Nonnegative Matrix Factor 2-D Deconvolution for Blind Single Channel Source Separation

Mikkel N. Schmidt and Morten Mørup

Technical University of Denmark
Informatics and Mathematical Modelling
Richard Petersens Plads, Building 321
DK-2800 Kgs. Lyngby, Denmark
{mns,mm}@imm.dtu.dk

Abstract. We present a novel method for blind separation of instruments in single channel polyphonic music based on a non-negative matrix factor 2-D deconvolution algorithm. The method is an extention of NMFD recently introduced by Smaragdis [1]. Using a model which is convolutive in both time and frequency we factorize a spectrogram representation of music into components corresponding to individual instruments. Based on this factorization we separate the instruments using spectrogram masking. The proposed algorithm has applications in computational auditory scene analysis, music information retrieval, and automatic music transcription.

1 Introduction

The separation of multiple sound sources from a single channel recording is a difficult problem which has been extensively addressed in the literature. Many of the proposed methods are based on matrix decompositions of a spectrogram representation of the sound. The basic idea is to represent the sources by different frequency signatures which vary in intensity over time.

Non-negative matrix factorization (NMF) [2, 3] has proven a very useful tool in a variety of signal processing fields. NMF gives a sparse (or parts-based) decomposition [3] and under certain conditions the decomposition is unique [4] making it unnecessary to impose constraints in the form of orthogonality or independence. Efficient algorithms for computing the NMF have been introduced by Lee and Seung [5]. NMF has a variety of applications in music signal processing; recently, Helén and Virtanen [6] described a method for separating drums from polyphonic music using NMF and Smaragdis and Brown [7] used NMF for automatic transcription of polyphonic music.

When polyphonic music is modelled by factorizing a magnitude spectrogram with NMF, each instrument note is modelled by an instantaneous frequency signature which can vary over time. Smaragdis [1] introduced an extension to NMF, namely the non-negative matrix factor deconvolution (NMFD) algorithm, in which each instrument note is modelled by a time-frequency signature which varies in intensity over time. Thus, the model can represent components with

temporal structure. Smaragdis showed how this can be used to separate individual drums from a real recording of drum sounds. This approach was further pursued by Wang and Plumbley [8] who separated mixtures of different musical instruments. Virtanen [9] presented an algorithm based on similar ideas and evaluated its performance by separating mixtures of harmonic sounds.

In this paper, we propose a new method to factorize a log-frequency spectrogram using a model which can represent both temporal structure and the pitch change which occurs when an instrument plays different notes. We denote this the non-negative matrix factor 2-D deconvolution (NMF2D). We use a log-frequency spectrogram because a pitch change in a log-frequency domain simply corresponds to a displacement on the frequency axis, whereas in a linear-frequency domain a pitch change would also change the distance between the harmonics. Where previous methods need one component to model each note for each instrument, the proposed model represents each instrument compactly by a single time-frequency profile convolved in both time and frequency by a time-pitch weight matrix. This model impressively decreases the number of components needed to model various instruments and effectively solves the blind single channel source separation problem for certain classes of musical signals. In section 2 we introduce the NMF2D model and derive the update equations for recursively computing the factorization based on two different cost functions. In section 3 we show how the algorithm can be used to analyze and separate polyphonic music and we compare the algorithm to the NMFD method of Smaragdis [1]. This is followed by a discussion of the results.

2 Method

We start by recalling the non-negative matrix factorization (NMF) problem: $\mathbf{V} \approx \mathbf{WH}$, where $\mathbf{V}$, $\mathbf{W}$, and $\mathbf{H}$ are non-negative matrices. The task is find factors $\mathbf{W}$ and $\mathbf{H}$ which approximate $\mathbf{V}$ as well as possible according to some cost function. Lee and Seung [5] devise two algorithms to find $\mathbf{W}$ and $\mathbf{H}$: For the least squares error and the KL divergence they show that the following recursive updates converge to a local minimum:

$$\begin{aligned} \text{Least squares error}: \quad & \mathbf{W} \leftarrow \mathbf{W} \bullet \frac{\mathbf{V}\mathbf{H}^T}{\mathbf{W}\mathbf{H}\mathbf{H}^T}, \quad \mathbf{H} \leftarrow \mathbf{H} \bullet \frac{\mathbf{W}^T\mathbf{V}}{\mathbf{W}^T\mathbf{W}\mathbf{H}}, \\ \text{KL divergence}: \quad & \mathbf{W} \leftarrow \mathbf{W} \bullet \frac{\frac{\mathbf{V}}{\mathbf{W}\mathbf{H}}\mathbf{H}^T}{\mathbf{1}\cdot\mathbf{H}^T}, \quad \mathbf{H} \leftarrow \mathbf{H} \bullet \frac{\mathbf{W}^T\frac{\mathbf{V}}{\mathbf{W}\mathbf{H}}}{\mathbf{W}^T\cdot\mathbf{1}}, \end{aligned}$$

where $A \bullet B$ denotes element-wise multiplication, $\frac{A}{B}$ denotes element-wise division, and $\mathbf{1}$ is a matrix with all elements unity. These algorithms can be derived by minimizing the cost function using gradient descent and choosing the stepsize appropriately to yield simple multiplicative updates.

2.1 NMF2D

We extend the NMF model to be a 2-dimensional convolution of $\mathbf{W}^\tau$ which depends on time, τ, and $\mathbf{H}^\phi$ which depends on pitch, ϕ. This forms the non-

negative matrix factor 2-D deconvolution (NMF2D) model:

$$\mathbf{V} \approx \mathbf{\Lambda} = \sum_{\tau} \sum_{\phi} \overset{\downarrow\phi}{\mathbf{W}}^{\tau} \overset{\rightarrow\tau}{\mathbf{H}}^{\phi}, \tag{1}$$

where $\downarrow \phi$ denotes the downward shift operator which moves each element in the matrix ϕ rows down, and $\rightarrow \tau$ denotes the right shift operator which moves each element in the matrix τ columns to the right, i.e.:

$$\mathbf{A} = \begin{pmatrix} 1\,2\,3 \\ 4\,5\,6 \\ 7\,8\,9 \end{pmatrix}, \quad \overset{\downarrow 2}{\mathbf{A}} = \begin{pmatrix} 0\,0\,0 \\ 0\,0\,0 \\ 1\,2\,3 \end{pmatrix}, \quad \overset{\rightarrow 1}{\mathbf{A}} = \begin{pmatrix} 0\,1\,2 \\ 0\,4\,5 \\ 0\,7\,8 \end{pmatrix}.$$

We note that the NMFD model introduced by Smaragdis [1] is a special case of the NMF2D model where $\phi = \{0\}$.

The NMF2D model can be used to factorize a log-frequency magnitude spectrogram of polyphonic music into factors corresponding to individual instruments: If the matrix $\mathbf{V}$ is a log-frequency spectrogram representation of a piece of polyphonic music, the columns of $\mathbf{W}^{\tau}$ correspond to the time-frequency signature of each instrument, and the rows of $\mathbf{H}^{\phi}$ correspond to the time-pitch signature of each instrument, i.e. which notes are played by the instrument at what time. In other words, the convolution in time, τ, accounts for the temporal structure of each instrument, and the convolution in pitch, ϕ, accounts for each instrument playing different tones.

2.2 Least Squares NMF2D

We now derive a set of recursive update steps for computing $\mathbf{W}^{\tau}$ and $\mathbf{H}^{\phi}$ based on gradient descent with multiplicative updates. We consider the least squares (LS) cost function which corresponds to maximizing the likelihood of a gaussian noise model:

$$C_{LS} = ||\mathbf{V} - \mathbf{\Lambda}||_f^2 = \sum_i \sum_j (\mathbf{V}_{i,j} - \mathbf{\Lambda}_{i,j})^2. \tag{2}$$

A given element in $\mathbf{\Lambda}$, defined in equation (1), is given by:

$$\mathbf{\Lambda}_{i,j} = \sum_{\tau} \sum_{\phi} \sum_{d} \mathbf{W}^{\tau}_{i-\phi,d} \mathbf{H}^{\phi}_{d,j-\tau}. \tag{3}$$

