


Figure 4.7: White Gaussian noise was added to data with signal to noise ratio (a) 30 dB and (c) 20 dB. Error rates with 95% confidence intervals averaged over ten trials versus λ values are shown in (b) and (d).

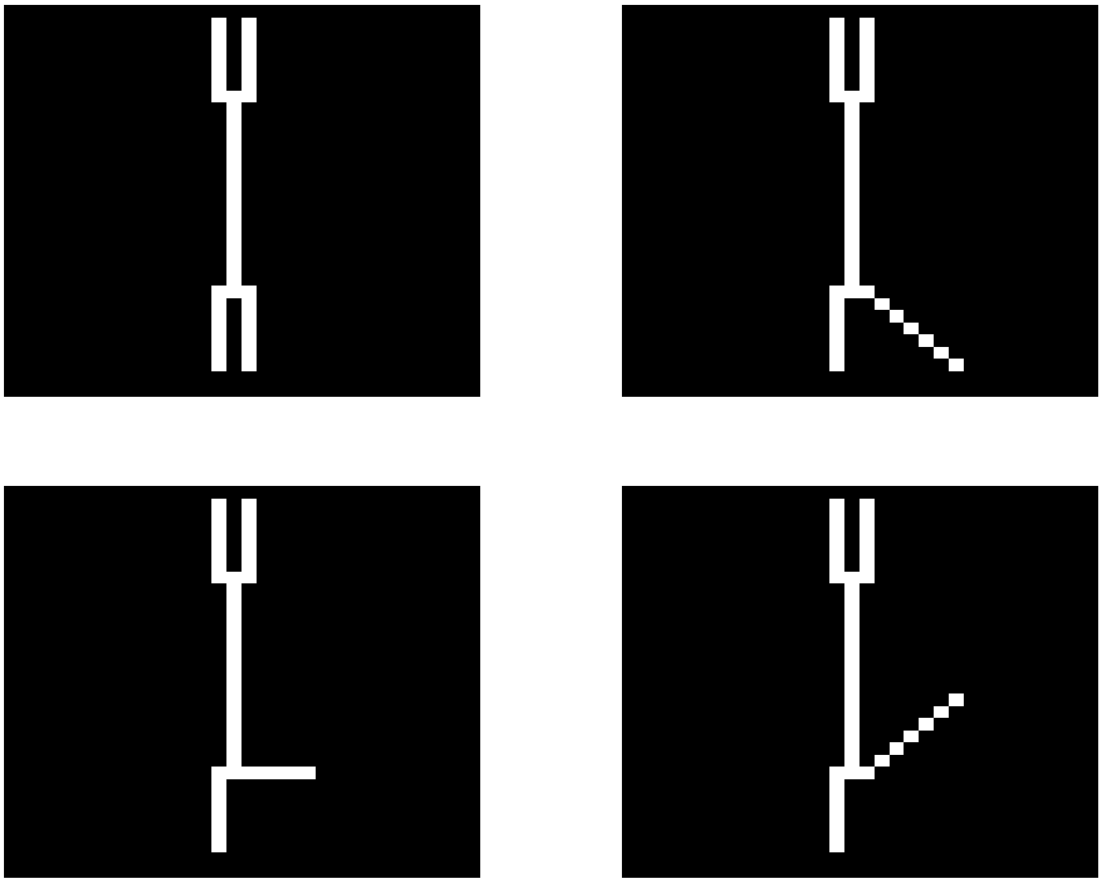


Figure 4.8: Four images form the swimmer dataset. Each image shows one of the four possible positions for the bottom right limb.

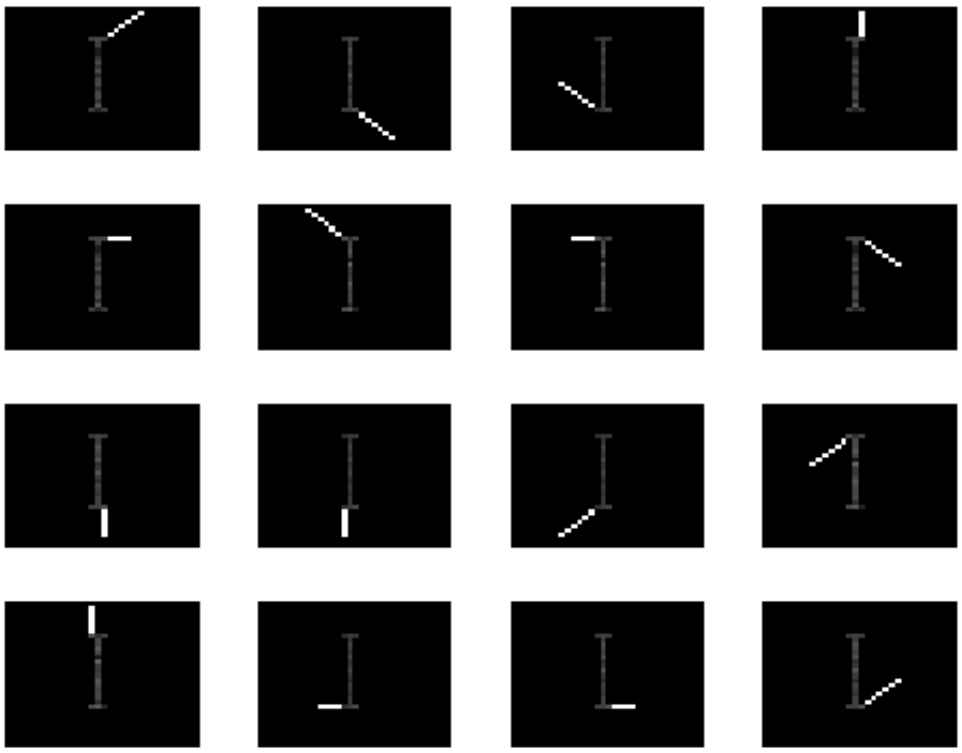


Figure 4.9: 16 pieces of the swimmer found by NMF.

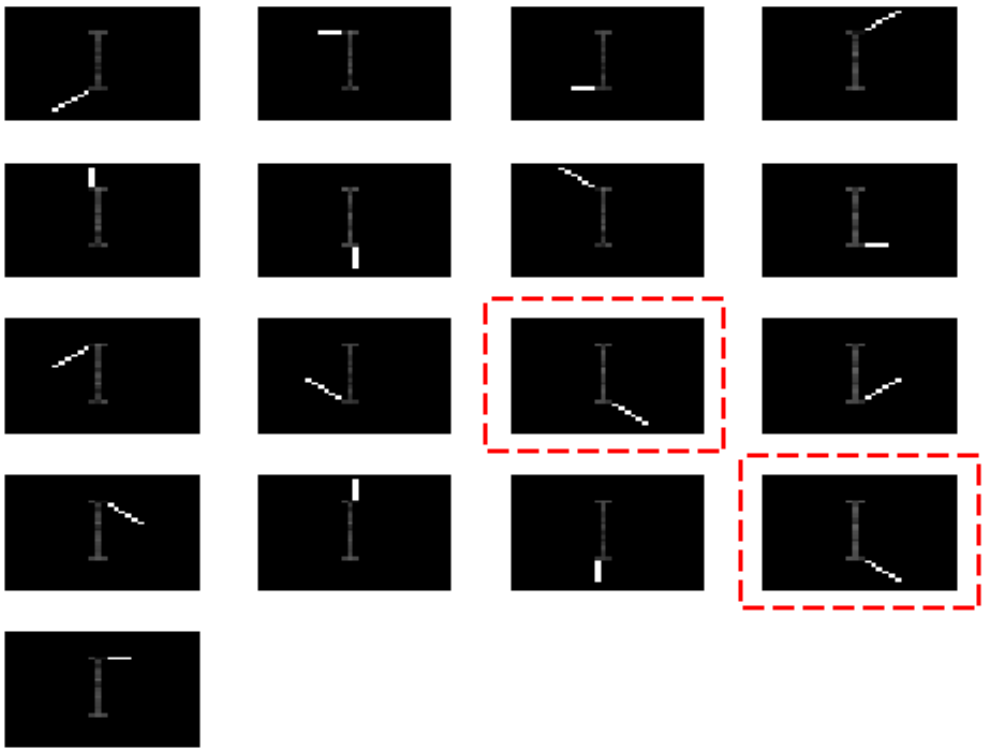


Figure 4.10: 17 pieces of the swimmer found by NMF. Note the two duplicates.

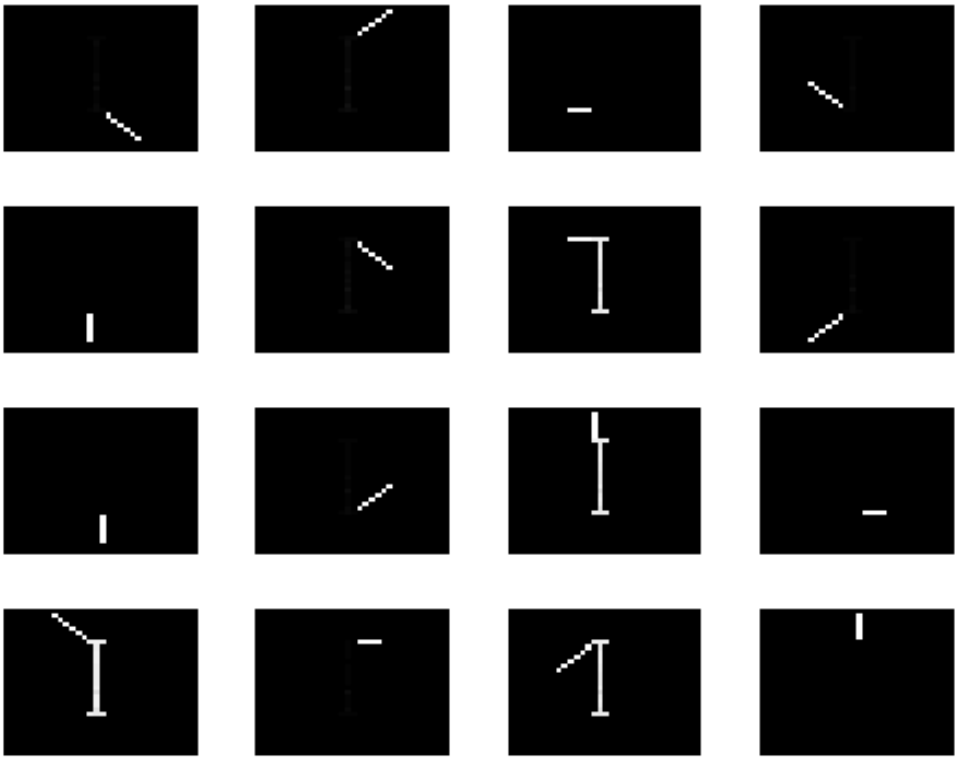


Figure 4.11: 16 pieces of the swimmer found by constraining NMF to have row sparsity in G.

