2014

Learning The Unknown by Masking The Known: Application of Sound Subtraction to Non-Negative Matrix Factorization

Yuexi Ma
Bard College

Learning The Unknown by Masking The Known: Application of Sound Subtraction to Non-Negative Matrix Factorization

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Yuexi Ma

Annandale-on-Hudson, New York
May, 2014

Abstract

When we listen to a recording of people's conversation or a piece of music, our ears can magically tell us who is talking or what musical notes are presented. For machines and computers, this is a much harder problem. In this project, I implemented a technique called non-negative matrix factorization (NMF) that is commonly used for blind signal separation (BSS) problems. This technique first converts audio recordings into time-frequency matrices called spectrograms, and then uses linear algebra to factorize the spectrogram matrices into two other matrices which have products approximate the input spectrograms and contain information about sound features in the original recordings. Besides NMF's ability to extract sound features blindly without any previous knowledge of the recordings, we can also use NMF as a machine learning algorithm and train it on samples before the separation process to improve the accuracy and get additional information about the input channels.

Like many other machine learning algorithms, NMF relies on minimizing errors between the actual recording spectrogram and estimated or simulated recording spectrogram. The final error values before the separation process terminates reflect how much the two factorization matrices can reproduce the original recording, and can thus be used as a way to evaluate the separation quality. Such evaluation, however, does not correlate to evaluation made by human hearing: a "good" NMF separation with low final error produces sound channels that do not sound like the original input channel. This inconsistency inspired me to develop a new method to evaluate the NMF separation by using more information from the training sample, as well as a new extension of NMF separation algorithm. The new algorithm was evaluated on recordings of several musical instruments using the evaluation method I developed. Results suggest that the new method outperforms the original one.

Contents

List of Figures

Dedication

To my memory of Bard.

Acknowledgments

This project would not have been completed without the help and support from my professors and my friends. I would like to thank my advisor Sven Anderson for his great advices and guidance. He always give me a lot of directions and inspirations about this project, and from him I learned about how to conduct a formal research and learn about the state of art. I greatly appreciate Xinran Guan for her love and support. I would also like to thank Professor Ethan Bloch who offered this wonderful latex template that saves me lot of time getting started with writing latex, Amanda Gersten for her help in editing the grammars in my project, and Xing Gao for her playing of harp as sample recordings in this project. Finally, I would like to thank all my families and friends for their love and support.

1
Introduction

Our human hearing system provides us the ability to focus our auditory attention on a particular stimulus while filtering out other stimuli, such as being able to catch a unpredicted mention of our name in a noisy environment, or focus on one specific person's speech while ignoring all others. This phenomenon is called The Cocktail Party Effect. For machines this type of signal separation is a much harder problem.

Over decades of study on the signal separation problem, numerous techniques have been developed to deal with different kinds of problems. However, to date, no algorithm has been built to solve this problem in a general way[?kedar]. Audio source separation without (or with very little) prior information about the source or the process of the source mixing is called Blind Signal Separation (BSS), or Blind Audio Signal Separation (BASS). Furthermore, for audio source separation, the problem is called under-determined when there are fewer recording microphones than the number of actual sources, and over-determined when there are more recording microphones than the number of sources. Independent Component Analysis (ICA) is a among one of the most popular techniques used for prob-

lems that are not under-determined. For under-determined problems, especially 'extremely under-determined' problems when there is only a single channel of mixture, Non-negative Matrix Factorization (NMF) is a very popular technique. Non-negative matrix factorization can also be amended for machine learning[6], in which case the problem is not one of blind signal separation. In such situation, the machine will be trained on samples of possible sources, which serve as prior information for the separation process. There are several advanced extensions of NMF algorithms developed to perform fast and efficient learning in different circumstances, such as an online algorithm which can be continuously trained on unlimited input size[3]. In this project, however, a rather simpler extension of NMF algorithm from [6] is adopted.

The structure of this writing is as follows: Chapter 2 presents some basic knowledge behind an NMF algorithm that needs to be understood. Chapter 3 describes the problem domain in this project, and describes an inconsistency in performance comparison found during the project, which inspired the development of a subtraction method that extends the original NMF method. Chapter 4 proposes a cross-correlation evaluation method and the subtraction method I develop is based on it. Chapter 5 provides evaluation of the algorithms, which suggests that the new algorithm outperformed the original extension of NMF algorithm, which is further discussed in Chapter 6.

2
Concepts

This chapter presents some basic knowledge about non-negative matrix factorization. Section 2.1 explained the basic maths behind matrix factorization; Section 2.2 explains how non-negative matrix factorization can be used to separate sound sources; Section 2.3 describes how non-negative matrix factorization can be used as an machine learning and be able to be pre-trained on a sound source before the separation process.

2.1 Matrix Factorization

The purpose of Matrix Factorization is to decompose a matrix into a product of two or more matrices. Depending on the properties of the matrix concerned, there are several popular techniques of factorization: Singular Value Decomposition (SVD), Eigenvalue Decomposition, QR Decomposition (QR), Lower Upper Decomposition (LU), and Non-Negative Matrix Factorization (NMF). This project focuses on Non-Negative Matrix Factorization. Before going into the more advanced approach of NMF, this section will present a simple

implementation of matrix factorization to show the basic math behind the technique.

Suppose the input is an i × j matrix R. Matrix factorization aims to find two matrices D (i × k) and A (k × j) such that $\hat{R} = DA$, where k is a given integer, denoting the number of what are frequency features that can be extracted from the input data matrix R:

$$R \approx D \times A = \hat{R}$$

In order to obtain D and A, the first step is to create two matrices of size $i \times k$ and $k \times j$ and assign them random values. The next step is to iteratively calculate the difference between their product $\hat{R} = DA$ and the original matrix R, and update the element values in D and A in order to minimize the difference $|R - \hat{R}|$. To find a local minimum of the difference, one takes a step proportional to the negative of the gradient of the cost function during each iteration. The procedure is commonly known as the Gradient Decent[10].

Let r_{ij} denote an element in the matrix R at position (i, j). An item in the estimated matrix $\hat{R}$ will then be:

$$\hat{r}_{ij} = D_i \times A_j = \sum_{l=1}^{k} d_{il} a_{lj}$$

The difference between an item in $\hat{R}$ and the actual item in R can thus be measured by the Euclidean distance[1]:

$$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 = (r_{ij} - \sum_{l=1}^{k} d_{il} a_{lj})^2$$

This difference is also called the *error term*. The error is squared to deal with the fact that the difference between the estimated value $\hat{r}_{ij}$ and data value r_{ij} can be either positive or negative.

To minimize this error, the direction in which the values of d_{ik} and a_{kj} should be updated must be known. This is simply the direction opposite the *gradient*. The following formulas describe how to differentiate the error between each estimated item value r_{ij} and $\hat{r}_{ij}$ with respect to the corresponding two items in the matrix D and A:

$$\frac{\partial}{\partial d_{ik}} e_{ij}^2 = -2(r_{ij} - r_{ij})a_{kj} = -2e_{ij}a_{kj}$$
$$\frac{\partial}{\partial a_{kj}} e_{ij}^2 = -2(r_{ij} - r_{ij})d_{ik} = -2e_{ij}d_{ik}$$

Using the gradients, the update rules for d_{ik} and a_{kj} can be derived:

$$d'_{ik} = d_{ik} + \alpha \frac{\partial}{\partial d_{ik}} e_{ij}^2 = d_{ik} - 2\alpha e_{ij}a_{kj}$$
$$a'_{kj} = a_{kj} + \alpha \frac{\partial}{\partial a_{kj}} e_{ij}^2 = a_{kj} - 2\alpha e_{ij}d_{ik}$$

Here α is a constant value called the *learning rate* that specifies the rate at which the values should be updated toward the desired minimum. In order to avoid ending up oscillating around the minimum, α is typically set to a small value. A rule that can determine when the values of the matrices should stop updating terminates the algorithm. One way to create such a rule is to calculate the total error, also called the *cost*, and have the values stop updating when the cost $E = \sum e_{ij}$ *converges*: The convergence can be defined by a lower bound threshold for the value of the final cost or the rate of change of the cost.

2.2 Non-Negative Matrix Factorization (NMF)

Just as its name suggests, Non-Negative Matrix Factorization is a matrix factorization with the assertion that every element is non-negative. This formulates a model of linearly combined elements for many problems in the real-world.

For sound separation, the input sound signal is first transformed into a spectrogram which is a visual representation of the spectrum of frequencies in a sound[9]. The spectrogram is

a 2D matrix, whose columns are time references and rows representing different frequencies. The item values in the spectrogram represents a the spectra energy of a particular frequency at a certain time location, so the difference in sound volumes does not affect the structure of the spectrogram much, as shows in Figure 2.2.1. Non-negative matrix factorization uses a spectrogram as the input matrix R of dimension $F \times T$ where F stands for frequency and T stands for time. We want to decompose it to a product of two matrices: the basis vectors D $(F \times K)$, also called the *dictionary* matrix, and the weight vectors A $(K \times T)$, also called the *activation* matrix, where K here is the number of sound frequency features we want to extract from the data, so that:

$$R \approx D \times A = \hat{R}$$

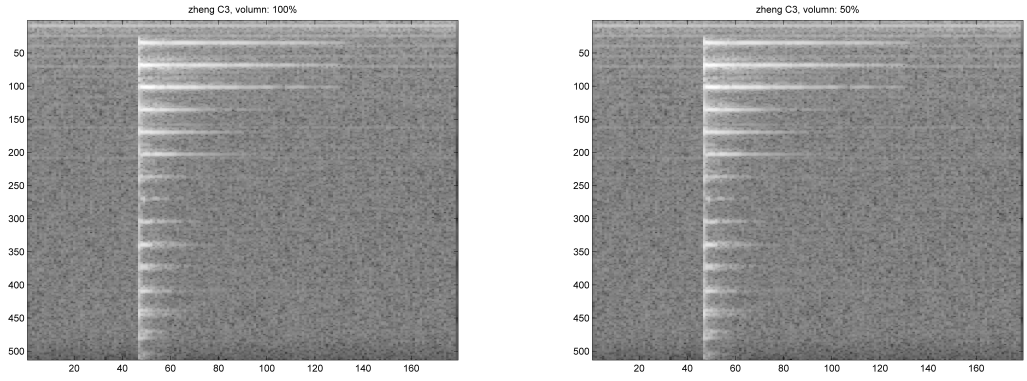

Figure 2.2.1. A note C played on zheng with different adjusted volumes. For the sound with its volume adjusted to 100% and 50% the spectrograms look the same.

Figure 2.2.2 shows an example model of a piano recording of Mary Had a Little Lamb separated by NMF.

Notice that in the figure we use one feature to represent one input source as showed in Figure 2.2.3, that is, we assume one column of frequency features would represents an input source well. For recording of notes played on a same piano, this is usually sufficient

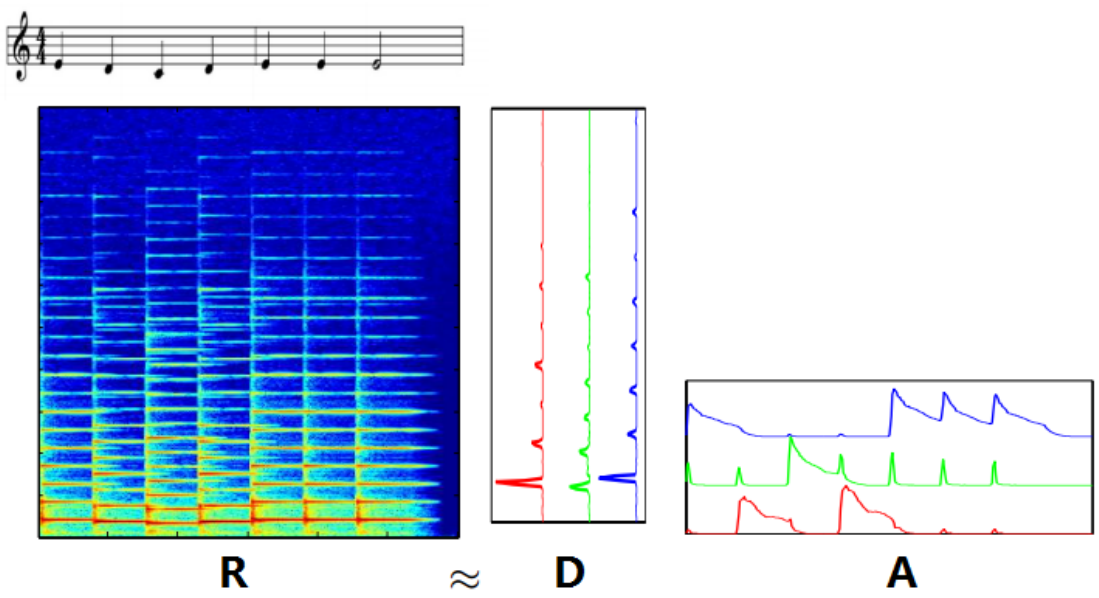

Figure 2.2.2. NMF model of Mary Had a Little Lamb with K = 3. The columns in the dictionary matrix represent frequency features of the three notes presented in the recording and the rows in activation matrix reflect how much each frequency feature happens at certain time. The graph is modified from [6]

since the same note played on a piano makes same sounds every time. Many other times, we use a larger number of features to represent a single input source, i.e. a note that can be played on different pianos or a person's voice.