In the following we will need the derivative of a given element $\mathbf{\Lambda}_{i,j}$ with respect to a given element $\mathbf{W}^{\tau}_{k,d}$:

$$\frac{\partial \mathbf{\Lambda}_{i,j}}{\partial \mathbf{W}^{\tau}_{k,d}} = \frac{\partial}{\partial \mathbf{W}^{\tau}_{k,d}} \left(\sum_{\tau} \sum_{\phi} \sum_{d} \mathbf{W}^{\tau}_{i-\phi,d} \mathbf{H}^{\phi}_{d,j-\tau} \right) \tag{4}$$

$$= \frac{\partial}{\partial \mathbf{W}^{\tau}_{k,d}} \left(\sum_{\phi} \mathbf{W}^{\tau}_{i-\phi,d} \mathbf{H}^{\phi}_{d,j-\tau} \right) \tag{5}$$

$$= \begin{cases} \mathbf{H}^{\phi}_{d,j-\tau} & \phi = i - k \\ 0 & otherwise. \end{cases} \tag{6}$$

Differentiating C_{LS} with respect to a given element in $\mathbf{W}^\tau$ gives:

$$\frac{\partial C_{LS}}{\partial \mathbf{W}^\tau_{k,d}} = \frac{\partial}{\partial \mathbf{W}^\tau_{k,d}} \sum_i \sum_j (\mathbf{V}_{i,j} - \mathbf{\Lambda}_{i,j})^2 \tag{7}$$

$$= -2 \sum_i \sum_j (\mathbf{V}_{i,j} - \mathbf{\Lambda}_{i,j}) \frac{\partial \mathbf{\Lambda}_{i,j}}{\partial \mathbf{W}^\tau_{k,d}} \tag{8}$$

$$= -2 \sum_\phi \sum_j (\mathbf{V}_{\phi+k,j} - \mathbf{\Lambda}_{\phi+k,j}) \mathbf{H}^\phi_{d,j-\tau}. \tag{9}$$

The recursive update steps for the gradient descent are given by:

$$\mathbf{W}^\tau_{k,d} \leftarrow \mathbf{W}^\tau_{k,d} - \eta \frac{\partial C_{LS}}{\partial \mathbf{W}^\tau_{k,d}}. \tag{10}$$

Similar to the approach of Lee and Seung [5], we choose the step size η so that the first term in (10) is canceled:

$$\eta = \frac{\mathbf{W}^\tau_{k,d}}{-2 \sum_\phi \sum_j \mathbf{\Lambda}_{\phi+k,j} \mathbf{H}^\phi_{d,j-\tau}}, \tag{11}$$

which gives us the following simple multiplicative updates:

$$\mathbf{W}^\tau_{k,d} \leftarrow \mathbf{W}^\tau_{k,d} \frac{\sum_\phi \sum_j \mathbf{V}_{\phi+k,j} \mathbf{H}^\phi_{d,j-\tau}}{\sum_\phi \sum_j \mathbf{\Lambda}_{\phi+k,j} \mathbf{H}^\phi_{d,j-\tau}}. \tag{12}$$

By noticing that transposing equation (1) interchanges the order of $\mathbf{W}^\tau$ and $\mathbf{H}^\phi$ in the model, the updates for $\mathbf{H}^\phi$ can easily be found. In matrix notation the updates can be written as:

$$\mathbf{W}^\tau \leftarrow \mathbf{W}^\tau \bullet \frac{\sum_\phi \overset{\uparrow\phi}{\mathbf{V}} \overset{\rightarrow\tau}{\mathbf{H}^\phi}^T}{\sum_\phi \overset{\uparrow\phi}{\mathbf{\Lambda}} \overset{\rightarrow\tau}{\mathbf{H}^\phi}^T} \qquad \mathbf{H}^\phi \leftarrow \mathbf{H}^\phi \bullet \frac{\sum_\tau \overset{\downarrow\phi}{\mathbf{W}^\tau}^T \overset{\leftarrow\tau}{\mathbf{V}}}{\sum_\tau \overset{\downarrow\phi}{\mathbf{W}^\tau}^T \overset{\leftarrow\tau}{\mathbf{\Lambda}}}. \tag{13}$$