Chapter 5

Real Data

5.1 Music transcription

The goal of music transcription is to take as input an audio recording of music and output its underlying sheet music. We begin with our audio signal in the form of a pressure time wave. We then take the magnitude of the short-time Fourier transform of the signal to get the data matrix X. We call this the spectrogram of the music signal. When finding the spectrogram we have several parameters to consider. The window type (such as gaussian or triangular), size, and overlap must be chosen. A small window gives good time accuracy but poor frequency accuracy (vice versa for a large window). In our experiments we make use of 100ms triangular windows overlapping by 50%. The frequencies at which to compute the spectrogram must also be considered. Though the human ear can hear up to 20kHz, music and speech generally fall below 5kHz. We have found that considering up to 2kHz is sufficient for our tasks.

Once we have the spectrogram, we can apply NMF to find B and G. Ideally B will contain the notes and G will contain the transcription. See Figure 5.1 for an outline of the process. As seen in the figure, B contains the seven notes as well as a vector corresponding to the note onsets. The rough transcription in G matches quite well the original sheet music.

Determining which basis vector corresponds to which note can be done by looking at the fundamental frequencies. The fundamental frequencies are the lowest frequency bands of each note. Each fundamental frequency (and hence each basis vector) can then be mapped to a single note.

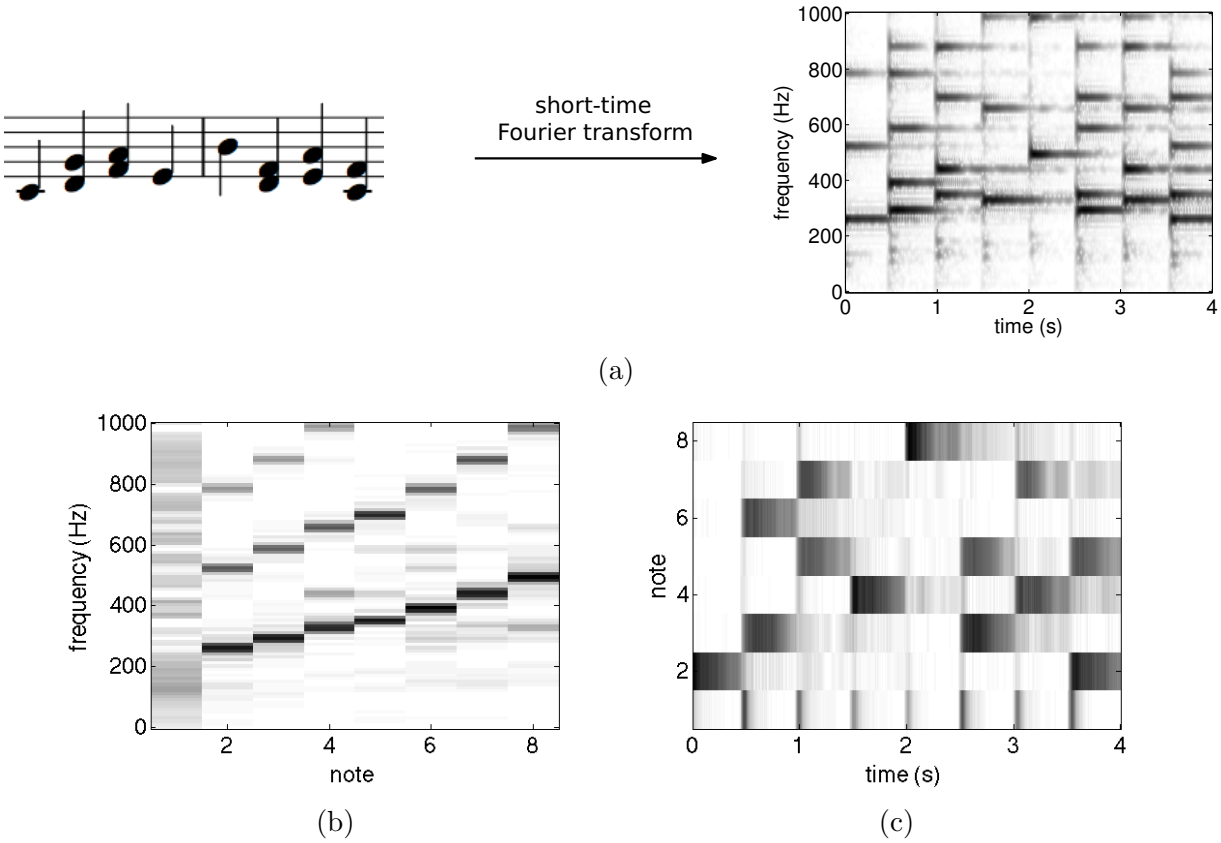


Figure 5.1: The short-time Fourier transform of an audio signal is taken to obtain the matrix X in (a). NMF produces (b) note matrix B and (c) note activation matrix G.

In practice it is generally better to learn B prior to the transcription. We can do so by computing B using NMF on a sample piece containing all possible notes. Thus, during NMF only G is updated which reduces the computation time and increases accuracy. See Algorithm 1 for an outline of the algorithm. Note that we scale the columns of B to have unit sum.

Once we have found our transcription matrix G, we must convert it to an actual transcription. Since the goal of our work is not to create a state of the art music transcriber, we propose a simple method of transcription. We convert G into a binary matrix under the following rule

$$G_{ij} = \begin{cases} 1 & \text{if } G_{ij} \geq \sigma \\ 0 & \text{if } G_{ij} < \sigma \end{cases}$$

Algorithm 1 Constrained NMF algorithm

for $scaling\_factor \leftarrow 10..\frac{1}{10}$ **do**
  for $i \leftarrow 1..num\_constraints$ **do**
    $\sigma_i \leftarrow scaling\_factor \cdot estimate(\sigma_i)$
  end for
  while change in $F(B, G) > \epsilon$ **do**
    $B_{ij} \leftarrow B_{ij} \frac{[\nabla^- F(X,BG)]_{ij}}{[\nabla^+ F(X,BG)]_{ij}}$
    $B \leftarrow scale(B)$
    $G_{ij} \leftarrow G_{ij} \frac{[\nabla^- F(X,BG)]_{ij}}{[\nabla^+ F(X,BG)]_{ij}}$
  end while
end for

where σ is the standard deviation of the elements of G. The binary G matrix is thus our transcription. That is, note i is activated at time j if and only if G_{ij} is equal to one.

We evaluate our transcription results by measuring the accuracies on the transcribed note onsets and offsets. We use four evaluation metrics. The first three are precision, recall, and F-score. These measure the accuracy of our transcribed note onsets. Precision is the fraction of correctly transcribed onsets out of the total number of transcribed onsets. Recall is the fraction of correctly transcribed onsets out of the total number of actual onsets. F-score (also called F-measure) is the harmonic mean of precision and recall.

A note onset is considered correct if it occurs within 50ms of the ground truth. Let *tp* and *fp* be the number of true and false positives respectively. Let *fn* be the number of false negatives. Then

$$\text{precision} = \frac{tp}{tp + fp}, \quad \text{recall} = \frac{tp}{tp + fn}, \quad \text{F-score} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

The last metric is mean overlap ratio (MOR). MOR is a measure of note offset accuracy. For each correctly transcribed note, we define on_g and on_t to be the ground truth onset time and transcribed onset time respectively. We define off_g and off_t for offset times similarly. The overlap ratio is then

$$\frac{\min\{off_g, off_t\} - \max\{on_g, on_t\}}{\max\{off_g, off_t\} - \min\{on_g, on_t\}}$$

We find MOR by averaging the overlap ratio of all the correctly transcribed notes.

5.1.1 MAPS

We evaluate our transcription algorithm on the MAPS (MIDI Aligned Piano Sounds) dataset [19] and compare our results to those found by Bertin et al. [8].

The MAPS dataset contains both synthetic and real audio samples. The synthetic pieces were generated from software. The real pieces were recorded on a piano in both ambient and close conditions. We test our algorithm on 30 pieces each of synthetic and real audio samples. Each sample is truncated to 30s. Our results as well as the results from Bertin et al. [8] are included below. Bertin et al. compare several different methods for music transcription. The following algorithms were tested in their work:

- NMF/MU - NMF minimizing Itakura-Saito divergence [20].
- S-NMF - space-alternating generalized expectation-maximization algorithm for NMF with smoothness constraint on G [20].
- Virtanen'07 - NMF with temporal continuity constraint minimizing Kullback-Leibler divergence [45].
- Vincent'08 - NMF with weighted Euclidean distance and harmonicity constraint [42].
- H-NMF/MU - Harmonic NMF [8].
- HS-NMF - Harmonic Smooth NMF [8].