There is one potential difficulty that we might face if we use more than one feature to represent a input source: we will experience difficulty locating the components of a source, since their column location of the K features in the dictionary matrix D is supposed to be random after separation. In pre-trained NMF problem, this is negligible since we will already have put values of the source features on certain known locations in the dictionary matrix in the beginning.

Section 2.1 explained an basic implementation of matrix factorization in detailed math. The algorithm used for regular non-negative factorization in this projects adopts a complex extension of matrix factorization integrated with many more sophisticated techniques. Since this project intends to add a subtraction step outside the inner part of NMF algorith-

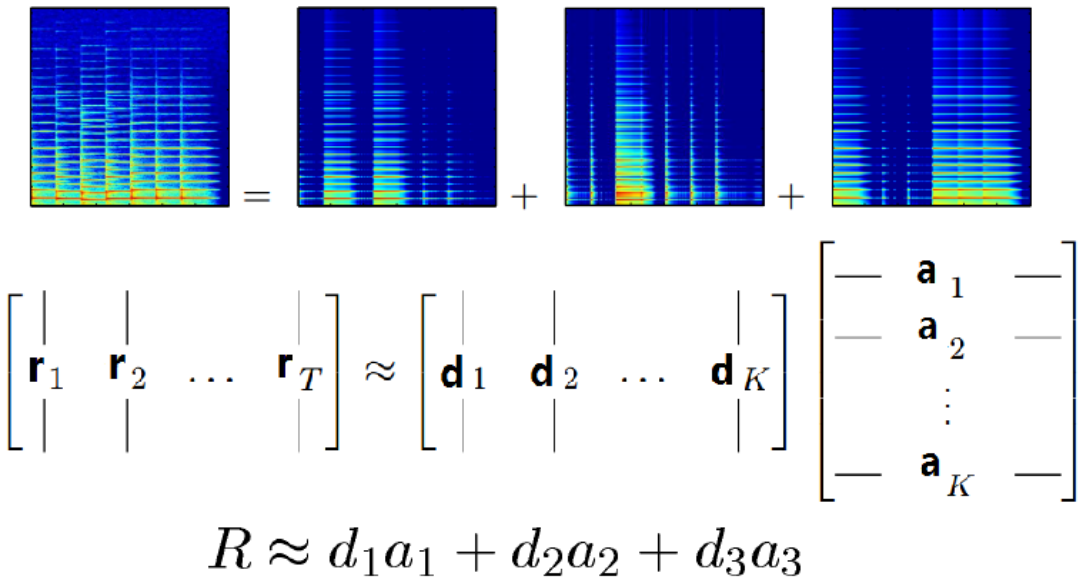

Figure 2.2.3. NMF model of Mary Had a Little Lamb with K = 3. Each frequency feature is an source in this case.

m, the original implementation is not the focus of this project, and he detailed algorithm should be referred from [?nmftoturial]. In fact, the subtraction method proposed in this project should fit to a wide range of different implementations of NMF.

2.3 Training on Non-Negative Matrix Factorization

Non-Negative Matrix factorization can also be used as a machine learning algorithm when pre-training is applied to the algorithm. During the training phase, the algorithm is fed training samples in order to obtain dictionary matrices that contain features of these samples. In the separation phase, the dictionaries are used as part of a larger dictionary for NMF separation.

For example, Figure 2.3.1 the spectrogram of a single note C played on a zheng. In the training phase, with a given feature number Kc, a regular NMF process on the sample gives a dictionary matrix Dc for the note C.

Next, in the separation phase, when initializing the dictionary matrix D, given that the feature number K is larger than Kc, the values in Dc either are used for initialization or are fixed on D. That is, to fix K columns in D with the values from D_c, as shown in Figure 2.3.2.

The algorithm can also be trained on several training samples of different sources to obtain dictionaries $D_1, D_2, ..., D_n$, as long as during the final separation phase $K_1+K_2+...+K_n \leq K$.

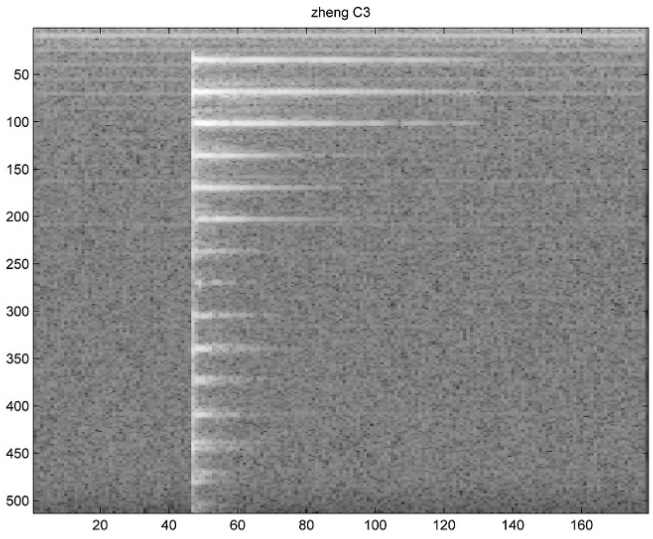

Figure 2.3.1. Spectrogram of a note C played on zheng

Pre-training for NMF provides extra information to the algorithm. This information not only allows the estimated sources to simulate the actual source more precisely, but also provides a clue about the locations of the compositions of a source in the final dictionary and activation matrices. This helps avoid the problem of not knowing the locations of source feature compositions mentioned in the previous section, as illustrated in Figure 2.3.3.

After the K_n columns from the trained dictionaries D_n are integrated into in the separation dictionary D, there are two ways to proceed: 1) fix the values on the K columns (not to update them during separation); and 2) just use values for initialization during the separation phase. Theoretically, both can improve the separation quality, but experiments in this project showed that for the musical instruments selected, simply using the trained dictionary as initialization does not improve the separation accuracy. On the other hand,

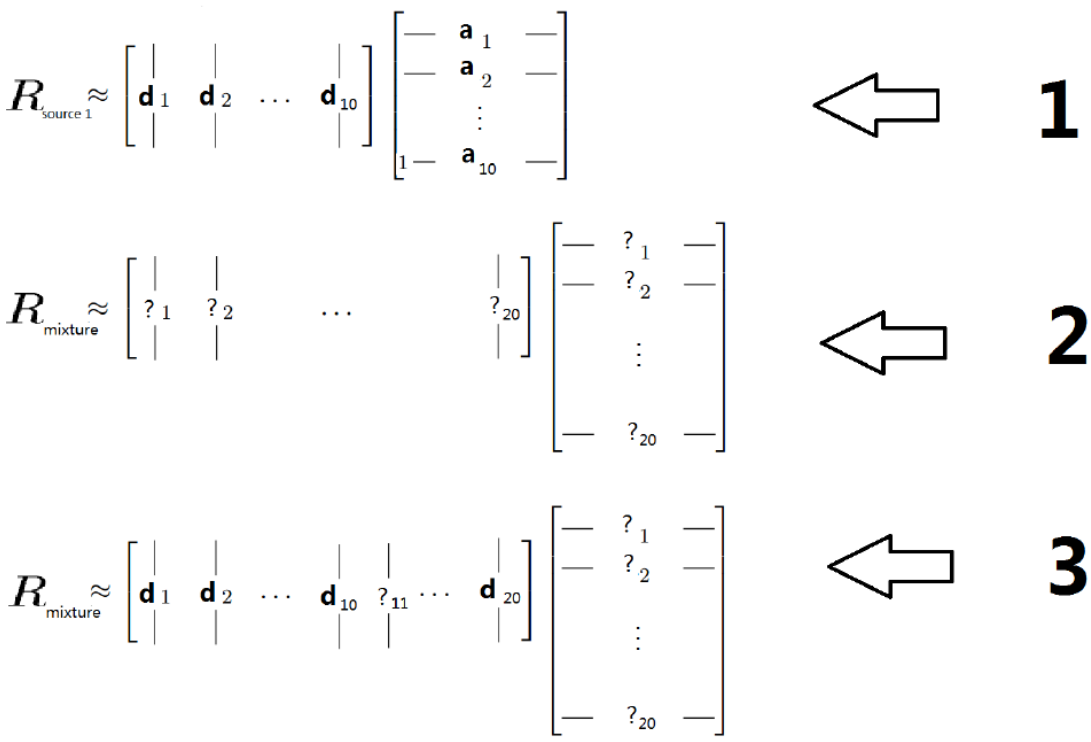

Figure 2.3.2. An NMF separation using a trained dictionary from the training sample. Step 1: run NMF with feature number specified to be 10 to obtain a dictionary consisting of $K_c = 10$ frequency feature columns. Step 2: initialize a larger dictionary with a feature number $K > K_c$. Step 3: initialize or fix the first K_c columns in the separation dictionary with the values from the trained dictionary.

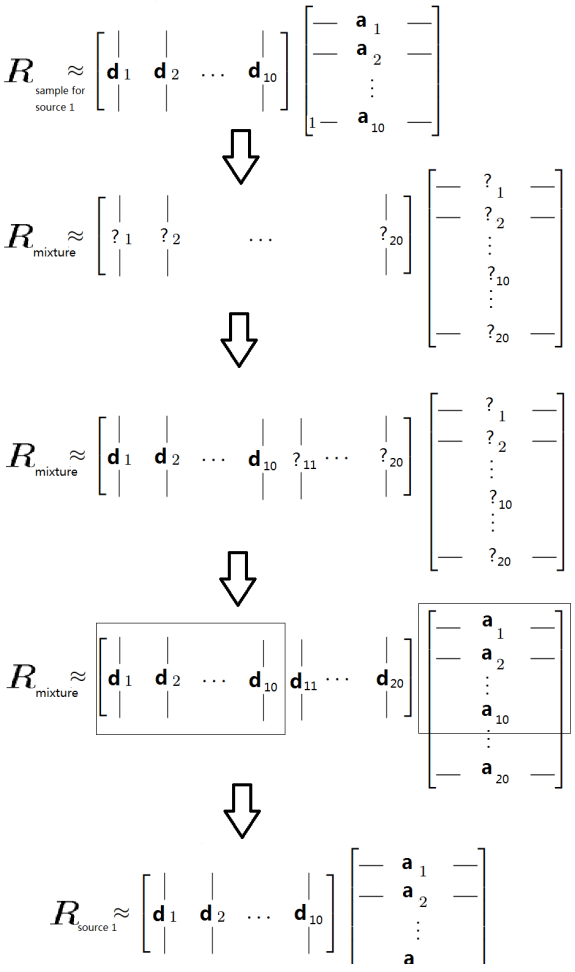

Figure 2.3.3. An NMF separation trained on the training sample of a source will know the feature locations (boxed) of the source in the output matrices D and A.

method 2 still provides clues about the locations of source feature compositions.

3
Experiments with Signal Separation Algorithms

This chapter developes and describes the signal separatio algorithms studied in this project. Section 1 describes the problem domain and lists the musical instruments as well as other environments used. Section 2 refines the algorithm specified in the last section in Chapter 2 and presents the more complicated algorithm of matrix fractionation used in this project. The last section explains an inconsistency between the two NMF separation evaluations found during the project. This inconsistency led to the development of a new extension of the NMF algorithm which is presented in Chapter 4.

3.1 Description of Problem Domain

Music instruments usually produce sound spectrograms with clear structures and the spectrograms are stable among different plays of a same note. Furthermore, many instruments produces spectrogram-rich sounds (i.e. sounds that have many different frequency bands on their spectrograms), and can simulate interesting frequency-overlapping problems such as overtone [`?overtone`]. Therefore, it is reasonable for an simple sound separation algorithm to work with recordings of musical instruments before working with more com-

plicated sounds such as human voices. This project also uses a lot of software generated instruments. The following is a list of all musical instruments used in this project:

1. real playing harp, which will be referred as harpR later.
2. real playing piano, which will be referred as pianoR later.
3. software generated piano sounds, which will be referred as pianoS later.
4. software generated violin notes pizzicato, which will be referred as violinS later.
5. software generated guitar sounds, which will be referred as guitarS later.
6. software generated zheng sounds, which will be referred as zhengS later.