2.3 Kullback-Leibler NMF2D

We can also find similar equations for computing the NMF2D based on the Kullback-Leibler (KL) divergence:

$$C_{KL} = \sum_i \sum_j \mathbf{V}_{i,j} \log \frac{\mathbf{V}_{i,j}}{\mathbf{\Lambda}_{i,j}} - \mathbf{V}_{i,j} + \mathbf{\Lambda}_{i,j}. \tag{14}$$

Minimizing the KL divergence corresponds to assuming a multinomial noise model. By taking similar steps as in the derivation of the LS updates we find the following recursive updates for the KL cost function:

$$\mathbf{W}^\tau \leftarrow \mathbf{W}^\tau \bullet \frac{\sum_\phi \left(\frac{\overset{\uparrow\phi}{\mathbf{V}}}{\overset{\uparrow\phi}{\mathbf{\Lambda}}}\right) \overset{\rightarrow\tau}{\mathbf{H}^\phi}^T}{\sum_\phi \mathbf{1} \cdot \overset{\rightarrow\tau}{\mathbf{H}^\phi}^T} \qquad \mathbf{H}^\phi \leftarrow \mathbf{H}^\phi \bullet \frac{\sum_\tau \overset{\downarrow\phi}{\mathbf{W}^\tau}^T \left(\frac{\overset{\leftarrow\tau}{\mathbf{V}}}{\overset{\leftarrow\tau}{\mathbf{\Lambda}}}\right)}{\sum_\tau \overset{\downarrow\phi}{\mathbf{W}^\tau}^T \cdot \mathbf{1}}. \tag{15}$$

3 Experimental Results

In order to demonstrate our NMF2D algorithm, we have analyzed a 4 second piece of computer generated polyphonic music containing a trumpet and a piano. For comparison we have also analyzed the same piece of music by the NMFD algorithm [1]. For both algorithms we used the least squares cost function. The score of the piece of music is shown in Fig. 1. The trumpet and the piano play a different short melodic passage each consisting of three distinct notes. We generated the music at a sample rate of 16 kHz and analyzed it by the short time Fourier transform with a 2048 point Hanning windowed FFT and 50% overlap. This gave us 63 FFT frames. We grouped the spectrogram bins into 175 logarithmically spaced frequency bins in the range of 50 Hz to 8 kHz with 24 bins per octave, which correponds to twice the resolution of the equal tempered musical scale. Then, we performed the NMF2D and NMFD factorization of the log-frequency magnitude spectrogram. The parameters of the two models were selected so that both methods were able to model the music almost perfectly:

For the NMF2D we used two factors, $d = 2$, since we seek to separate two instruments. The NMF2D method requires at least as many factors as the number of individual instruments. We empirically chose to use seven convolutive components in time, $\tau = \{0, \ldots, 6\}$, corresponding to approximatly 45 ms, which we found to capture the temporal structure of the instruments well. The pitch of the notes played in the music span three whole notes. Consequently, we chose to use 12 convolutive components in pitch, i.e. $\phi = \{0, \ldots, 11\}$.

For the NMFD we used six factors, $d = 6$, corresponding to the total number of different tones played by the two instruments. The NMFD method requires at least as many factors as the number of distinctive instrument notes. Similar to the experiment with NMF2D we used seven convolutive components in time. For the experiment with NMFD we used our formulation of the NMF2D algorithm with $\phi = \{0\}$, since the NMFD is a special case of the NMF2D algorithm.

The results of the experiments with NMFD and NMF2D are shown in Fig. 2 and Fig. 3 respectively. As we expected, the NMFD algorithm factorized each individual note from each instrument into a separate component, whereas the NMF2D algorithm factorized each instrument into a separate component.