We should note that in the methods tested by by Bertin et al., no training was done prior to transcription. The columns of B are either initialized with harmonic estimates prior to transcription or classified after the fact using a harmonic comb-based technique from [42]. This is in contrast to our method which makes use of an initial training phase to learn B from synthetic music generated from software [1]. In Tables 5.1 and 5.2 we list the results reported by Bertin et al. (first six rows) as well as the results of our constrained NMF method (last row).

In Tables 5.3, 5.4, and 5.5 we show the values of λ that gave the best results experimentally. Here, λ_{sm1}, λ_{sm2}, and λ_{sp} refer to the λ values corresponding to the piecewise smooth, piecewise smooth with aligned breakpoints, and element wise sparsity constraints respectively. For the synthetic data, the sparsity constraint alone is able to give good results. Adding in either piecewise smooth constraints shows little to no improvement. However, when we consider real data, we see improvements when considering both sparsity and piecewise smoothness.

Algorithm	F-score	Precision	Recall	MOR
NMF/MU	0.549	0.634	0.561	0.512
Vincent'08	0.584	0.607	0.600	**0.548**
H-NMF/MU	0.524	0.587	0.591	0.460
S-NMF	0.495	0.624	0.433	0.507
Virtanen'07	0.536	0.559	0.564	0.521
HS-NMF	**0.607**	**0.658**	**0.645**	0.443
Constrained NMF	**0.706** ($\pm$0.055)	**0.727** ($\pm$0.051)	**0.701** ($\pm$0.066)	**0.529** ($\pm$0.035)

Table 5.1: The results of Bertin et al. and constrained NMF for synthetic music samples. The top two results in each column are in bold.

Algorithm	F-score	Precision	Recall	MOR
NMF/MU	0.408	0.433	0.434	0.477
Vincent'08	0.361	0.387	0.374	**0.500**
H-NMF/MU	0.413	0.430	0.427	0.446
S-NMF	0.366	0.462	0.320	0.456
Virtanen'07	0.336	0.342	0.348	0.471
HS-NMF	**0.450**	**0.466**	**0.453**	0.432
Constrained NMF	**0.539** ($\pm$0.063)	**0.563** ($\pm$0.068)	**0.544** ($\pm$0.071)	**0.565** ($\pm$0.040)

Table 5.2: The results of Bertin et al. and constrained NMF for real (ambient) music samples. The top two results in each column are in bold.

λ_{sm1}	λ_{sm2}	λ_{sp}	F-score	Precision	Recall	MOR
0.0	0.0	0.0	0.379 ($\pm$0.033)	0.257 ($\pm$0.027)	0.751 ($\pm$0.055)	0.574 ($\pm$0.033)
0.0	0.0	1.8	0.704 ($\pm$0.049)	0.665 ($\pm$0.046)	0.759 ($\pm$0.063)	0.521 ($\pm$0.035)
0.0	1.5	0.0	0.431 ($\pm$0.046)	0.376 ($\pm$0.037)	0.540 ($\pm$0.079)	0.541 ($\pm$0.051)
0.0	0.5	1.5	0.706 ($\pm$0.055)	0.727 ($\pm$0.051)	0.701 ($\pm$0.066)	0.529 ($\pm$0.035)
0.5	0.0	0.0	0.587 ($\pm$0.056)	0.608 ($\pm$0.045)	0.586 ($\pm$0.072)	0.549 ($\pm$0.049)
0.1	0.0	0.9	0.696 ($\pm$0.069)	0.730 ($\pm$0.072)	0.692 ($\pm$0.069)	0.544 ($\pm$0.042)

Table 5.3: Constrained NMF results for synthetic data with 95% confidence intervals.

5.1.2 Sparseness

Choosing a suitable r is not always clear. In music transcription, r should be equal to the number of notes in the piece, which is not always known in advance. In other applications,

λ_{sm1}	λ_{sm2}	λ_{sp}	F-score	Precision	Recall	MOR
0.0	0.0	0.0	0.251 ($\pm$0.030)	0.158 ($\pm$0.021)	0.644 ($\pm$0.066)	0.565 ($\pm$0.035)
0.0	0.0	2.4	0.502 ($\pm$0.055)	0.480 ($\pm$0.058)	0.542 ($\pm$0.058)	0.495 ($\pm$0.040)
0.0	1.0	0.0	0.367 ($\pm$0.040)	0.302 ($\pm$0.031)	0.511 ($\pm$0.078)	0.536 ($\pm$0.049)
0.0	0.5	1.5	0.493 ($\pm$0.060)	0.509 ($\pm$0.063)	0.503 ($\pm$0.067)	0.546 ($\pm$0.030)
0.5	0.0	0.0	0.456 ($\pm$0.052)	0.446 ($\pm$0.047)	0.509 ($\pm$0.078)	0.523 ($\pm$0.055)
0.1	0.0	0.9	0.539 ($\pm$0.063)	0.563 ($\pm$0.068)	0.544 ($\pm$0.071)	0.565 ($\pm$0.040)

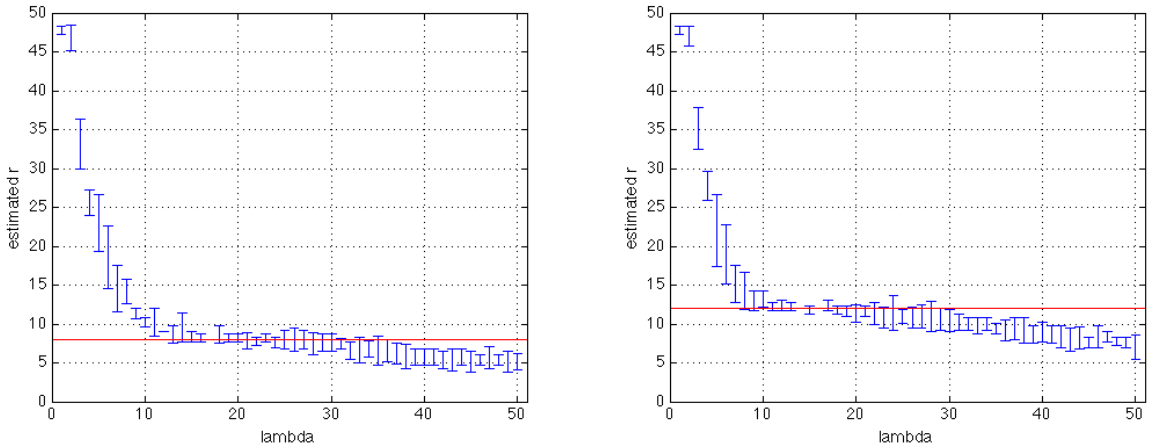
Table 5.4: Constrained NMF results for real (ambient) data with 95% confidence intervals.

λ_{sm1}	λ_{sm2}	λ_{sp}	F-score	Precision	Recall	MOR
0.0	0.0	0.0	0.314 ($\pm$0.026)	0.198 ($\pm$0.019)	0.789 ($\pm$0.046)	0.631 ($\pm$0.026)
0.0	0.0	1.8	0.661 ($\pm$0.049)	0.597 ($\pm$0.058)	0.762 ($\pm$0.045)	0.558 ($\pm$0.031)
0.0	1.0	0.0	0.442 ($\pm$0.026)	0.341 ($\pm$0.021)	0.657 ($\pm$0.058)	0.628 ($\pm$0.031)
0.0	0.5	1.5	0.698 ($\pm$0.043)	0.685 ($\pm$0.046)	0.724 ($\pm$0.052)	0.584 ($\pm$0.031)
0.5	0.0	0.0	0.618 ($\pm$0.041)	0.591 ($\pm$0.037)	0.667 ($\pm$0.058)	0.641 ($\pm$0.036)
0.1	0.0	0.6	0.715 ($\pm$0.041)	0.699 ($\pm$0.045)	0.747 ($\pm$0.052)	0.637 ($\pm$0.025)

Table 5.5: Constrained NMF results for real (close) data with 95% confidence intervals.

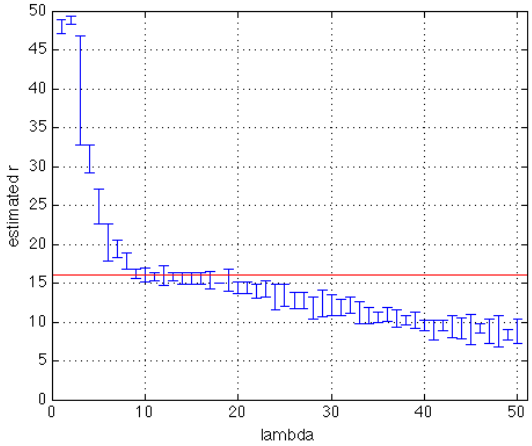
such as facial analysis, there may be no correct value of r. We propose a method to automatically select a suitable r by imposing sparsity on the rows of G. In order to test its effectiveness we generated music data from midi synthesizer software [1]. First, the number of notes, r, in the piece is chosen. Next, r notes are chosen randomly from a number of adjacent octaves. These r notes will make up the piece. For every time step of duration t, we randomly select $k < r$ notes from the possible notes and play them for t seconds. This process is repeated until desired. Each note in the piece is thus never played by itself (i.e. it is always accompanied by $k-1$ other notes). We run our constrained NMF algorithm on each piece and count the number of nonzero rows of G (discounting those that correspond to note onsets). The row count is our guess at the underlying r value. The results are included in Figure 5.2.

As can be seen, the estimated means are within error of the true mean for a large range of λ values. Hence, though the choice of λ is important to get good results, a precise choice is not necessary. Also note that for smaller r the algorithm appears to be more accurate.