All the examples in this project use recordings of software generated zheng, while in the evaluation chapter, all the instruments are used. Recordings from these instruments in this project are all lasting sounds, i.e. they disappear only by fading away naturally, and they are also all non-sliding sounds, i.e. every single note playing keeps a stable frequency.

The mixture of notes in this project all consists of two notes. During evaluation, the mixture is produced by adding two separately recorded notes together through software to simulate a real mixture of two notes playing together. The process of mixing will be explained in Chapter 5.
The algorithms in this project are implemented in Matlab, and run on an Intel I7-3820 4-core @2.7Hz processor.

3.2 Error scores differ with ear judgments

The NMF algorithm I used terminates when it determines a convergence: when the rate of change of the overall error, or the *cost* E, is than 0.01 percent. The cost E measures

the difference between the estimated input data $\hat{R}$ and the actual recording data R. Thus, a lower cost E indicates that the estimated matrix $\hat{R}$ is similar to the input matrix R, including a better the separation of the audio sources.

Based on cost minimization, when comparing two NMF separation experiments on the same audio mixture, the one with lower final overall error should model the mixture of the sources better. In other words, the one with local final overall error will estimate audio sources that sound more like the actual audio sources.

To verify this hypothesis, I conducted the following three NMF separation experiments on a recording of a note C_3 and a note D_3 playing simultaneously on the software zheng instrument. In the experiments, a features number 10 is used during training phase, and a features number 20 is used during separation phase.

Trial A: trained on note C; fixed values

> Train the algorithm on note C with a feature number of 10 and created dictionary D_c with 10 feature columns.
>
> During separation process the dictionary D was constructed with 20 columns.
>
> The first 10 columns of D were *fixed* with values from D_c.
>
> The second 10 columns of D was to be updated during the separation process.

Trial B: *trained on both notes C and D; initialized values*

> Trained on note C with a feature number of 10 and created dictionary D_c with 10 columns.
>
> Then trained in note D with a feature number of 10 and created dictionary D_d with

10 columns.

During separation process the dictionary D was constructed with 20 columns.

The first 10 columns of D were *initialized* with values from D_c, and the second 10 columns were *initialized* with values from D_d.

The dictionary D was to be updated during the separation process.

Trial C: *trained on both note C and D; fix values*

Trained in note C with a feature number of 10 and created dictionary D_c with 10 columns.

Then trained in note D with a feature number of 10 and created dictionary D_d with 10 columns.

During separation process the dictionary D was constructed with 20 columns.

The first 10 columns of D were *fixed* with values from D_c, and the second 10 columns were *fixed* with values from D_d.

The dictionary D was *not* to be updated during the separation process.

Our instinct might tell us that trial C should yield a better result than trial A because C was trained on both notes while A was trained after pre-training on one note. Both A and C had the dictionary elements from training fixed. As for trial B it is not so clear whether it should have better or worse separation than A or C without conducting an experiment.

The following table shows the comparison of the two evaluation methods. The NMF cost values are averaged from 30 runs.

	hearing evaluation	final *cost* of NMF
Trial A	bad	0.0009 (best)
Trial B	bad	0.0011
Trail C	good	0.0014 (worst)

As shown above, the first comparison suggested trial C yielded the best result but the NMF error comparison suggests it to be the worst among the three trials. This seems to be a contradiction. If we assume that listening evaluation is the good standard, the NMF error scores have limited Value.

The reason that the two comparison methods give different results is that they are comparing different things: the cost function compares the mixtures while ear evaluation compares the sources. This will be further explained in Section 4.1

In this project, we are dealing with a simpler problem, compared to problems in real life, assuming that each note is a lasting note. If the timestamps match, the spectrogram of a ideally separated source should be in the same with the training sample of that note after normalization. Is there a way to compare the separated audio channel with the initial recording, if available, used to train the algorithm?

If we have training samples of the possible sound sources, we might want to not only use the trained dictionary matrix, but also the trained activation matrix to perform the separation. This is explored in the next chapter.

4
Extensions of NMF-based Signal Separation

This chapter describes extensions of the NMF algorithm. As explained in the previous chapter, the regular NMF cost function does not correctly evaluate its separation accuracy. In this chapter, a new comparison method that will correlate better to human hearing evaluation is proposed in Section 1. Section 2 presents a new extension of NMF algorithm based on sound subtraction and the new comparison method.

4.1 A cross-correlation comparison method

For a mixture $R = Source_1 + Source_2$, the original NMF cost function measures the difference between the estimated mixture matrix $\hat{R}$ and actual mixture R in $R \approx D \times A = R$. The separation creates estimation of sources S_1 and S_2. The evaluation performed by hearing mentioned in the previous chapter, compares the estimated source S_1 and S_2 with the actual source $Source_1$ and $Source_2$. A regular, untrained NMF algorithm does not have information about the actual source, but the pre-trained NMF will have the desired

information, i.e. a training sample of that source.

As mentioned previously, there are several ways to compare the two matrices: one of the most popular general approaches is the Euclidean method, and one of the most popular approaches for spectrogram comparison is the Itakura-Saito divergence[2] "which allows to capture fine structure in the power spectrum"[4]. However, to compare an estimated source matrix with a training sample, *time shift* becomes an additional problem. Time shift occurs when a sound occurs at different times on the two spectrograms. A desired comparison measures the matrix similarity at particular locations on the matrices where the sound occurs.

Therefore, comparison of the estimated and actual source consists of two steps: 1) find the time shift; and 2) compute the difference. There are several approaches that can efficiently perform these two steps. An ideal method such as [5] involves dynamic programming and can do both steps at the same time. Due to the limited time, this project uses a simpler approach containing the above two steps.

4.1.1 Step 1: Find the time shift

The following figure shows the *time shift* between the two matrices of the estimated source note C and the training sample of that note in Trial A in Section 3.2.

The arrow points to the location where the note C occurs in both spectrograms. By looking at the graphs, one can tell that the note C happens at around column 40 in the estimated source spectrogram and at 20 in the training sample. Actually, this method of looking at the figure finds the peaks of sound energy in both spectrograms. The same thing

can be done algorithmically by converting the spectrogram into a sound energy graph and then looking for the peak.

Although this peak-of-energy-finding method does not seem robust enough to detect the correct time of occurrence of a sound in circumstances with large separation errors, the effect will be mitigated by the method proposed in Step 2. Additionally, the time location found using this method can be further adjusted by the indicator matrix shift method that will be presented in the next section.

4.1.2 Step 2: Compute difference

Step 1 finds where in the matrices the sound occurs. Since the only purpose is to address whether or not a particular sound in the training samples occurs in the mixture, there is no need to compare the difference in sound energy in this step. Thus, the matrices should be normalized before calculating the difference.

Furthermore, the two spectrograms consist of exactly the same dictionary matrix fixed in the separation process, so rather than comparing the spectrograms, it will be sufficient to compare only the activation matrices, which are the extra information taken from the training phase.

Since the activation matrix indicates when and how much each of the frequency features occur, it accurately reflects the component structures of the sounds. Given the time locations, or column positions derived from Step 1, it is only *roughly* known where the sound occurs in both matrices, since the method in Step 1 may not give an accurate estimate of the column positions. The next step is to take 3 columns before the column

positions and 6 columns after the column positions from both matrices to construct two 10-column matrices. Each of these matrices should be normalized by dividing all elements by their maximum values. The last step is to calculate the sum of the dot products of all the possible two-column combinations in the selected matrices. This method will now be referred to as the *cross-correlation method*.

The *cross-correlation method* serves as a weighted comparison according to the occurrence of each frequency feature. Consequently, the method gives more weight to the center of the selected parts of the activation matrices (since they are around the volume peaks) than the edges of the matrices. Thus, error in the energy peak-finding method in Step 1 is mitigated by the *cross-correlation method*.

Note that the parameters chosen for the number of columns to select before and after the peak volume positions may not be an optimal choice. The task of finding optimal parameters is left to future research.

The three types of comparison are shown below.

Comparison#	Trial A	Trial B	Trial C
Evaluation by hearing	worst	medium	best
NMF final cost	best	medium	worst
Cross-correlation	4.1×10^4 (worst)	9.6×10^4 (medium)	2.1×10^5 (best)

The table above shows that the cross-correlation method agrees more closely with hearing evaluation. Together with a threshold value such as 5×10^3, the new comparison method will be able to determine whether a certain note exists in the mixture. The significance of this method is not, however, what measurements to take given the sound occurrence locations. The dot product sums method can be replaced with a Euclidean or an Itakura-Saito method. The key of this method is how to find the sound occurrence locations, and this

method is further developed with an *indicator matrix shift verification method* in Section 4.2.1.

4.2 A subtraction method in non-negative matrix factorization

The previous chapter showed that the basic NMF separation failed to separate two music notes C and D playing simultaneously, even though it had been pre-trained on the note C. On the other hand, the previous section explained how to determine that a pre-trained note exists in the mixture even if the separation itself is not successful, and provides a method for determining the location of sound occurrence in a recording. The comparison table in the previous chapter also showed that the algorithm works best when it has been trained on both notes and has the dictionary values fixed. All of these suggest that the algorithm might be improved by an extension that performs a subtraction of training samples from the mixture in order to get a pseudo-training sample of the sound elements for which we do not have training samples.

The new *cross-correlation method* can determine that note C exists in the mixture at a certain time location in the mixture. Given a recording of notes C and D, if the sound of note C is subtracted, the remaining note will be D. This might serve as a pseudo-training sample for that note. An extension of NMF separation could be based on first extracting a note and then subtracting it from the source, thereby leaving the remaining note or notes. A fixed dictionary could then be used to perform a final separation.

In order to perform the subtraction, there are two problems must be solved:

- The time location found by peak energy search might be inaccurate.

- The training sample of the sound usually has a different energy from that of the sound when it is part of a mixture.

The following two steps address these two problems.

4.2.1 Step 1: Indicator matrix shift

The purpose of this step is to adjust the time location found by the energy-peak method. This step uses small shifts of the matrices to find an optimal time displacement.
To find the optimal displacement d with which to match matrix R_1 and matrix R_2:

1. Normalize the matrix R_1 to R_1' using the maximum element value max_{R_1} found in R_1.

2. Normalize R_2 to R_2' using max_{R_2} found in R_2.

3. Calculate the mean values $\mu_{R_1'}$ and $\mu_{R_2'}$ of the matrices R_1' and R_2'.

4. Convert the matrices into indicator matrices $I_{R_1'}$ and $I_{R_2'}$ consisting of value 1 and 0's where 1 indicates an matrix element larger than its mean value, and 0 indicates an matrix element smaller than its mean.

5. Perform an element-wise *AND* operation on the two matrices and calculate the sum of the result $\sum\{I_{R_1'} \& \ I_{R_2'}\}$.

6. Starting from the time locations found by the peak-of-volume method, shift the columns of matrix $I_{R_1'}$ to the left by 1 column and get a shifted matrix $I'_{R_1'}$. Then, perform element-wise *AND* operation again on the shifted matrix $I'_{R_1'}$ and the other matrix $I_{R_2'}$, and check whether the sum $\sum\{I'_{R_1'} \& \ I_{R_2'}\}$ is larger than $\sum\{I_{R_1'} \& \ I_{R_2'}\}$.

7. Repeat the last step but shift the columns of matrix $I_{R_1'}$ to the right instead.

8. Repeat step 6 and 7 but shift the columns by n units instead of 1, where n increases from 1 to a larger number by a step size of 1 each time, remembering what number of shifts n, as well as the shifting direction (positive for right and negative for left), that yields a maximum value of $\sum\{I_{R_1'} \& \ I_{R_2'}\}$.

9. Finally, if n is not zero, add the value to the time shift value derived from the peak-of-volume method previously, and apply it to R_1 to get R_{1s} .