We used the NMF2D factorization to reconstruct the individual instruments separately. First, we reconstructed the spectrogram of each individual instrument by computing equation (3) for each specific value of d. These reconstructed spectrograms could be used directly to reconstruct the instruments, but since the log-frequency spectrograms are computed with a relatively low frequency resolution we reconstructed the signals using spectrogram masking: We mapped the reconstructed log-frequency spectrograms back into the linear-frequency spectrogram domain and computed a spectrogram mask for each instrument which assigned each spectrogram bin to the instrument with the highest power at that bin. We filtered the original spectrogram based on the masks, and computed the inverse filtered spectrogram using the original phase. The separation of the two instruments in the music is shown in Fig. 4. Informal listening test indicated, that the NMF2D algorithm was able to separate the two instruments very well.

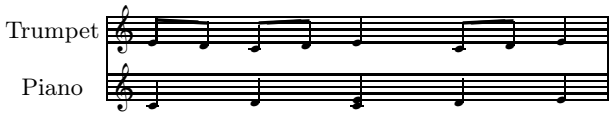

Fig. 1. Score of the piece of music used in the experiments. The music consists of a trumpet and a piano which play different short melodic passages each consisting of three distinct notes.

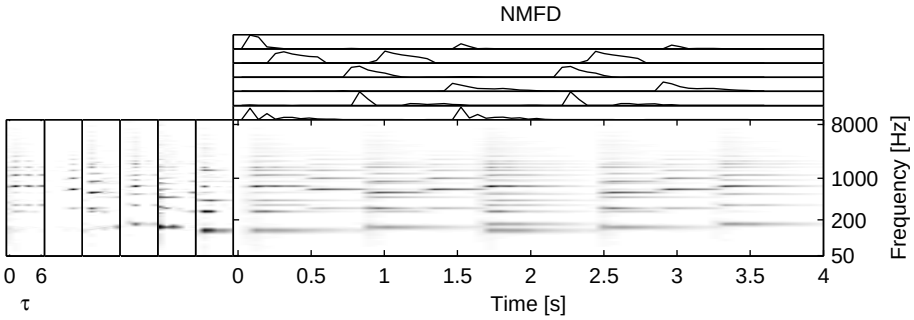

Fig. 2. Factorization of the piece of music using NMFD. The six time-frequency plots on the left are $\mathbf{W}^{\tau}$ for each factor, i.e. the time-frequency signature of the distincts tone played by the two instruments. The six plots on the top are the rows of $\mathbf{H}$ showing how the individual instrument notes are placed in time. The factors have been manually sorted so that the first three corresponds to the trumpet and the last three correspond to the piano.

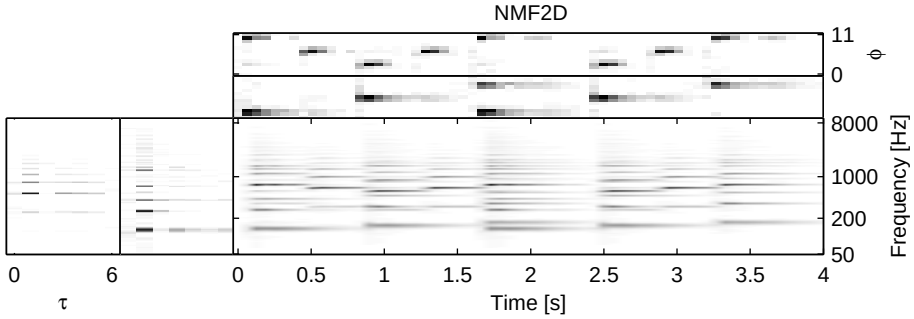

Fig. 3. Factorization of the piece of music using NMF2D. The two time-frequency plots on the left are $\mathbf{W}^{\tau}$ for each factor, i.e. the time-frequency signature of the two instruments. The two time-pitch plots on the top are $\mathbf{H}^{\phi}$ for each factor showing how the two instrument notes are placed in time and pitch.