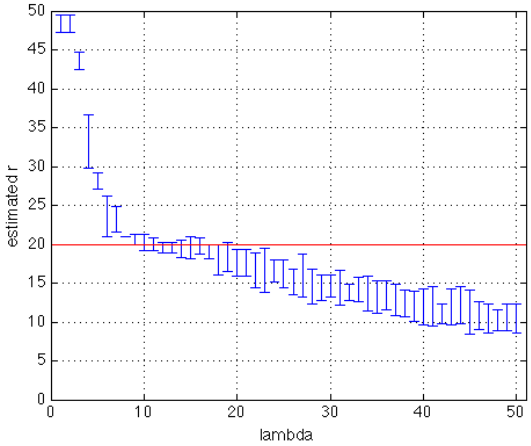


(a)

(b)



(c)



(d)

Figure 5.2: The mean r estimates over five trials with $k = 3$ and true r values (solid line) (a) $r = 8$, (a) $r = 12$, (a) $r = 16$, and (a) $r = 20$.

5.2 Instrument separation

Up until now we have only considered single instrument music. In real music there are usually many instruments playing simultaneously. For our next experiment we generate synthetic music of a violin and clarinet. At every time step, we pick three notes at random from either instrument from a single octave. In our first experiment we see if regular NMF is able to pick out the 24 distinct notes (12 notes in an octave per instrument). We set $r = 25$ (one for the note onsets) and run the algorithm with no constraints. See Figure 5.3 for the results. As shown in the figure, NMF is able to pick out 12 pairs of notes.

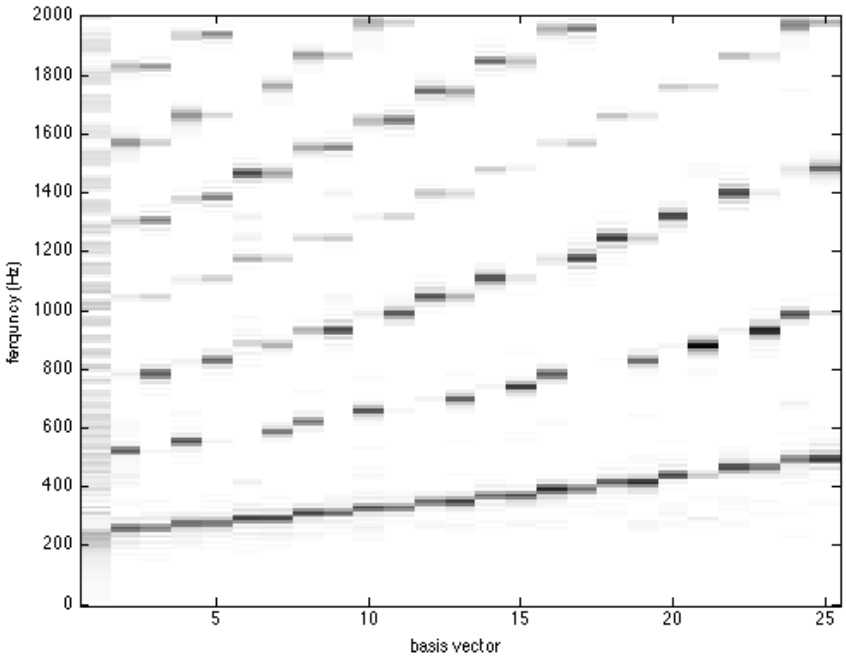


Figure 5.3: Basis vectors found by NMF on violin and clarinet music. Note the pairs of notes.

In general we may not know the number of instruments or the number of notes in a piece of music. As in our earlier case where we used row sparsity to guess the number of piano notes used, we can see if constrained NMF is able to distinguish the different instruments. See Figure 5.4 for the results with $r = 100$ and $\lambda = 25$. The algorithm finds 25 basis vectors: 11 pairs of notes, one single note, one extraneous overtone, and one onset vector.

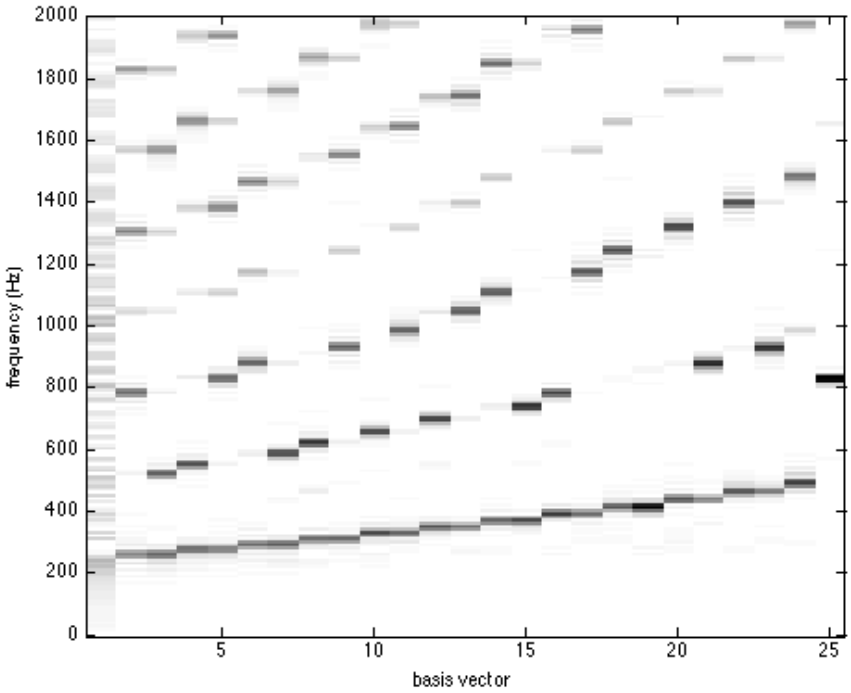
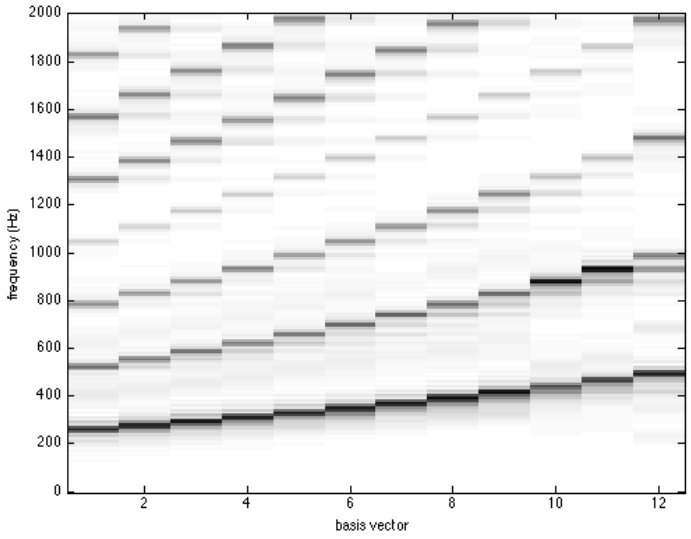


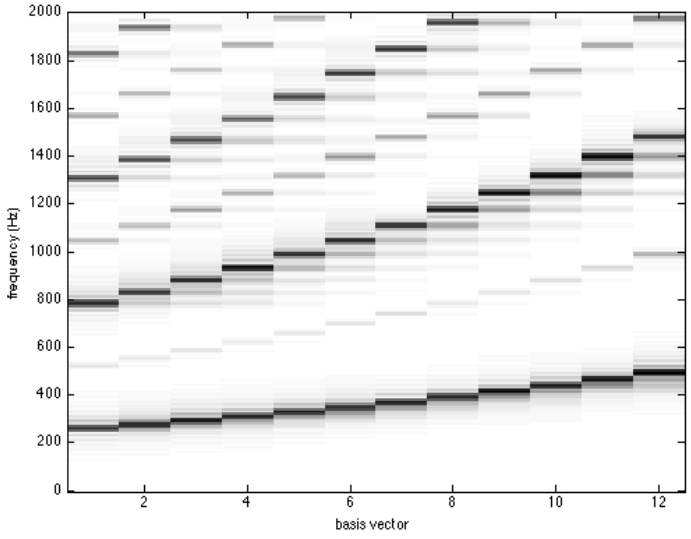
Figure 5.4: Basis vectors found by NMF on violin and clarinet music using the row sparsity constraint with $\lambda = 25$. Note that only basis vectors corresponding to nonzero rows of G are included.

The next step would be to classify the basis vectors into the two instrument classes. We leave such classification for future work. Instead, in our next experiment we learn the basis vectors for each instrument separately so there is no need for classification. See Figure 5.5 for a comparison of the basis vectors. Though both instruments share the same harmonic frequencies for each note, they differ in how each harmonic is emphasized. Note that the clarinet emphasizes odd harmonics.

Once we have learned the basis vectors, we combine them into one basis matrix. Next we generate random test music. For every half second increment we randomly select three notes from a single octave. We then randomly assign each note to be either played by violin or clarinet. We run piece wise smooth NMF with varying λ and record the transcription F-scores in Figure 5.6 for the violin and clarinet notes separately. As can be seen, choosing $0 < \lambda < 1$ results in the highest F-scores.