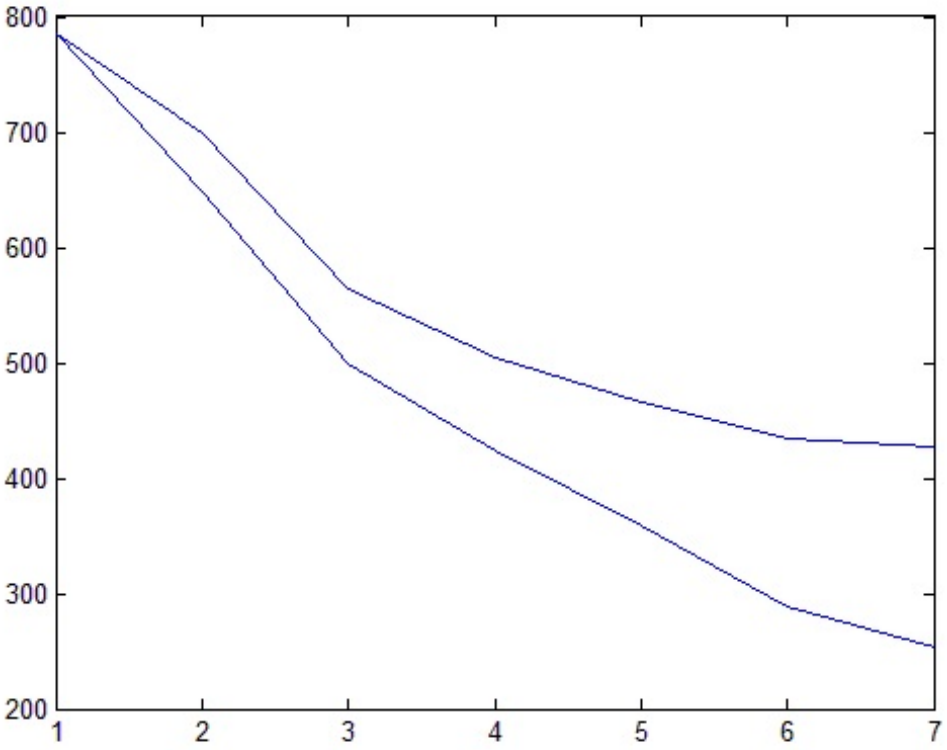

Figure 4.2.1. The values of indicator matrices of the training sample and the mixture during a shifting search for a maximum value. The values are the sum of the overlapping part of the matrices calculated by *AND* operation on the two matrices. The higher and lower lines on this graph are values of the sum when the matrix shift right and left respectively by the amount of x value in the graph. The highest point of both lines being at $x = 1$ indicates that the sound occurrence locations found by this method agrees with the peak-of-energy method.

4.2.2 Step 2: Energy matching

The *indicator matrix method* solves the time shift problem. Step 2 now solves the volume difference problem. Intuitively, given the time-shift-adjusted training matrix R_{1s} and the mixture R_2, the aim is to find a multiple β so that $\sum |\beta * R_{1s} - R_2|$ has a minimum value, with the restriction that *all elements are still non-negative.*

In practice, however, with the presence of noise, a strict non-negative restriction will terminate the process earlier than it should be terminated. Instead, it is better to remove the non-negative element restriction, and search for the minimum total number of elements in the subtracted matrix that have an absolute value larger than the mean element value μ of the original matrix R_2:

- Find β so that $\sum(|R - |\beta * R_{training}|| > \mu)$ reaches its minimum.

One approach to find β is to perform a linear search, starting with a small value of β and increasing it incrementally, meanwhile checking for a minimum value of the sum $\sum |\beta * R_{1s} - R_2|$. In this way, the search is limited by the size of β.

Figure 4.2.2 shows the value of sums $\sum(|R - |\beta * R_{training}|| > \mu)$ obtained by a linear search:

The graph shows that the sum falls to the minimum and the rises monotonically. This suggests that a search similar to a binary search for the minimum value will improve the efficiency from $O(n)$ to $O(\lg n)$ time. Such a search compares two or more adjacent β values each time in search for a value of beta before which the sum is falling and after which the sum is rising. The binary search should terminates when it runs out of choices. Figure 4.2.3 illustrates this idea. Although this method improves the speed of the search,

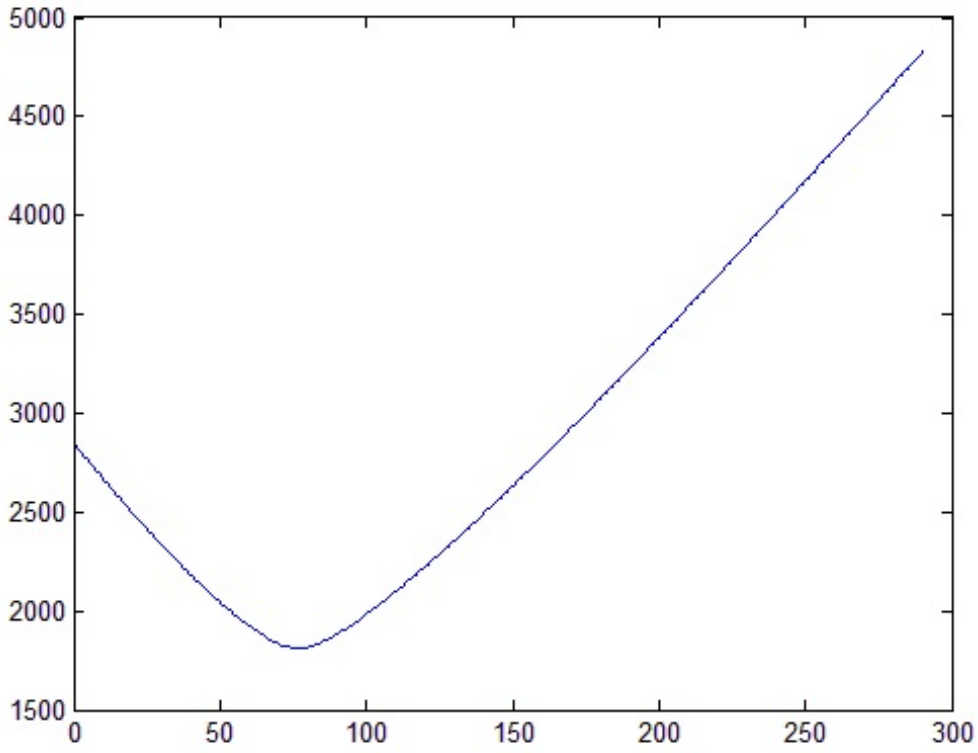

Figure 4.2.2. The values of $\sum(|R - |\beta * R_{training}|| > \mu)$ during a linear search for its minimum by trying 300 different values of β increasing from 0.1 to 3.0 with a step size of 0.01. The graph shows when $\beta = 87 \times 0.01 = 0.87$ the above equation reaches its minimum value.

it gives the same result as a linear search.

Figure 4.2.4 shows the process of subtraction during a linear search.

The matrix resulting from subtraction can then be used as a pseudo training sample of note D along with some noise. Additional NMF training on the pseudo training sample will produce an extra dictionary that if fixed with the dictionary derived from the original training sample, can be used in the final NMF separation .

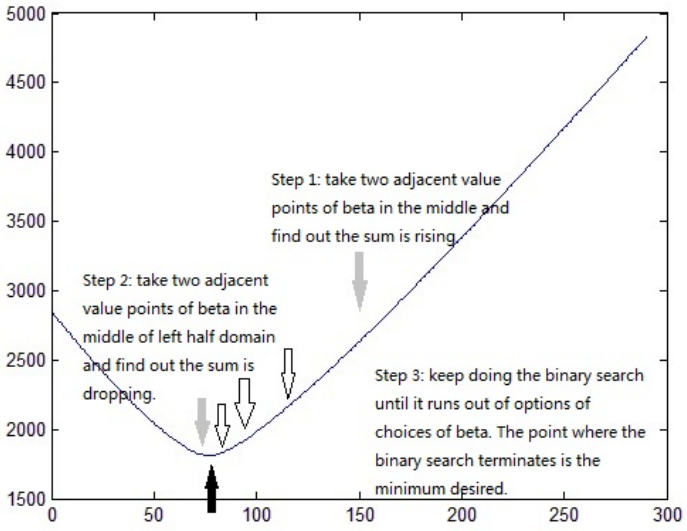

Figure 4.2.3. The process of a possible binary search that finds the minimum of $\sum(|R - |\beta * R_{training}|| > \mu)$ for the subtraction method by looking for the lower bound in the graph.

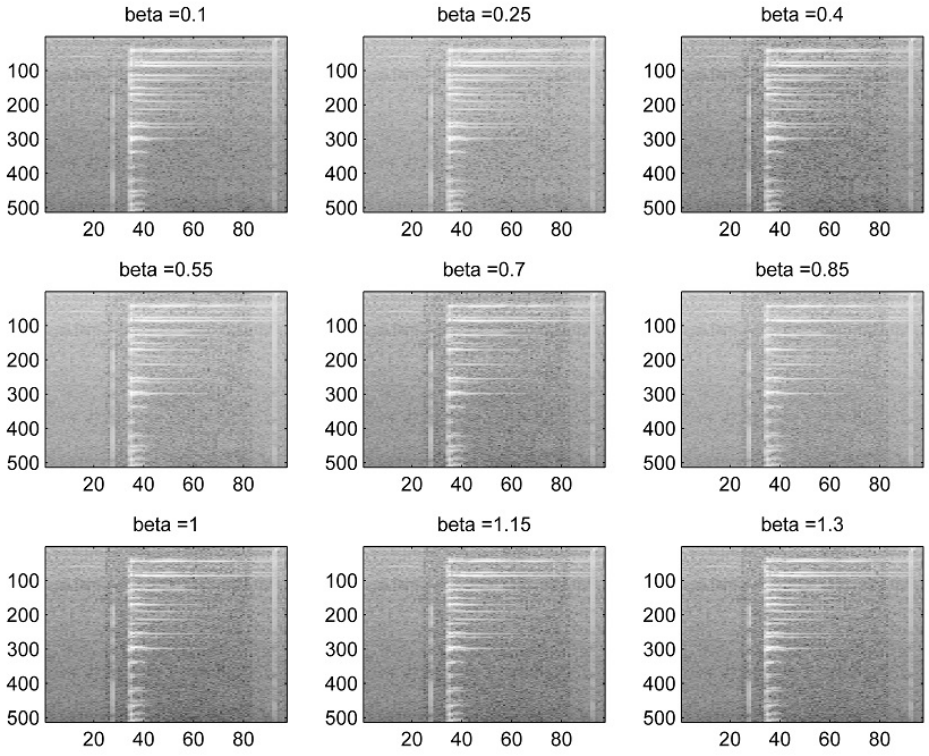

Figure 4.2.4. The process searching linearly for a minimum value of $\sum(|R-|\beta * R_{training}|| > \mu)$ by trying different values of β increasing from 0.1 to 3.0 with a step size of 0.15. The frequency bands of note C faded away when β reaches around 0.85, and then starts reappear as the values in spectrogram becomes negative.

5
[final, frozen]Results: Evaluating Signal Separation

This chapter describes evaluation of the algorithms proposed in Chapter 4. The algorithms were tested with recordings of the six musical instruments listed on Section 3.1. During each test, recordings of two notes were selected and added to simulate simultaneous play of the two notes. More realistic stimuli in which some frequency spectra overlap were also evaluated.

Stimuli with overlapping spectra are more difficult to separate, This section contains evaluations of the algorithm on the following categories of cases: 1) with few overlapping frequencies; 2) with some visible overlapping; and 3) spectra mostly overlapped. In each case, three methods of NMF separation are performed on different combinations of notes recorded from six musical instruments, and the separated signals are evaluated by eight comparison methods.

The three methods used to separate a mixture are: 1) regular NMF trained on one note; 2) regular NMF trained on two notes; and 3) the extension of regular NMF (with the

subtraction method proposed) trained on one note. The purpose of this evaluation is to show whether Method 3 outperforms Method 1. Method 2 is used as a reference for the performance since it trains NMF on both source notes and thus has an optimal record of stimuli spectral content.

The eight comparison methods mentioned above are described in the following table:

#	Name	Description
1	Euclidean cost	The final Euclidean distance calculated after the updating process in the NMF separation. This method is also used to construct the updating rule for NMF as explained in Section **??**. It compares the simulated input $\hat{R}$ with the actual input mixture R.
2	Itakura-Saito cost	The final cost measured by Itakura-Saito divergence after the NMF separation complete. It compares the simulated input $\hat{R}$ with the actual input mixture R.
3	cross-correlation with dot products	The cross-correlation comparison measured by summing dot products of columns around the sound occurrence locations. It compares the estimated source $\hat{S}_1$ with the actual corresponding source S_1. In real problems without information about the actual source, this method will compare $\hat{S}_1$ with the corresponding training sample.
4	cross-correlation with Euclidean divergence 1	The cross-correlation comparison measured by first aligning the matrix columns according to the sound occurrence locations and then calculate the Euclidean divergence between them. It compares the estimated source $\hat{S}_1$ with the actual corresponding source S_1. In real problems without information about the actual source, this method will compare $\hat{S}_1$ with the corresponding training sample.
5	cross-correlation with Euclidean divergence 2	Same with method 4, but instead compares the other estimated source $\hat{S}_2$ with the actual source S_2. Such comparison is only available during the evaluation when the input sources are all known.
6	energy ratio 1	Spectrogram energy ratio of estimated source $\hat{S}_1$ and the actual corresponding source S_1: $E(\hat{S}_1)/E(S_1) - 1$.
7	energy ratio 2	Spectrogram energy ratio of estimated source $\hat{S}_2$ and the actual corresponding source S_2: $E(\hat{S}_2)/E(S_2) - 1$.
8	energy ratio	The ratio of the estimated sources spectrograms energy ratio and the actual source spectrograms energy ratio: $(E(\hat{S}_1)/E(\hat{S}_2))/(E(S_1)/E(S_2)) - 1$.