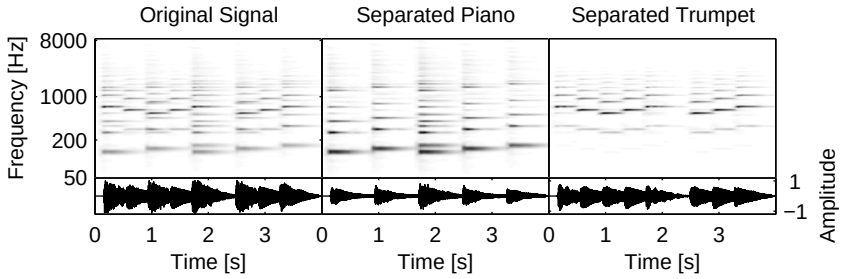

Fig. 4. Single channel source separation using NMF2D. The plots show the log-frequency spectrogram and the waveform of the music and the separated instruments.

4 Discussion

In the previous section we compared the proposed NMF2D algorithm with NMFD. Both the NMF2D and the NMFD algorithm can be used to separate the instruments in polyphonic music. However, since in NMFD the notes of the individual instruments are spread over a number of factors, these must first be grouped manually or by other means. The advantage of the NMF2D algorithm is, that it implicitly solves the problem of grouping notes.

If the assumption holds, that all notes for an instrument is an identical pitch shifted time-frequency signature, the NMF2D model will give better estimates of these signatures, because more examples (different notes) are used to compute each time-frequency signature. Even when this assumption does not hold, it migth still hold in a region of notes for an instrument. Furthermore, the NMF2D algorithm might be able to explain the spectral differences between two notes of different pitch by the 2-D convolution of the time-frequency signature.

Both the NMFD and NMF2D models almost perfectly explained the variation in the spectrogram. However, the number of free parameters in the two models is quite different. If the dimensionality of the spectrogram is $I \times J$, and n_τ, n_ϕ denote the number of convolutive lags in time and pitch, NMFD has $(n_\tau I + J) \cdot d = (7 \cdot 175 + 63) \cdot 6 = 7728$ parameters whereas NMF2D has $(n_\tau I + n_\phi J) \cdot d = (7 \cdot 175 + 12 \cdot 63) \cdot 2 = 3962$ parameters. Consequently, the NMF2D was more restricted making the NMF2D the best model from an Occam's razor point of view.

Admittedly, the simple computer generated piece of music analyzed in this paper favors the NMF2D algorithm since each instrument key is almost a simple spectral shift of the same time-frequency signature. However, when we analyze real music signals the NMF2D also gives very good results. Demonstrations of the algorithm for different music signals can be found at *www.intelligentsound.org*.

It is worth noting, that while we had problems making the NMFD algorithm converge in some situations when using the updates given by Smaragdis [1], the updates devised in this paper to our knowledge always converge.

In the experiments above we used the NMF2D based on least squares. However, using the algorithm based on minimizing the KL divergence gave similar results. It is also worth mentioning that the NMF2D analysis is computationally inexpensive; the results in the previous section took approximatly 20 seconds to compute on a 2 GHz Pentium 4 computer.

It is our belief that the NMF2D algorithm can be useful in a wide range of areas including computational auditory scene analysis, music information retrieval, audio coding, automatic music transcription, and image analysis.

References

1. Smaragdis, P.: Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. Lecture Notes in Computer Science **3195** (2004) 494–499
2. Paatero, P., Tapper, U.: Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. Environmetrics **5**(2) (1994) 111–126
3. Lee, D., Seung, H.: Learning the parts of objects by non-negative matrix factorization. Nature **401**(6755) (1999) 788–91
4. Donoho, D., Stodden, V.: When does non-negative matrix factorization give a correct decomposition into parts? NIPS (2003)
5. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: NIPS. (2000) 556–562
6. Helén, M., Virtanen, T.: Separation of drums from polyphonic music using non-negative matrix factorization and support vector machine. In: 13th European Signal Processing Conference. (2005)
7. Smaragdis, P., Brown, J.C.: Non-negative matrix factorization for polyphonic music transcription. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA) (2003) 177–180
8. Wang, B., Plumbley, M.D.: Musical audio stream separation by non-negative matrix factorization. In: Proceedings of the DMRN Summer Conference. (2005)
9. Virtanen, T.: Separation of sound sources by convolutive sparse coding. SAPA (2004)