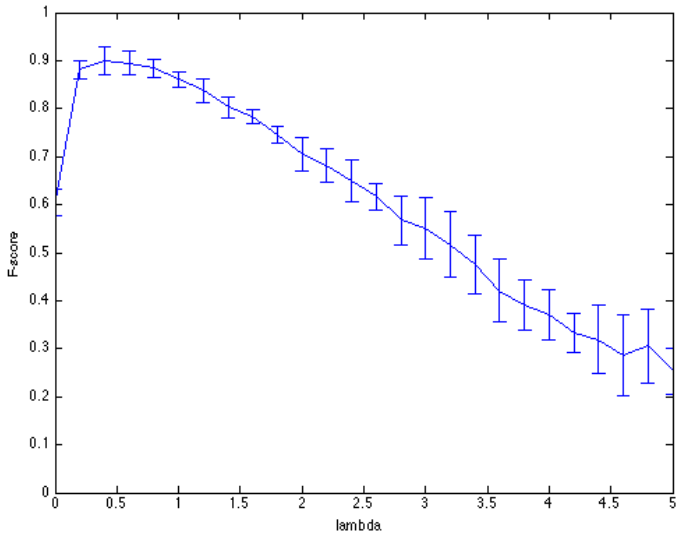


(a)

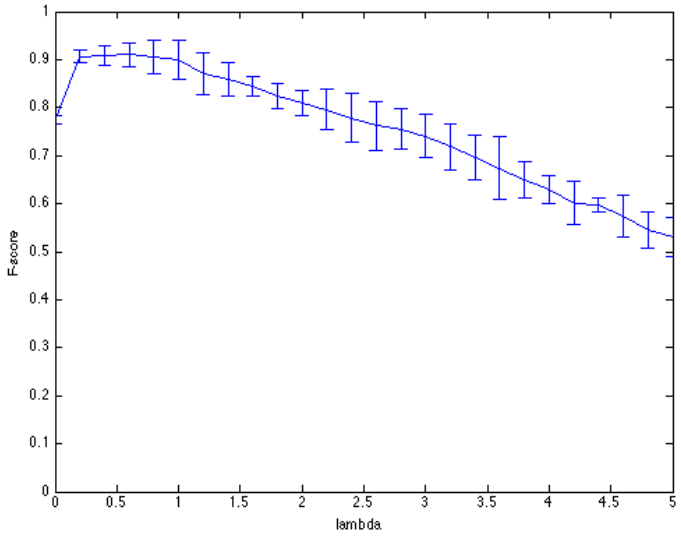


(b)

Figure 5.5: Basis vectors learned for (a) violin and (b) clarinet. Note how the clarinet differs by emphasizing odd harmonics.



(a)



(b)

Figure 5.6: Mean transcription F-scores over five trials with 95% confidence intervals for (a) violin notes and (b) clarinet notes.

5.3 Speech

In many ways, speech is similar to music. Both are audio signals and are composed of parts. The parts of music are notes, and the parts of speech are phonemes. This motivates us to use NMF for speech transcription. Let us first look at an excerpt of speech in Figure 5.7. One can, with a small bit of effort, pick out where one phoneme ends and the next begins. However, many of the different phonemes look very similar. This was not the case with music where two notes were noticeably distinct.

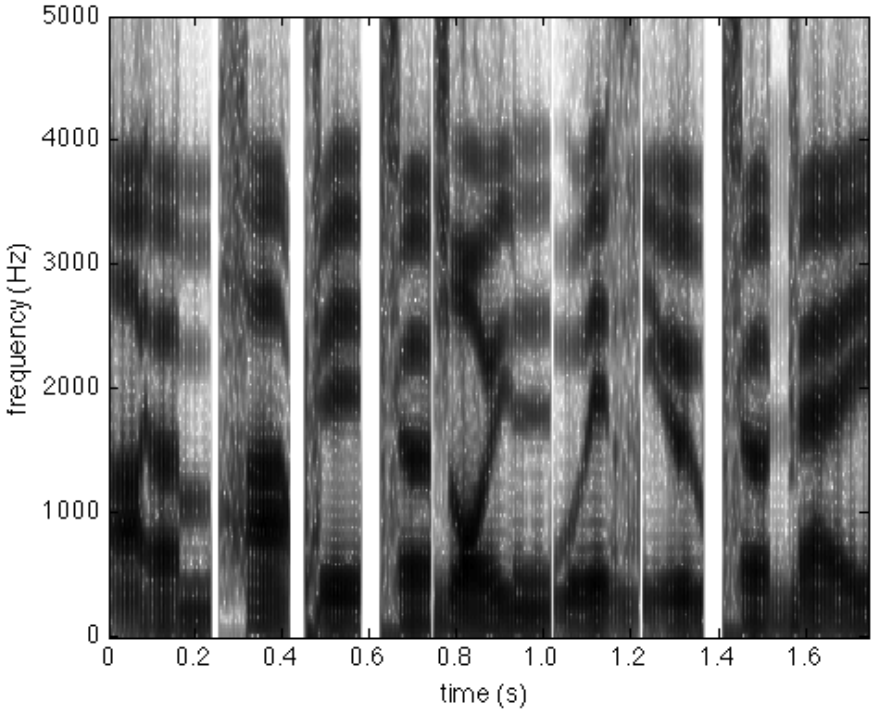
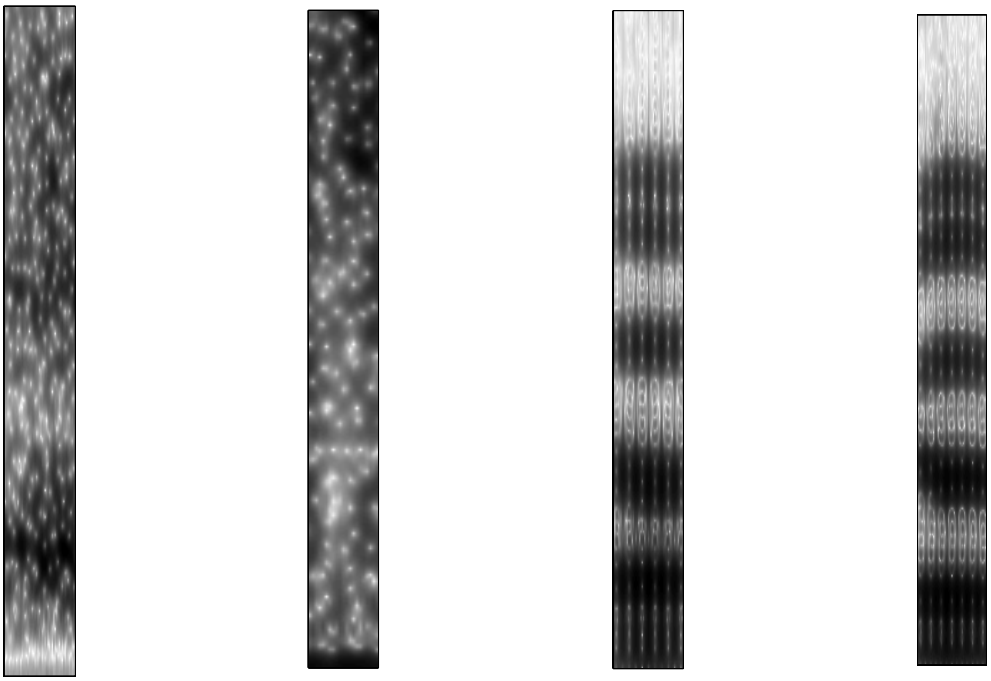


Figure 5.7: Spectrogram of speech. The text is: "I am happy to join with you today". The audio data was generated using Praat [10].

Upon first inspection it may seem that each phoneme is fairly easily distinguishable from the others. This would imply that NMF may be able to perform accurate speech transcription. However, let us consider some specific phonemes from the spectrogram. See Figure 5.8 for and example of four different phonemes. The 'h' and 't' phonemes in Figures 5.8a and 5.8b are difficult to distinguish because they have very little structure.

However, even phonemes with structure can be difficult to distinguish. Consider the 'ɐ' and 'ə' phonemes in Figures 5.8c and 5.8d.



(a) (b) (c) (d)

Figure 5.8: The spectrogram of four different phonemes: (a) h, (b) t, (c) ɐ, (d) ə.

Another issue with speech data is that there is only a single phoneme being spoken at a given point in time. This is in contrast to music where there are usually multiple notes being played at once. Since NMF tries to segment data points into parts, it tends to have difficulty handling single phonemes. For instance, NMF may split a single phoneme into multiple parts. Suppose we run NMF on the audio sample from Figure 5.7. The factorization is included in Figure 5.9. Though some basis vectors if Figure 5.9a look similar to phonemes (say vectors 6 and 15), the transcription in Figure 5.9b shows that there is not a one-to-one correspondence between phonemes and basis vectors. That is, there is no discernible time at which a single basis vector is activated. In fact at most time points a large number of basis vector are activated.

5.3.1 Speaker separation

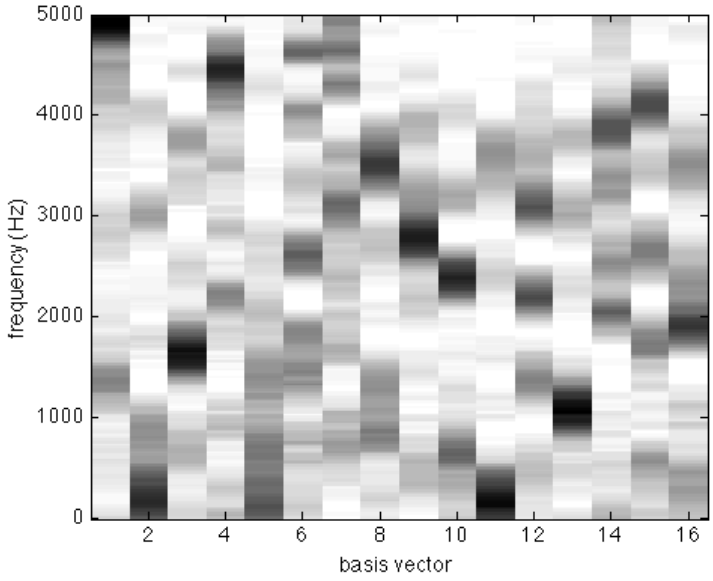
Though NMF seems to fail for single speaker data, there may be hope for multiple speaker data. We encounter such situations regularly. For example, we may be having a conversation with an individual in a noisy coffee shop. In such an environment, we need to separate out a single speaker from background noise as well as other nearby conversations. In order to investigate this problem, we consider an audio recording in which two speakers are speaking simultaneously. We wish to separate each speaker individually. NMF based approaches have been studied in the literature [37]. We generate synthetic speech data using Praat [10]. In our experiments we first train on speech data from the individual speakers. For each speaker we set $r = 20$ and learn a B matrix. We then concatenate the two B matrices into one so that $r = 40$. We now run NMF on the mixed speaker sample in Figure 5.10 while holding B constant. Once we have learned G we can reconstruct each speakers spectrograms. The spectrogram of speaker 1 is found by multiplying the first half columns of B with the first half rows of G. The spectrogram for speaker 2 is found similarly. See Figures 5.11 and 5.12 for a comparison of the the ground truth and estimated spectrograms.