Table of comparison method during evaluation

Comparison 4 and 5 directly compare the estimated sources with the corresponding actual input sources. This method resembles comparison performed by human ear most, but it omits the spectrogram energy difference comparison. Comparison 6, 7, and 8, on the other hand, provide measurements of the energy ratios.

5.1 Evaluating the cross-correlation comparison method

Before going to use the comparisons method listed in this chapter to compare the NMF algorithms, these comparison methods first need to be evaluated. Since the ultimate goal is to produce separation result that can be appreciated by ear judgment, a good way to evaluate comparison methods is to compare them with a hearing evaluation mentioned in Section 3.2.

During this evaluation process, at each time two notes from an instrument are randomly selected to create a simulated mixture. Then the three NMF methods listed previously separate the sources. After the separation ends, each estimated source and input source are played through the speaker of computer for a person to hear and judge. Ten comparison are performed and the person participated decided that the following relationship between the three separation methods:

NMF method	separating the trained source	separating the untrained source
Method 1	bad	bad
Method 2	good	good
Method 3	good	good but slightly less good than Method 2

Table 1: ear judgment for different NMF methods

At last, the same ten samples are compared by the seven comparison listed previously. The detail of the comparison results will be shown in the next section, and here is a summary about the comparison results:

Table 1: Human judgments for different NMF methods

Finally, the same ten samples are compared by the seven comparison listed previously. The detail of the comparison results will be shown in the next section. Here is a summary of the comparison results:

Comparison method	best method for separating the trained source	best method separating the untrained source
Comparison 1	Method 1 (not significantly)	Method 1 (not significantly)
Comparison 2	Method 1	Method 1
Comparison 3	Method 2 and 3	Method 2 and 3
Comparison 4	Method 2 and 3	Method 2 and 3
Comparison 5	Method 2 and 3	Method 2 and 3, 2 slightly better than 3 (not significantly)
Comparison 6	no significant results	no significant results
Comparison 7	no significant results	no significant results

Table 2: Results of different comparison methods performed to the samples used in ear judgment evaluation

By comparing table 1 and table 2, it suggests that Comparison 1 and 2 is not consistent with ear judgment, which agrees with Section 3.2, and Comparison 3, 4 and 5 is consistent with ear judgment.

5.2 Evaluating the extension of non-negative matrix factorization with a subtraction method

Chapter 4 and Chapter 3 used the example of notes C and D playing simultaneously by hand on zheng. During the evaluations, however, separate recordings of two single notes are selected and added up to obtain a simulated recording of the two notes playing together. The mixing can be performed by first using the peak-of-energy method proposed in Section 4.1.1 to find sound occurrence location in both sources and then adds them together after aligning them. At last, a little random magnification factor is applied to the sound volume. This addition not only provides a convenient way to simulate different mixtures, but also provides the source recordings within the mixture for more accurate evaluation.

In the following three subtractions, automatically generated mixture samples using the above addition method are evaluated against the three cases regarding different level of spectra overlapping.

5.2.1 Case 1: few frequencies overlapping

Figure 5.2.1 and 5.2.2 show the spectrograms of the note C and D played on a software zheng and a real piano and their simulated mixture. The graphs shows few visible overlapping frequency bands from the two sources in the graph on the right side.

During the evaluation, each of the three NMF methods is performed 40 times on mixtures

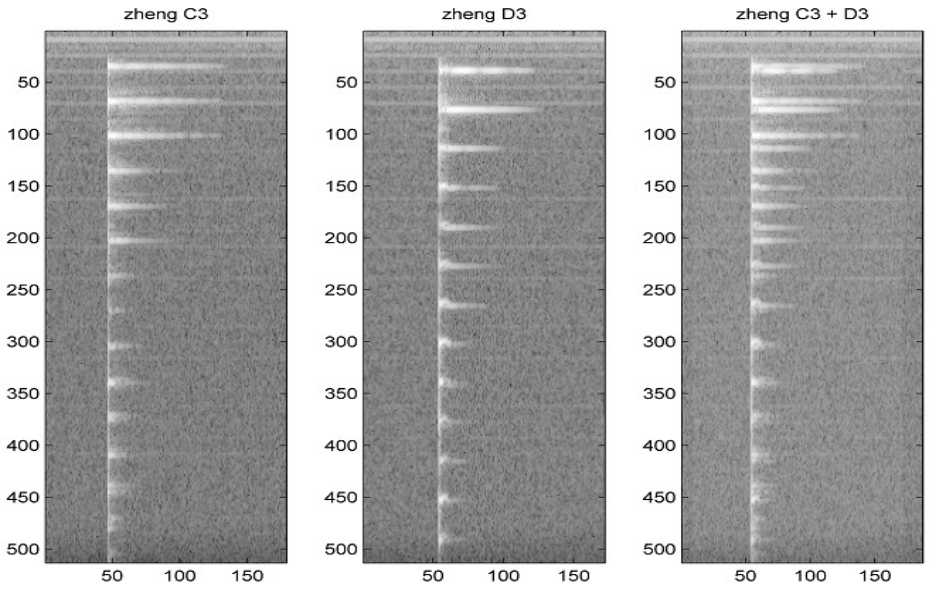

Figure 5.2.1. Simulation of note C and D playing simultaneously on a zheng. The X-axis represents time references and Y-axis represents frequencies. The *peak-of-energy method* finds the locations of the sound occurrences in both spectrogram and then a addition of the two spectrograms produces the simulated mixture after aligning the two matrices.

of note C and D, totaling 720 separations. The following table shows the averaged results for each instrument:

	1	2	3	4	5	6	7
pianoR	0.0002	0.1487	160.0574	0.0054	0.0034	-0.4505	-0.0012
harpR	0.0004	0.1512	2496.4127	0.0548	0.065	-0.381	0.009
pianoS	0.0011	0.1404	2640.4665	0.2762	0.1556	-0.6425	0.6358
zhengS	0.0014	0.1438	5948.7292	0.1656	0.1559	-0.5399	0.2476
guitarS	0.0013	0.1388	5177.0325	0.1273	0.1047	-0.5369	0.1877
violinS	0.0013	0.1451	8932.7014	0.0689	0.0466	-0.3878	-0.0177
All	0.0009	0.1447	4215.7914	0.1157	0.0884	-0.489	0.1748

Table 1.1: Evaluation of the regular NMF separation trained on one of the two notes

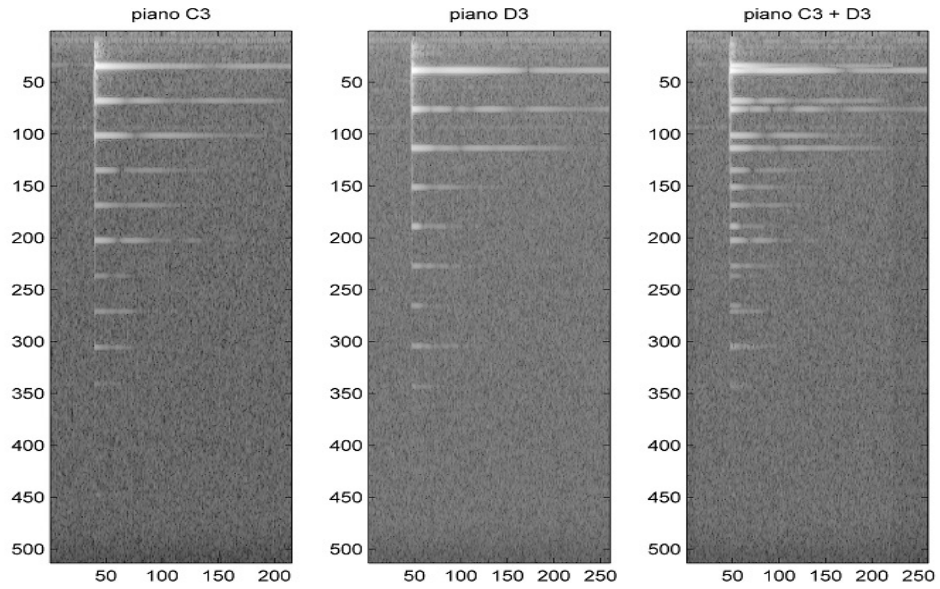

Figure 5.2.2. Spectrograms of note C (left) and D(middle) recorded from a piano. The graph on the right is the simulated mixture of the two notes.

	1	2	3	4	5	6	7
pianoR	0.0003	0.1538	358.0945	0.0014	0.0006	-0.1885	-0.1854
harpR	0.0006	0.1564	4927.5195	0.0038	0.0161	-0.1413	-0.1592
pianoS	0.0015	0.1534	15704.3145	0.0067	0.0009	-0.1753	-0.1492
zhengS	0.0019	0.1512	13994.5218	0.009	0.005	-0.1635	-0.139
guitarS	0.0021	0.1447	11607.6026	0.0099	0.0078	-0.1638	-0.1157
violinS	0.0018	0.1588	12694.4722	0.0113	0.0106	-0.1795	-0.1966
All	0.0014	0.153	9839.1777	0.007	0.007	-0.1682	-0.157

Table 1.2: Evaluation of the regular NMF separation trained on both notes

	1	2	3	4	5	6	7
pianoR	0.0002	0.1515	402.8204	0.0008	0.0022	-0.1282	-0.2279
harpR	0.0004	0.1522	5531.1413	0.0005	0.036	-0.0816	-0.2012
pianoS	0.0013	0.1434	15666.7192	0.0055	0.002	-0.1617	-0.1721
zhengS	0.0016	0.1467	13935.2905	0.0046	0.0144	-0.1087	-0.1954
guitarS	0.0015	0.1405	12817.1356	0.0027	0.0471	-0.12	-0.1513
violinS	0.0014	0.1487	13730.4936	0.0044	0.029	-0.1535	-0.219
All	0.001	0.1472	10316.7959	0.003	0.0223	-0.125	-0.194

Table 1.3: Evaluation of the NMF separation extended with subtraction method trained on one of the two notes

To analyze the results with more information including confidence intervals, Figures 5.2.1-5.2.1 give seven box plots of the seven comparisons performed on the three NMF separation methods.

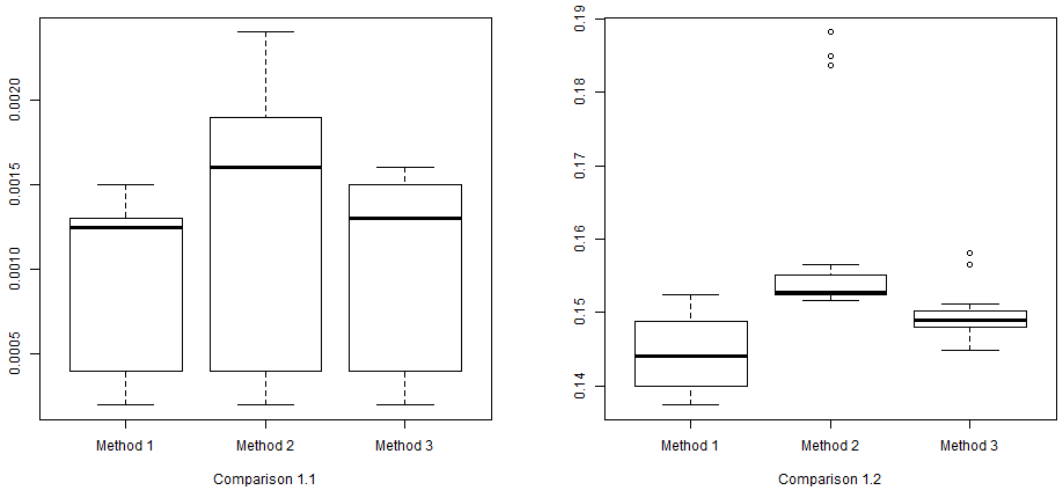

Figure 5.2.3. Comparison of final cost measured by Euclidean distance (left) and Itakura-Satori distance (right) on simulated input $\hat{R}$ with and actual input mixture R. Lower is better. The confidences intervals of the Euclidean costs of three comparisons largely overlap, suggesting that this comparison does not give significant result.

In Figure 5.2.1 Comparison 1.1 on the left shows that the Euclidean divergence of simulated input and the actual input mixtures is low for Method 1 and higher for Method 2 and 3, but not significantly. Comparison 1.2 measured with Itakura-Saito divergence shows Method 1 is most accurate and Method 2 least accurate. This does not agree with the evaluations (Comparison 1.4 and 1.5 in Figure **??**) performed to compare the estimated sources directly with the actual input sources. This inconsistency agrees with the inconsistency found in Section 3.2.

Figure 5.2.4 shows the cross-correlation comparison method with dot product sums pro-

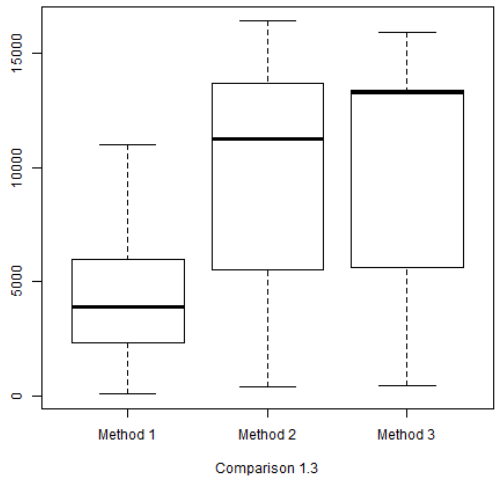

Figure 5.2.4. Cross-correlation comparison with dot product sums on the estimated source $\hat{S}_1$ and the actual corresponding source S_1. Higher is better.

posed in Section 4.1. It suggests that Methods 2 and 3 separate the first source much better than Method 1. This is consistent with Comparisons 1.4 and 1.5. It also suggests that Method 3 slightly outperformed Method 2 on separating the first note.

Figure 5.2.5 shows that the Euclidean divergence comparisons between the estimated sources and the actual input sources suggest Method 2 and Method 3 separate the first source more accurately than Method 1. They also suggest that Method 2 outperforms Method 3 slightly in separating the second source, but not significantly. It also means that the cross-correlation comparison method with dot product sums proposed in Section 4.1 is consistent with these comparisons.

Figure 5.2.1 shows that the first estimated source separated by Method 1 is very different from the actual input source considering spectra energy. This suggest that Method 1 is not very accurate and its separated source 1 may contain part of sounds that is originally

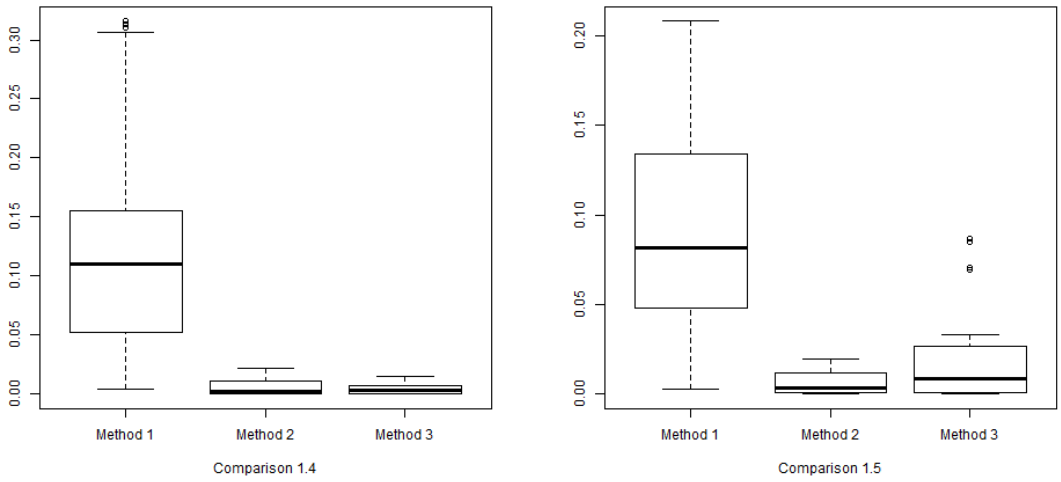

Figure 5.2.5. Cross-correlation comparison with Euclidean divergence on the estimated source $\hat{S}_1, \hat{S}_2$ and the actual corresponding source S_1, S_2. Lower is better.

presented in the second input source. The figure also shows that Method 2 has spectra energy ratio of source 1 further than zero than that of Method 3 and of source 2 closer to zero than that of Method 3, suggesting that Method 2 might have recognized some part of spectra of source 2 as part of source 1 more compared with Method 1, but the difference is not significant at all.

5.2.2 Case 2: some visible frequency overlapping

The second part of the evaluation is performed on sources which have some overlapping frequency bands on their spectrograms. Figure 5.2.7 and 5.2.8 shows that note C and G partially overlap each other in the mixture spectrogram at at around 100, 200 and 305 on Y-axis. In such case, it might be harder for a NMF algorithm to separate the notes

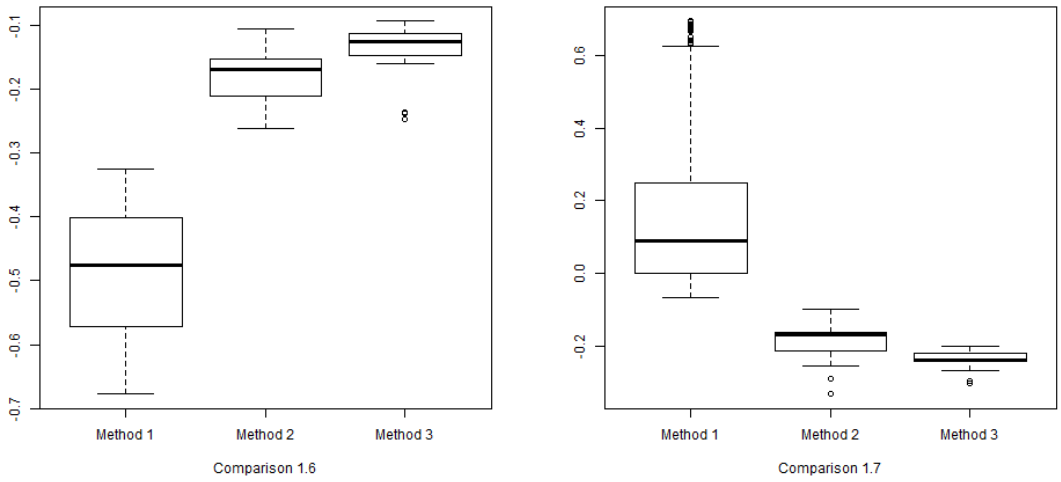

Figure 5.2.6. Comparisons of spectra energy ratios $E(\hat{S}_1)/E(S_1) - 1$ and $E(\hat{S}_2)/E(S_2) - 1$ of the estimated sources $\hat{S}_1, \hat{S}_2$ and the actual corresponding sources S_1, S_2. The closer the value is to 0, the better.

because the two notes share some of their frequency features.

The results in this part of evaluation agree with the results in Case 1 in the previously section.

	1	2	3	4	5	6	7
pianoR	0.0002	0.1474	211.4757	0.0038	0.0035	-0.4033	0.053
harpR	0.0004	0.15	2109.9792	0.0688	0.0934	-0.3222	0.0679
pianoS	0.0012	0.1425	2991.5478	0.2617	0.1446	-0.6671	0.604
zhengS	0.0014	0.1453	6477.3179	0.1475	0.1206	-0.5264	0.199
guitarS	0.0014	0.1438	6289.3404	0.1056	0.1181	-0.4697	0.0855
violinS	0.0012	0.1425	8740.7468	0.0776	0.0506	-0.4199	-0.0321
All	0.001	0.1453	4463.0444	0.1102	0.0889	-0.4662	0.1606

Table 2.1: Evaluation of the regular NMF separation trained on one of the two notes

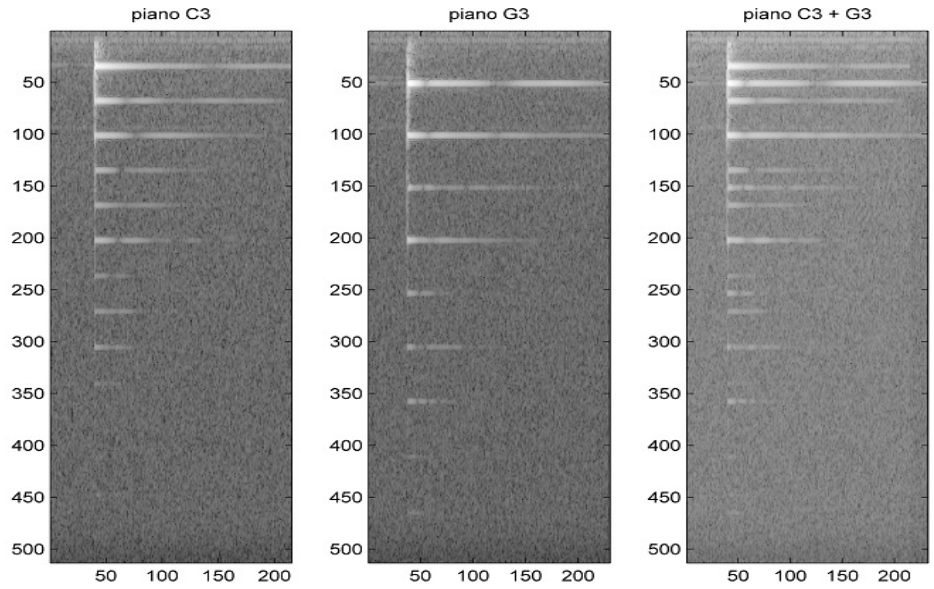

Figure 5.2.7. Spectrograms of note C (left) and G(middle) recorded from a piano. The graph on the right is the simulated mixture of the two notes. It is visible that C and G overlap in their frequency band at position around 100 and 200 on Y-axis.

pianoR	0.0002	0.1524	460.9769	0.0001	0.0003	-0.1305	-0.1811
harpR	0.0005	0.1541	5456.1388	0.0006	0.0184	-0.0977	-0.0904
pianoS	0.0021	0.1636	14061.6941	0.0097	0.0086	-0.2197	-0.2767
zhengS	0.002	0.1536	13280.5881	0.0105	0.0409	-0.1712	-0.2291
guitarS	0.0018	0.1538	13051.822	0.004	0.0383	-0.1303	-0.1347
violinS	0.0017	0.1511	12947.9362	0.009	0.0025	-0.1936	-0.249
All	0.0014	0.1547	9860.356	0.0056	0.0184	-0.156	-0.1914

Table 2.2: Evaluation of the regular NMF separation trained on both of the two notes

pianoR	0.0002	0.1494	470.058	0.0	0.0003	-0.0891	-0.2167
harpR	0.0004	0.1508	5592.2567	0.0002	0.0198	-0.0642	-0.1139
pianoS	0.0014	0.1457	14460.2168	0.0072	0.0212	-0.1885	-0.338
zhengS	0.0015	0.1472	14440.703	0.0034	0.0875	-0.0992	-0.3159
guitarS	0.0014	0.144	13658.3506	0.0015	0.0489	-0.0899	-0.161
violinS	0.0014	0.1466	13815.0925	0.0028	0.0087	-0.1573	-0.2838
All	0.0011	0.1473	10385.8322	0.0025	0.0311	-0.1137	-0.2356

Table 2.3: Evaluation of the NMF separation extended with subtraction method trained on one of the two notes

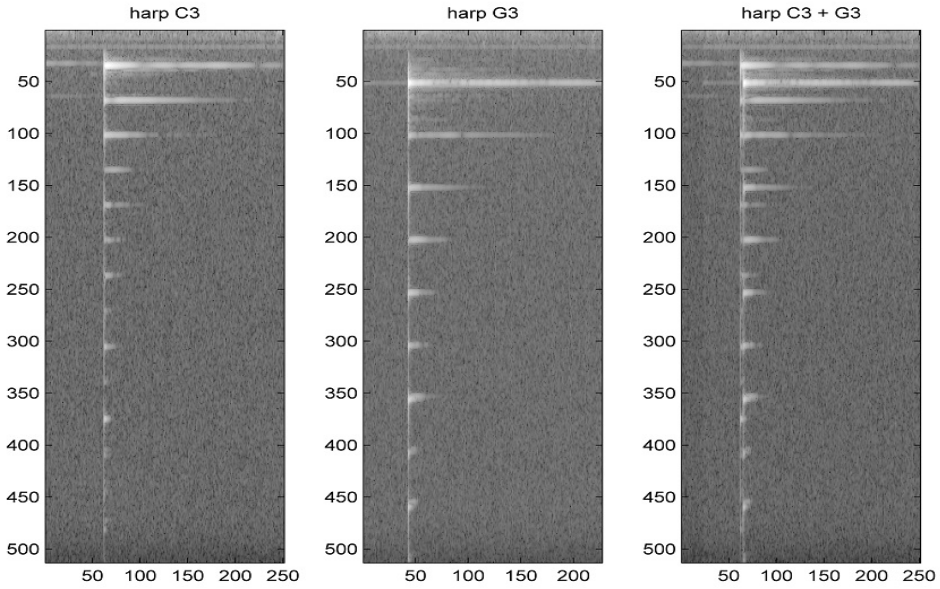

Figure 5.2.8. Spectrograms of note C (left) and G(middle) recorded from a harp. The graph on the right is the simulated mixture of the two notes. It is visible that C and G overlap in their frequency band at position around 100 and 200 on Y-axis.