We see in the figures that NMF is able to correctly separate the two speakers' spectrograms (at least at lower frequencies). This suggests that NMF learns basis vectors which are unique to each speaker. Ideally we would like to separate speakers without any prior training, as humans are generally able to do. Humans, however, are able to make use of more than just auditory information. We also make use of visual cues, such as lip movements, in order to aid us in hearing in a noisy environment [47]. Hence, developing methods which incorporate more than just auditory methods is important for the task of speaker separation.

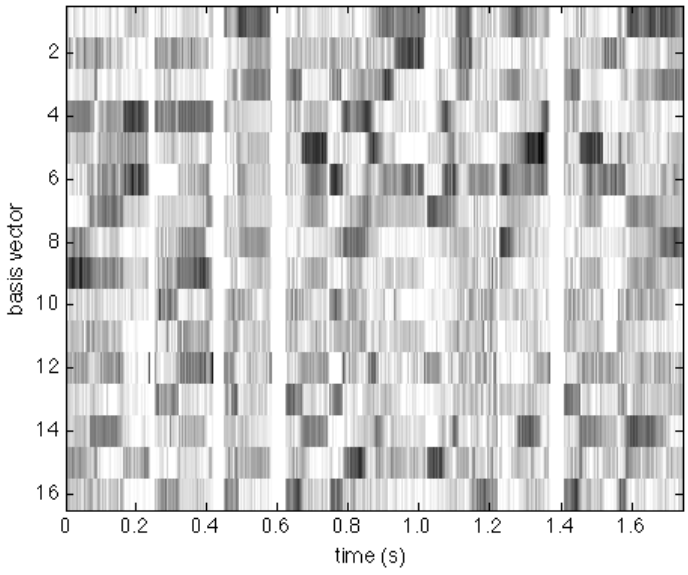
5.4 Character analysis

We now look at the problem of character analysis. Here we consider digits taken from the MNIST database of handwritten digits [27]. Each digit is a 28×28 greyscale image. The digits are centred using their pixel centres of mass. We reshape each image into a single vector to make the columns of our X matrix. Figure 5.13 shows the basis vectors found by NMF with the element wise sparseness constraint. Note as λ increases some basis vectors begin to resemble full digits. Other basis vectors converge to small dots. This would suggest that as the cost for activating a basis vector increases the basis vectors tend toward full digit prototypes (similar to k-means).

Now suppose we impose the row sparsity constraint. Figure 5.14 shows the basis vectors for varying λ values. Note how most basis vectors resemble full digits. Further note that when $\lambda \geq 45$ the digits '2', '5', and '8' appear to be missing. This can be explained by looking at the distribution of digits in Figure 5.15. As can be seen, the three digits with the fewest examples are eliminated from the bases vectors as λ increases. We also see some digits represented by multiple basis vectors (for example, the digits '0', '1', and '7' when $\lambda = 55$). This would suggest that those digits have the greatest variability in how they are written.



(a)



(b)

Figure 5.9: The results of running NMF of the speech data in Figure 5.7. (a) B, (b) G.

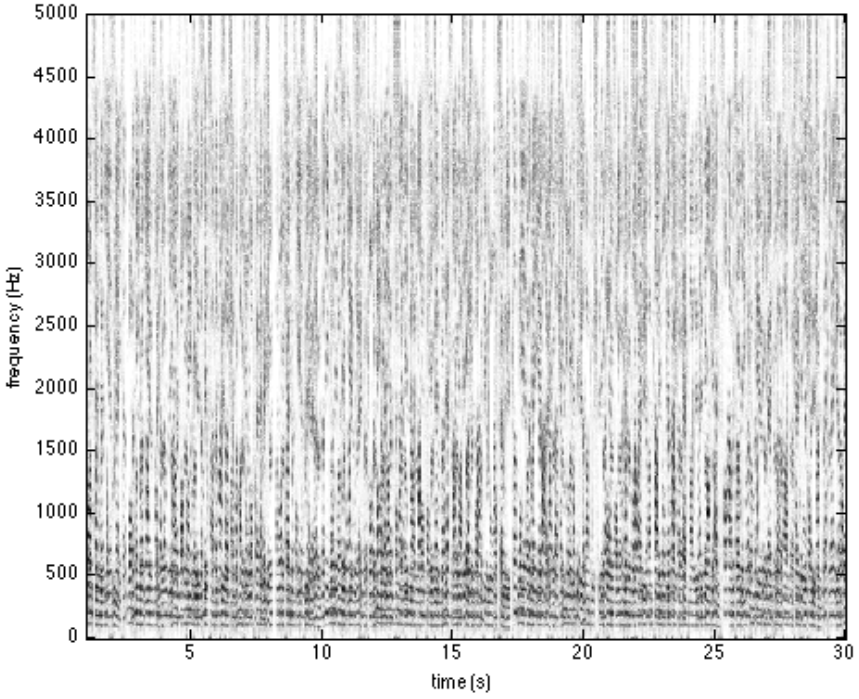
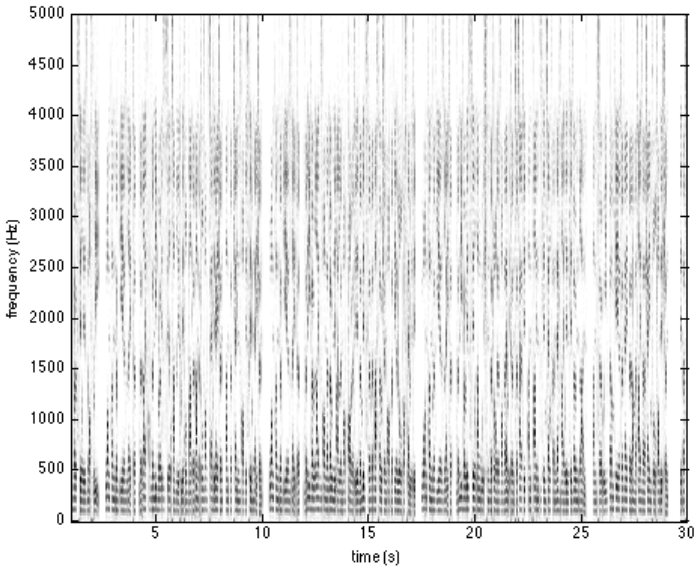
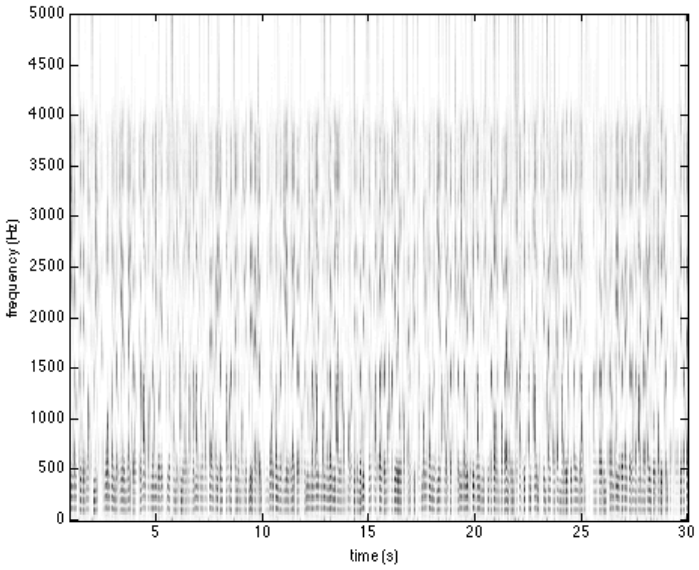


Figure 5.10: Spectrogram of mixed speech from two speakers.

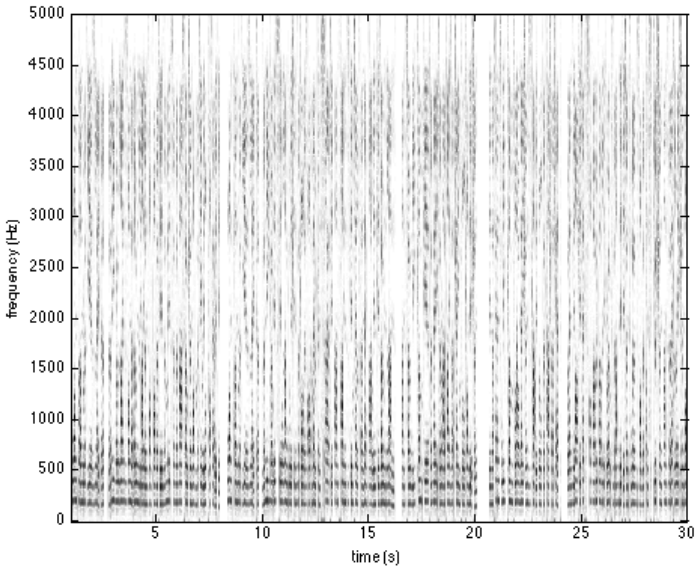


(a)

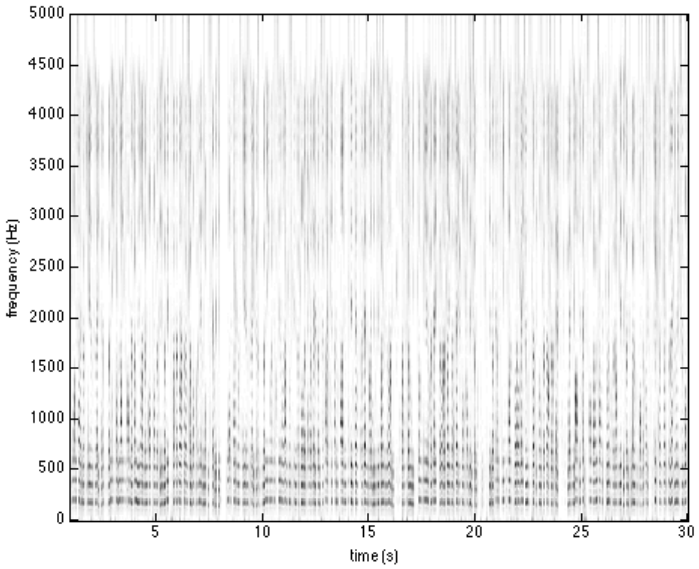


(b)

Figure 5.11: (a) Original and (b) reconstructed spectrogram of speaker 1.

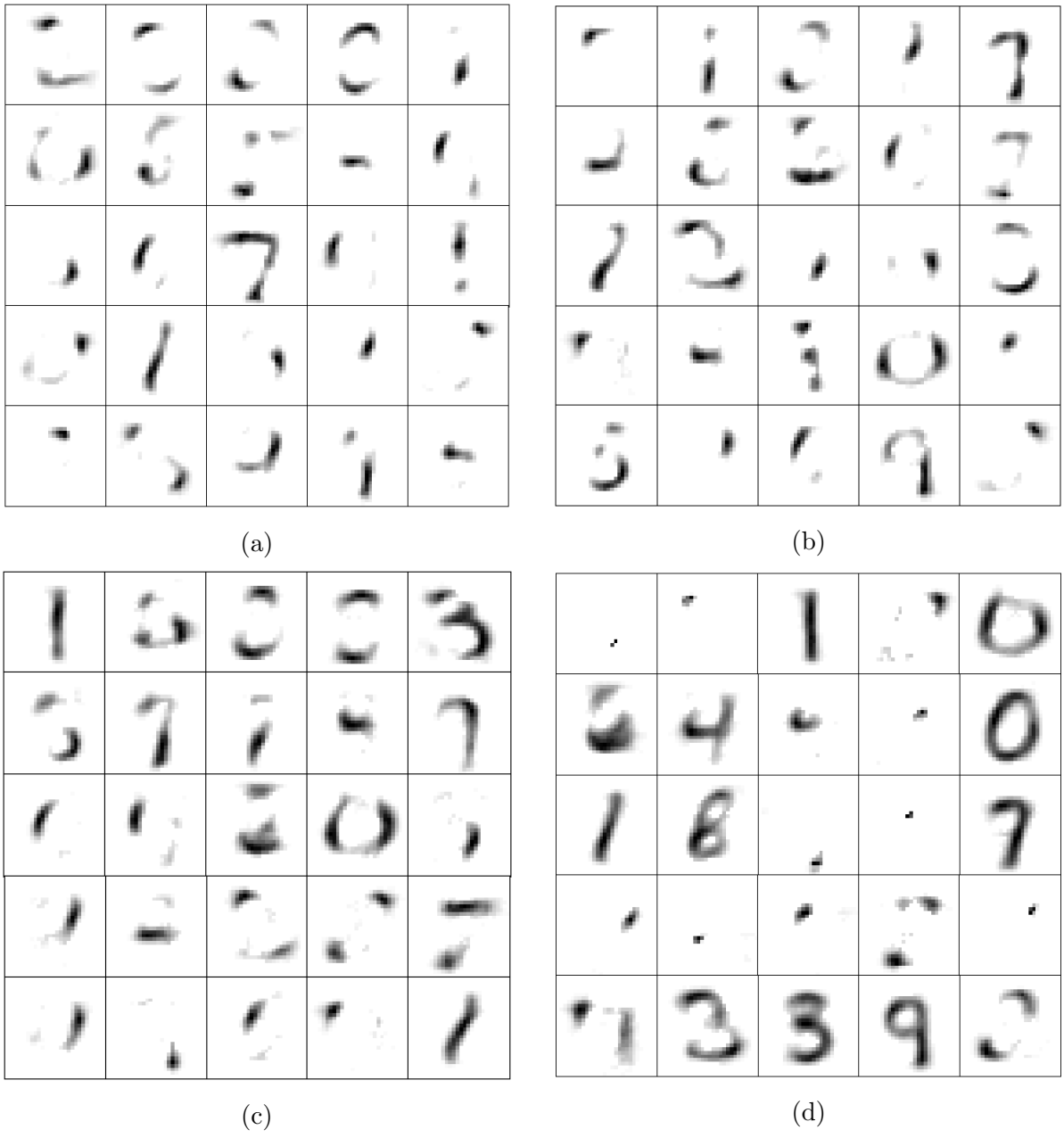


(a)



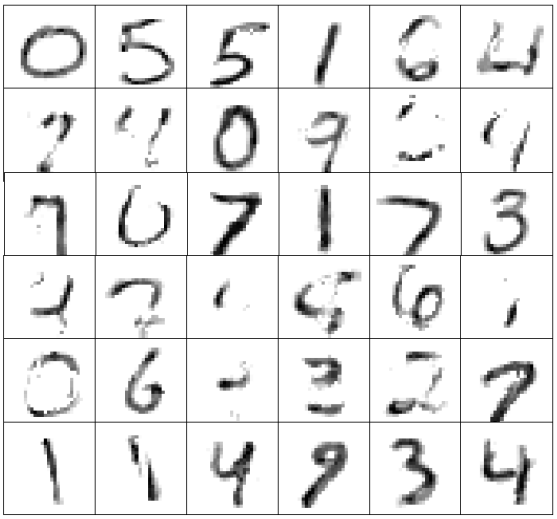
(b)

Figure 5.12: (a) Original and (b) reconstructed spectrogram of speaker 2.

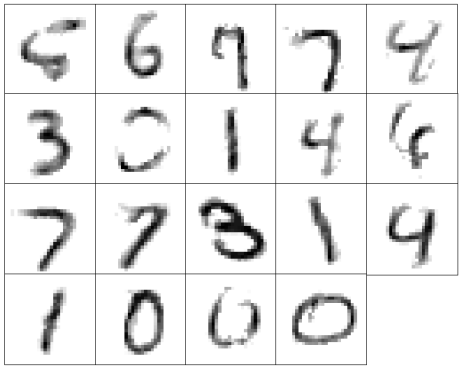


(a) (b) (c) (d)

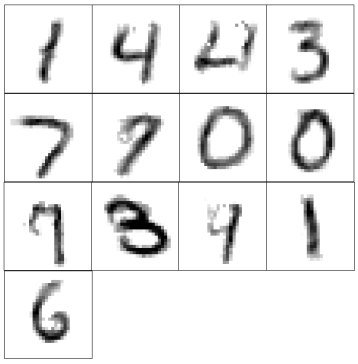
Figure 5.13: Basis vectors found from digit dataset with varying sparseness constraint. (a) $\lambda = 0$, (b) $\lambda = 1$, (c) $\lambda = 3$, (d) $\lambda = 5$.



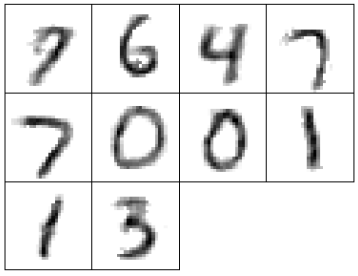
(a)



(b)



(c)



(d)

Figure 5.14: Basis vectors found from digit dataset with varying row sparseness constraint. (a) $\lambda = 25$, (b) $\lambda = 35$, (c) $\lambda = 45$, (d) $\lambda = 55$.

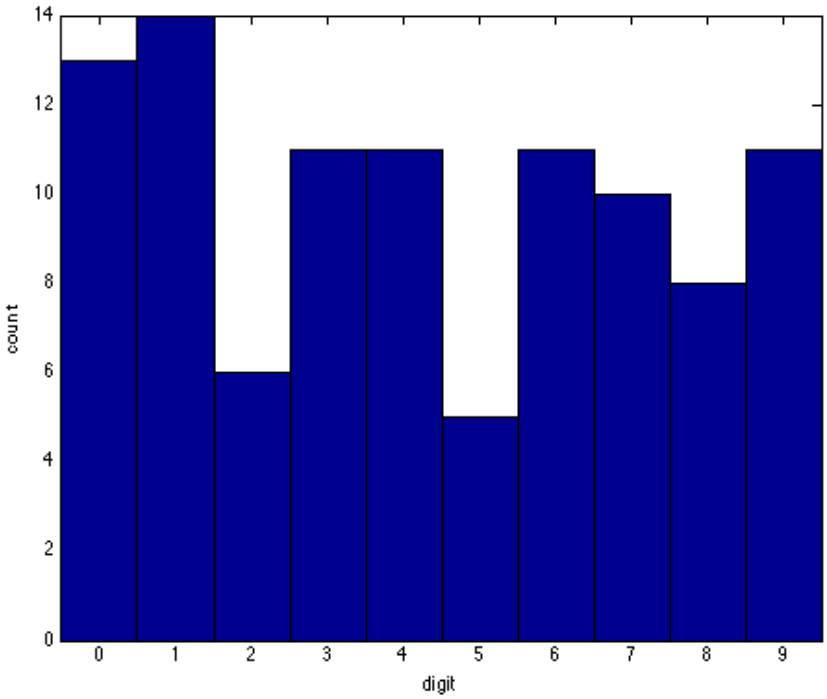


Figure 5.15: Distribution of digits in first 100 samples of the MNIST dataset.

Chapter 6

Conclusion and Future Work

In this work we have explored using NMF in several applications. Most notably we have demonstrated the effectiveness of NMF for music transcription. The key reason why NMF performs so well in transcription is because of the additive nature of music. Each time point in a musical piece is composed of a sum of notes. When we consider a time-frequency representation of music, we further note that each sum is nonnegative. Thus, NMF is able to exploit this nonnegative additivity in order to factorize out both the individual notes and the transcription. At first these results may seem too good to be true, but in reality it is the consequence of finding the correct model for our data.