5.2.3 Case 3: frequency mostly overlapped

Figure 5.2.13 and 5.2.14 are spectrograms of note C_3 and C_4 as well as their simulated mixture played on a real piano and a real harp. In such cases when two notes are differed only over an octave[8]. By looking at the graphs one can see large visible overlapping spectra of the two nodes.

Figure 5.2.2 to 5.2.12 in Case 2 and Figure 5.2.3 to 5.2.18 in Case 3 shows the evaluation results are consistent throughout the three cases. Therefore, it can be concluded that for all the three different cases, the evaluations in this chapter suggest the extension of NMF with subtraction proposed in Section 4.2 improved the original NMF algorithm on separating both the trained note and the untrained note. The subtraction NMF algorithm also reaches a high output accuracy close to an NMF separation trained on both sources.

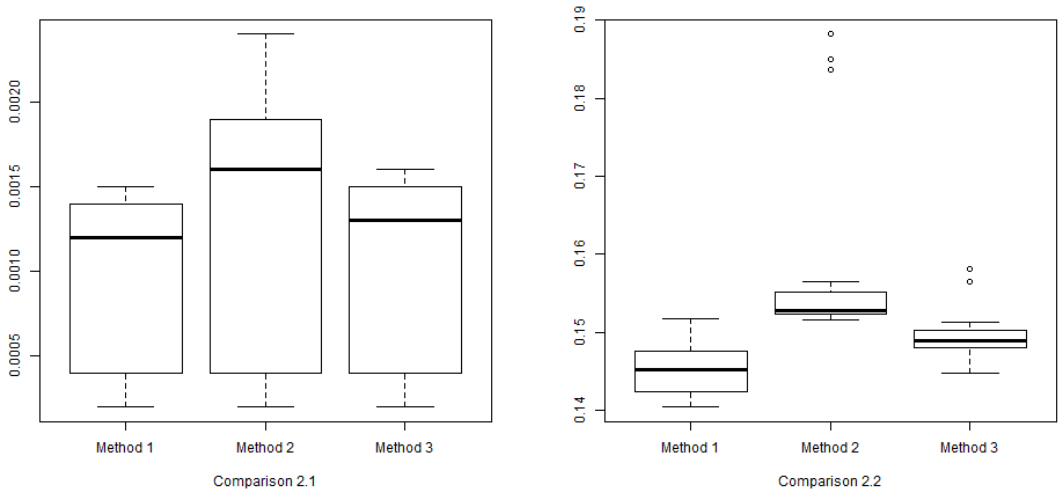

Figure 5.2.9. Comparisons of final cost measured by Euclidean (left) and Itakura-Saito (right) divergence on simulated input $\hat{R}$ with and actual input mixture R. Lower is better.

	1	2	3	4	5	6	7
pianoR	0.0002	0.1464	160.6147	0.0047	0.0044	-0.5077	0.2511
harpR	0.0004	0.1482	2690.3559	0.0528	0.0607	-0.4288	0.2153
pianoS	0.0012	0.1435	5069.297	0.1893	0.1673	-0.5823	0.6869
zhengS	0.0014	0.1525	3658.1715	0.2336	0.1993	-0.6641	0.3662
guitarS	0.0013	0.1472	6772.82	0.1042	0.056	-0.4485	-0.0045
violinS	0.0014	0.1441	7980.8549	0.1217	0.0979	-0.4561	0.1057
All	0.001	0.147	4397.5922	0.1167	0.0966	-0.5126	0.2658

Table 3.1: Evaluation of the regular NMF separation trained on one of the two notes

pianoR	0.0002	0.152	455.3597	0.0001	0.0001	-0.2019	-0.1843
harpR	0.0004	0.1554	5544.016	0.0003	0.0009	-0.1552	-0.1757
pianoS	0.0015	0.1523	16053.5318	0.0021	0.008	-0.157	-0.161
zhengS	0.0024	0.187	13894.8681	0.0096	0.0193	-0.1994	-0.2905
guitarS	0.0019	0.1543	11111.5114	0.0213	0.0119	-0.2144	-0.1645
violinS	0.0019	0.1539	12948.4271	0.0114	0.003	-0.151	-0.2101
All	0.0014	0.1591	9957.8176	0.0076	0.0072	-0.1799	-0.197

Table 3.2: Evaluation of the regular NMF separation trained on both of the two notes

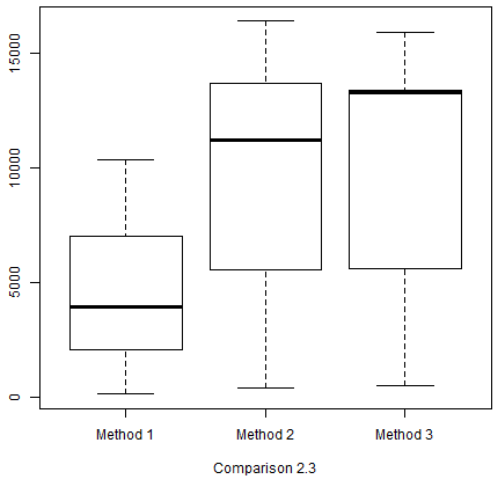

Figure 5.2.10. Cross-correlation comparison with dot product sums on the estimated source $\hat{S}_1$ and the actual corresponding source S_1. Higher is better.

pianoR	0.0002	0.1486	471.3035	0.0	0.0001	-0.1453	-0.2649
harpR	0.0004	0.1501	5627.5656	0.0001	0.0012	-0.1101	-0.2402
pianoS	0.0013	0.1458	15808.0032	0.0024	0.009	-0.1182	-0.2384
zhengS	0.0016	0.1568	13028.5208	0.0154	0.0349	-0.2571	-0.2088
guitarS	0.0014	0.1484	13124.2628	0.0133	0.0784	-0.0988	-0.2436
violinS	0.0015	0.1484	13334.2349	0.0053	0.0087	-0.1443	-0.2171
All	0.0011	0.1497	10210.071	0.0061	0.0225	-0.1446	-0.2357

Table 3.3: Evaluation of the NMF separation extended with subtraction method trained on one of the two notes

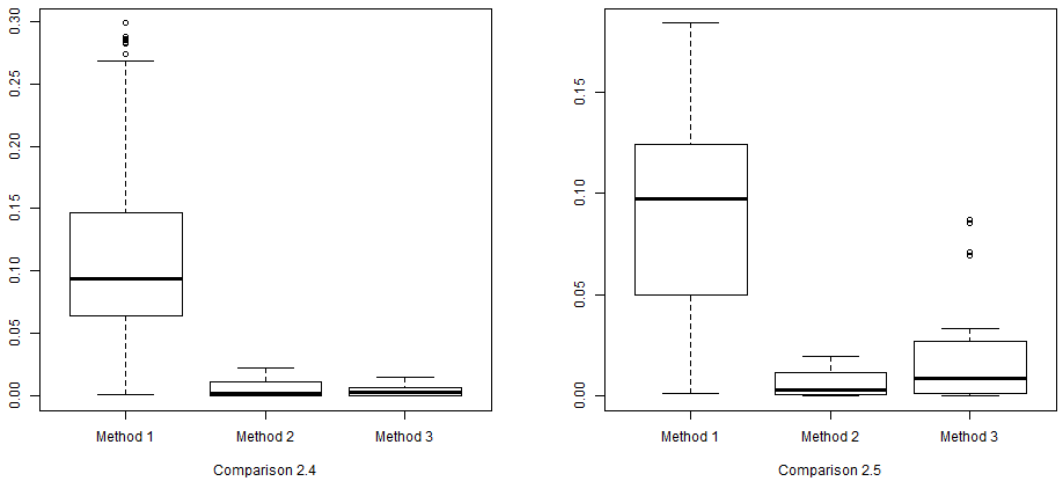

Figure 5.2.11. Cross-correlation comparison with Euclidean divergence on the estimated source $\hat{S}_1, \hat{S}_2$ and the actual corresponding source S_1, S_2. Lower is better.

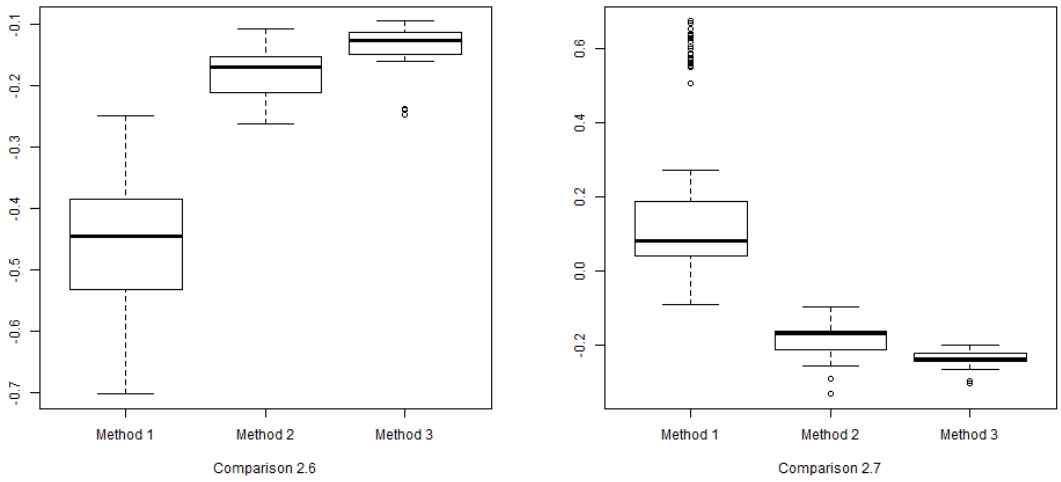

Figure 5.2.12. Comparisons of spectra energy ratios $E(\hat{S}_1)/E(S_1) - 1$ and $E(\hat{S}_2)/E(S_2) - 1$ of the estimated sources $\hat{S}_1, \hat{S}_2$ and the actual corresponding sources S_1, S_2. The closer the value is to 0, the better.

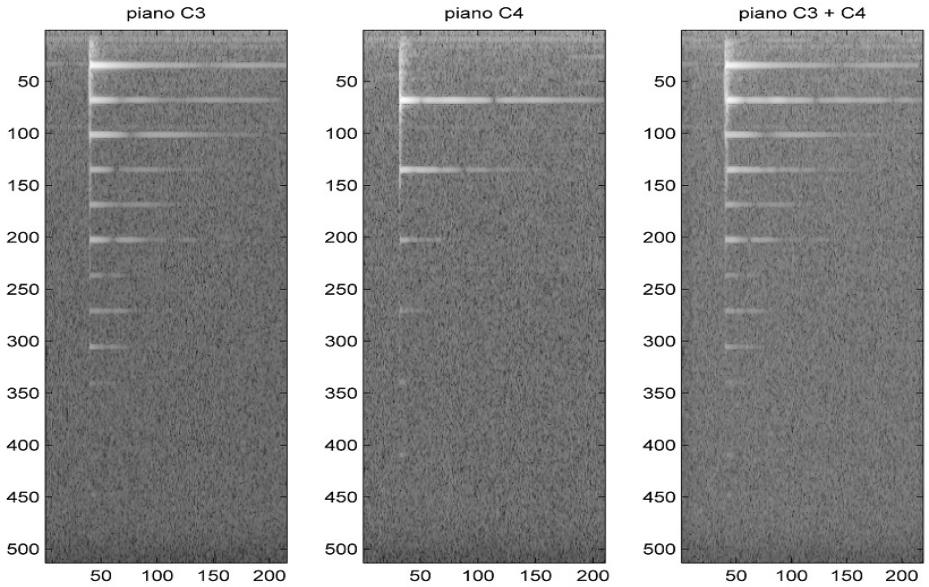

Figure 5.2.13. Spectrograms of note C_3(left) and C_4(middle) recorded from a real piano. The graph on the right is the simulated mixture of the two notes, with large visible overlapping frequency bands from the two nodes.