In order to improve the performance of NMF we considered different constraints on the cost function. The constraints are derived from observations of real music data. We note that music is generally composed of piecewise smooth parts, and hence we impose a piecewise smooth constraint. Furthermore, note onsets/offsets often occur simultaneously, inspiring another contraint which favours aligned breakpoints. Finally, relatively few notes tend to be activated at a given time, and so we impose sparsity. We demonstrated the novelty of the constraints on synthetic and real music data. We show promising results which exceed the current state of the art. We also explored other interesting applications, such as instrument and speaker separation and handwritten character analysis.

There remain many avenues of future research in NMF. In our work we only consider a single cost function (divergence). Investigating different cost functions and their behaviour with constraints is a possible future route. For example, we may consider the squared Frobenius norm as in [30]. Other optimization methods can also be explored. Several different methods have been explored in the literature including: regularized alternating least squares [14], projected gradient descent [31], active set methods [24], and block principal

pivoting [25].

Recently, Arora et al. [3] proved that a polynomial time algorithm for NMF exists if the separability condition holds. A factorization is separable if we can permute r columns of G to get the identity. In the context of music transcription, this would mean that each note is played in isolation at some point in the piece. However, this assumption does not hold in general as can be seen even in our simple example from Figure 1.6. Further investigation is required in order to determine if a similar polynomial time algorithm can be found for music transcription.

Another possible area of interest is adding higher level concepts to NMF. In the case of music, our constrained NMF model only has the notion of notes that are piecewise smooth and sparse. There are many other properties, such as the notion of bars, tempo, time signature, and key that could be added into the model. In the case of speech we may wish to add the notion of words or phrases. Furthermore, we could use methods from natural language processing in order to improve the speech factorizations.

The use of NMF for audio analysis makes several assumptions about the behaviour of sound. One assumption is that when two sounds occur at the same time, they add together. However, Wegel and Lane [48] demonstrate that a masking effect occurs when multiple sounds at similar frequencies are heard simultaneously. For example, a very loud noise can mask a quieter sound that would otherwise be audible. Another assumption is that the human ear responds linearly to changes in volume. However, in order to achieve the large dynamic range of our audio system, our cochleas compress the volume of loud sounds [34]. In other words, the perceived difference between 10 and 20 dB sounds is greater than the difference between 80 and 90 dB sounds. Investigating ways to modify NMF in order to account for these assumptions is a possible line of future research.

References

[1] Aria Maestosa version 1.4.9. http://sourceforge.net/projects/ariamaestosa/, 2013.

[2] S.A. Abdallah and M.D. Plumbley. Polyphonic transcription by non-negative sparse coding of power spectra. In *5th int. conf. on music information retrieval*, pages 318–325, 2004.

[3] Sanjeev Arora, Rong Ge, Ravi Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization – provably. *CoRR*, abs/1111.0952, 2011.

[4] E Benetos and S Dixon. A shift-invariant latent variable model for automatic music transcription. *Computer Music Journal*, 36(4):81–94, Dec 2012.

[5] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[6] Michael W. Berry, Murray Browne, Amy N. Langville, V. Paul Pauca, and Robert J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. In *Computational Statistics and Data Analysis*, pages 155–173, 2006.

[7] N. Bertin, R. Badeau, and G. Richard. Blind signal decompositions for automatic transcription of polyphonic music: NMF and K-SVD on the benchmark. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 1, pages I–65–I–68, April 2007.

[8] N. Bertin, R. Badeau, and E. Vincent. Enforcing harmonicity and smoothness in bayesian non-negative matrix factorization applied to polyphonic music transcription. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(3):538–549, March 2010.

[9] Andrew Blake and Andrew Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, MA, USA, 1987.

[10] Paul Boersma. Praat, a system for doing phonetics by computer. *Glot International*, 5(9/10):341–345, 2001.

[11] G. E. P. Box and S. L. Andersen. Permutation theory in the derivation of robust criteria and the study of departures from assumption. *Journal of the Royal Statistical Society. Series B (Methodological)*, 17(1):pp. 1–34, 1955.

[12] Ioan Buciu. Non-negative matrix factorization, a new tool for feature extraction: Theory and applications. *Int. J. Computers, Communications and Control*, 3:67–74, 2006.

[13] Moody Chu and Robert Plemmons. Nonnegative matrix factorization and applications. *Bulletin of the International Linear Algebra Society*, 34:2–7, 2005.

[14] Andrzej Cichocki and Rafal Zdunek. Regularized alternating least squares algorithms for non-negative matrix/tensor factorization. In Derong Liu, Shumin Fei, Zengguang Hou, Huaguang Zhang, and Changyin Sun, editors, *Advances in Neural Networks ISNN 2007*, volume 4493 of *Lecture Notes in Computer Science*, pages 793–802. Springer Berlin Heidelberg, 2007.

[15] Arshia Cont. Realtime multiple pitch observation using sparse non-negative constraints. In *International Conference on Music Information Retrieval*, 2006.

[16] M. Das Gupta and Jing Xiao. Non-negative matrix factorization as a feature selection tool for maximum margin classifiers. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2841–2848, June 2011.

[17] David Donoho and Victoria Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems 16*, pages 1141–1148. MIT Press, 2004.

[18] Konstantinos Drakakis, Scott Rickard, Ruair De Frin, and Andrzej Cichocki. Analysis of financial data using non-negative matrix factorization. *Int. Mathematical Forum*, 3(38):1853–1870, 2008.

[19] V. Emiya, R. Badeau, and B. David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *Trans. Audio, Speech and Lang. Proc.*, 18(6):1643–1654, August 2010.

[20] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis. *Neural Comput.*, 21(3):793–830, March 2009.

[21] B. Fuentes, R. Badeau, and G. Richard. Adaptive harmonic time-frequency decomposition of audio using shift-invariant plca. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 401–404, May 2011.

[22] G. Grindlay and D.P.W. Ellis. Transcribing multi-instrument polyphonic music with hierarchical eigeninstruments. *Selected Topics in Signal Processing, IEEE Journal of*, 5(6):1159–1169, Oct 2011.

[23] Sen Jia and Yuntao Qian. Constrained nonnegative matrix factorization for hyperspectral unmixing. *Geoscience and Remote Sensing, IEEE Transactions on*, 47(1):161–173, Jan 2009.

[24] Hyunsoo Kim and Haesun Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM J. Matrix Anal. Appl.*, 30(2):713–730, July 2008.

[25] J. Kim and H. Park. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, 33(6):3261–3281, 2011.

[26] William H. Lawton and Edward A. Sylvestre. Self modeling curve resolution. *Technometrics*, 13(3):pp. 617–633, 1971.

[27] Yann Lecun and Corinna Cortes. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/. Accessed: 2013-05-07.

[28] Cheng-Te Lee, Yi-Hsuan Yang, and H.H. Chen. Multipitch estimation of piano music by exemplar-based sparse representation. *Multimedia, IEEE Transactions on*, 14(3):608–618, June 2012.

[29] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[30] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562. MIT Press, 2000.

[31] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Comput.*, 19(10):2756–2779, October 2007.

[32] K. Ochiai, H. Kameoka, and S. Sagayama. Explicit beat structure modeling for non-negative matrix factorization-based multipitch analysis. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 133–136, March 2012.

[33] K. O'Hanlon, H. Nagano, and M.D. Plumbley. Structured sparsity for automatic music transcription. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 441–444, March 2012.

[34] Andrew J. Oxenham and Sid P. Bacon. Cochlear compression: Perceptual measures and implications for normal and impaired hearing. *Ear and Hearing*, 24(5), 2003.

[35] Pentti Paatero. Least squares formulation of robust non-negative factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 37(1):23 – 35, 1997.

[36] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[37] Mikkel N. Schmidt and Rasmus K. Olsson. Single-channel speech separation using sparse non-negative matrix factorization. In *International Conference on Spoken Language Processing (INTERSPEECH)*, 2006.

[38] P. Smaragdis and J.C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pages 177–180, Oct 2003.

[39] Paris Smaragdis, Bhiksha Raj, and Madhusudana Shashanka. A probabilistic latent variable model for acoustic modeling. In *In Workshop on Advances in Models for Acoustic Processing at NIPS*, 2006.

[40] Suvrit Sra and Inderjit S. Dhillon. Nonnegative matrix approximation: Algorithms and applications. Technical Report TR-06-27, Department of Computer Science, the University of Texas at Austin, June 2006.

[41] I. M. Ulbrich, M. R. Canagaratna, Q. Zhang, D. R. Worsnop, and J. L. Jimenez. Interpretation of organic components from positive matrix factorization of aerosol mass spectrometric data. *Atmospheric Chemistry and Physics*, 9(9):2891–2918, 2009.

[42] E. Vincent, N. Bertin, and R. Badeau. Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription. In *Acoustics, Speech and Signal*

Processing, 2008. ICASSP 2008. IEEE International Conference on, pages 109–112, March 2008.

[43] E. Vincent, N. Bertin, and R. Badeau. Adaptive harmonic spectral decomposition for multiple pitch estimation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(3):528–537, March 2010.

[44] T. Virtanen. Sound source separation using sparse coding with temporal continuity objective. In *Proc. Int. Comput. Music Conf*, pages 231–234, 2003.

[45] T. Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(3):1066–1074, March 2007.

[46] Yu-Xiong Wang and Yu-Jin Zhang. Nonnegative matrix factorization: A comprehensive review. *Knowledge and Data Engineering, IEEE Transactions on*, 25(6):1336–1353, June 2013.

[47] Richard M. Warren. *Auditory Perception*. Cambridge University Press, 2008.

[48] R. L. Wegel and C. E. Lane. The auditory masking of one pure tone by another and its probable relation to the dynamics of the inner ear. *Phys. Rev.*, 23:266–285, Feb 1924.