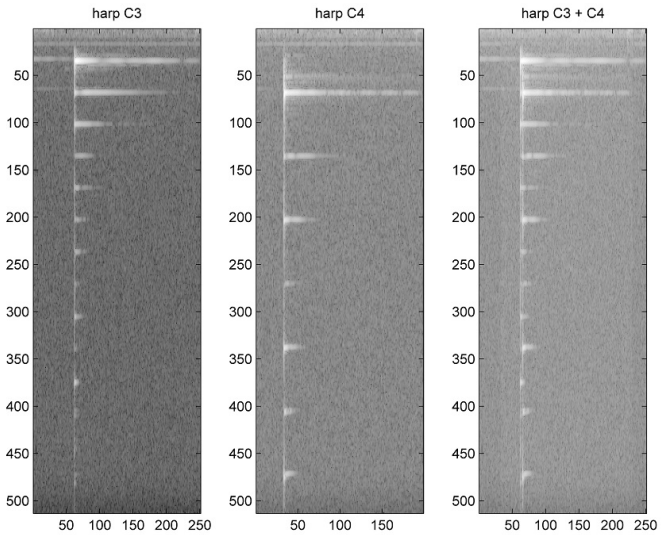

Figure 5.2.14. Spectrograms of note C_3(left) and C_4(middle) recorded from a real harp. The graph on the right is the simulated mixture of the two notes, with large visible overlapping frequency bands from the two nodes.

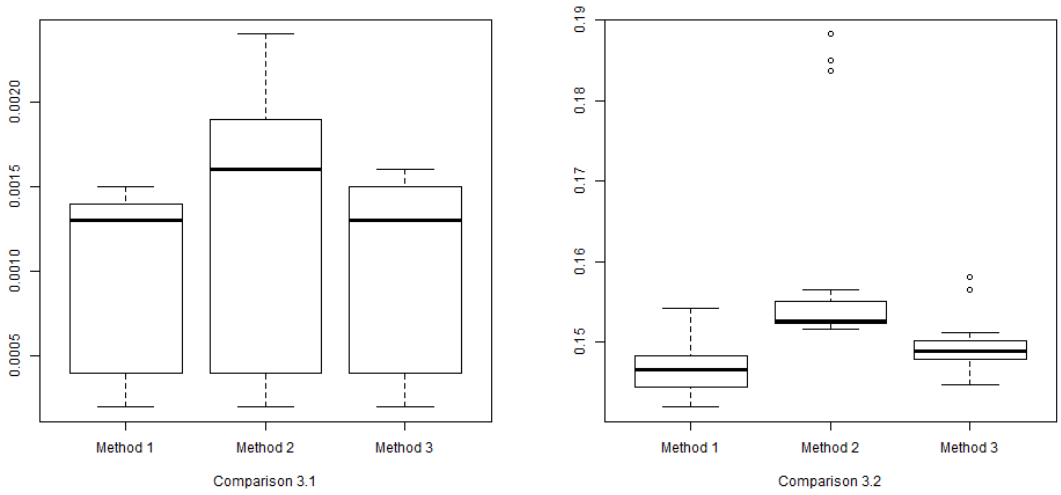

Figure 5.2.15. Comparisons of final cost measured by Euclidean (left) and Itakura-Saito (right) divergence on simulated input $\hat{R}$ with and actual input mixture R. Lower is better.

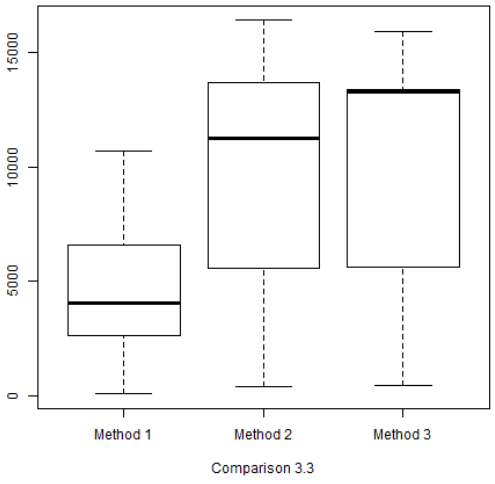

Figure 5.2.16. Cross-correlation comparison with dot product sums on the estimated source $\hat{S}_1$ and the actual corresponding source S_1. Higher is better.

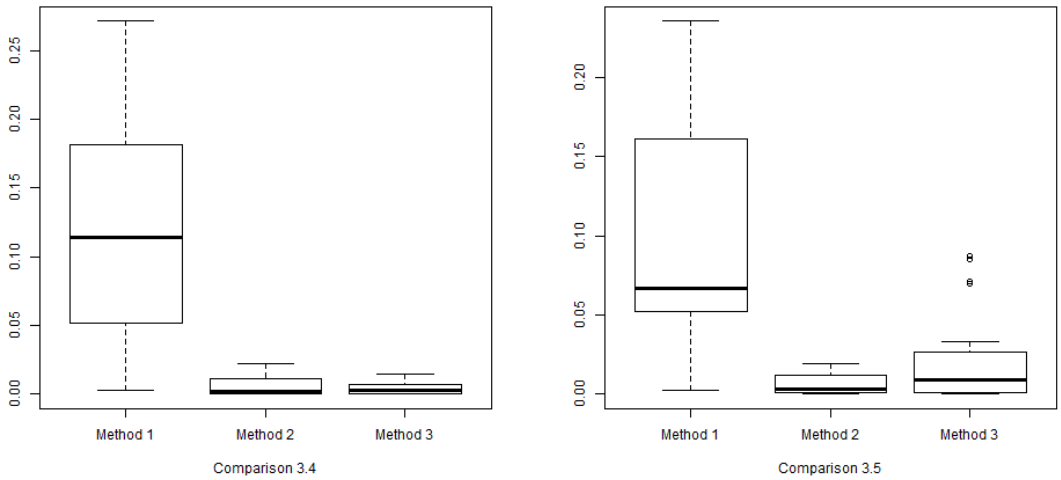

Figure 5.2.17. Cross-correlation comparison with Euclidean divergence on the estimated source $\hat{S}_1, \hat{S}_2$ and the actual corresponding source S_1, S_2. Lower is better.

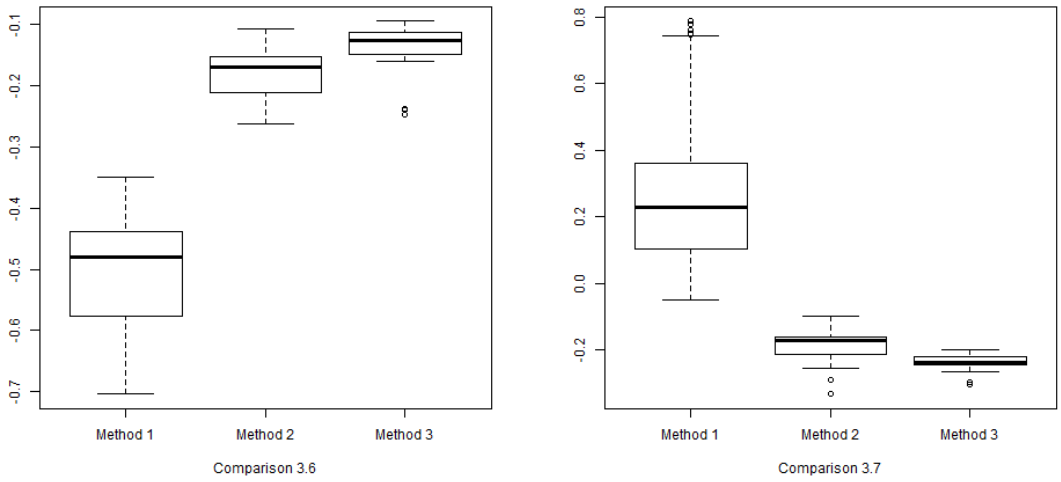

Figure 5.2.18. Comparisons of spectra energy ratios $E(\hat{S}_1)/E(S_1) - 1$ and $E(\hat{S}_2)/E(S_2) - 1$ of the estimated sources $\hat{S}_1, \hat{S}_2$ and the actual corresponding sources S_1, S_2. The closer the value is to 0, the better.

6
[final, frozen]Discussion

The evaluations in the previous chapter demonstrate that the extension of NMF algorithm with a subtraction method proposed in Section 4.2 outperforms the original NMF algorithm. The extended algorithm reaches an accuracy close to the NMF separation trained on both the sources using the original algorithm. The new method subtracts the first source from the mixture and obtains an pseudo-training sample of the untrained source along with any noises and distortions of the first source. Therefore, the second dictionary in the algorithm may contain every feature needed to deal with noises and distortions in the mixture. This might explain why the results also suggest that the new method slightly outperformed the original algorithm trained on both sources for separating the already-trained source while not for the non-trained one. In short, the subtraction method enables the original NMF algorithm to learn the untrained source, the Unknown, by subtracting the already-trained sample, the Known, from the mixture.

A possible limitation to this subtraction method is that in some circumstances the method might find a β value larger than desired. For example, when the sample is multiplied with

a volume larger than its actual volume in the mixture, it can still be subtracted from the mixture without violating the non-negative property:

$$R_2 = Rremaining + \beta_{actual} * R_1 s \text{ and}$$
$$R_2 - \beta * R_{actual} * R_1 s > 0, \text{ where } \beta > \beta_{actual}$$

In the music mixture separation problems, it may happen when the spectra of the trained note completely overlaps the spectra of the other notes. Therefore, in Section 5.2 it was originally hypothesized that the results might differ for the three cases since spectra overlapping might affect the subtraction process. The results in Chapter 5 are consistent throughout the different cases, and the subtraction process is not affected when spectra of sources partially overlaps.

A possible future application of this algorithm is a self-retraining loop. For example, given a mixture of three notes C, D and E in which note C and D are first played simultaneously and then when they fade out, notes D and E are played simultaneously, and the algorithm trained only on note C, a self-retraining loop program will first use the peak-of-energy method from Section 4.1.1 to locate two sound occurrences in the mixture. The application might then take the first sound occurrence and decided that note C is present by a method mentioned in Section 4.1.2, and use the subtraction method developed in Section 4.2 to be trained with a pseudo-training sample of note D. Now the program is trained on notes C and D. The algorithm then determines that note D exists in the second sound occurrence and will therefore be able to separate the third notes out using the same subtraction method. The result of this separation may be close to the regular NMF trained on all the three notes, while a regular NMF trained on only one notes may likely fail to separate the mixture. However, the loop will be slow since it needs to search for existence

of known sources in every sound occurrence.

Bibliography

[1] Michel Marie Deza Elena Deza, *Encyclopedia of Distances, page 94*, Springer, 2009.

[2] Alan H. S. Chan and Sio-Iong Ao, *Advances in industrial engineering and operations research, page 51*, Springer, 2008.

[3] Augustin Lefevre and Francis Bach and Cedric Fevotte, *Online algorithms for Nonnegative Matrix Factorization with the Itakura-Saito divergence*, IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (2011).

[4] Nancy Bertin and Jean-Louis Durrieu Cedric Fevotte, *Nonnegative matrix factorization with the Itakura-Saito divergence. With application to music analysis*, TELECOM ParisTech (2008).

[5] S.E. Anderson and A.S. Dave and D. Margoliash, *Automatic recognition and analysis of birdsong syllables from continuous recordings*, jasa **100** (1996), 1209–1219.

[6] Dennis Sun Nicholas Bryan, *Introduction to Non-Negative Matrix Factorization* (2013), `https://ccrma.stanford.edu/~njb/teaching/sstutorial/`.

[7] Albert Au Yeung, *Matrix Factorization: A Simple Tutorial and Implementation in Python* (2010), `http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/`.

[8] wikipedia, *Overtone*, `http://en.wikipedia.org/wiki/Octave`.

[9] ———, *Spectrogram*, `http://en.wikipedia.org/wiki/Spectrogram`.

[10] ———, *Gradient descent*, `http://en.wikipedia.org/wiki/Gradient_descent`.