Abstract

We augment the nonnegative matrix factorization method for audio source separation with cues about directionality of sound propagation. This improves separation quality greatly and removes the need for training data, but doubles the computation.

1 Introduction

Nonnegative matrix factorization (NMF) has proven to be an effective and extensible method for time-frequency-based audio analysis and source separation. We show how to guide NMF to identify discrete sound sources by providing cues in the form of potentially noisy direction of arrival (DOA) estimates for each time-frequency bin. We do so by forming a (potentially sparse) frequency × time × direction tensor $X$ indicating the distribution of energy in the soundscape and fitting it by finding $B$, $W$, and $H$ in a factorization of the form

$$X(f, t, d) \approx \sum_{s,z} B(d, s) W(f, z, s) H(t, z, s),$$

where $s$ and $z$ index sources and subcomponents thereof. Advantages of our approach include:

- Perceived separation quality – much better than traditional NMF;
- No supervision – clean training data for individual sources is not needed;
- Minimal overhead – computational resources on the same order as traditional NMF;
- Suitability for small arrays – usable DOA estimates can be obtained from arrays much smaller than typical sound wavelengths, where beamforming techniques do not apply.

*Analog Devices | Lyric Labs: Cambridge, MA 02139. noah.stein@analog.com.
Several approaches to extending NMF-like techniques to multichannel audio appear in the literature. FitzGerald, Cranitch, and Coyle apply a form of nonnegative tensor factorization (NTF) to the frequency × times × number of channels tensor produced by stacking the spectrograms for all channels [2]. This approach has two drawbacks. First, each dictionary element in the decomposition has its own gain to each channel, so a post-NTF clustering phase is needed to group dictionary elements into sources. Second, it relies on level differences between channels to distinguish sources, so it only applies to widely spaced microphones or artificial mixtures.

Ozerov and Févotte address the first drawback by constraining the factorization so the gains from source to channel are shared between the dictionary elements which comprise each source [6]. Lee, Park, and Sung use a similar factorization and confront the second drawback by stacking spectrograms from beamformers in place of raw channels [5].

While independent from our work, [5] can be viewed as an instance of the general method we present in Section 3.1 below. The method of [5] improves over the previous methods by allowing for microphones which are less widely spaced, but still requires spacing wide enough for beamforming. Furthermore the computational resources required are proportional to the number of beamformers used, so for good spatial resolution the cost may be high. The authors only consider experiments with beamformers pointed north, south, east, and west, which may provide poor separation when the sources of interest are not axis-aligned. We address these issues in Section 3.3.

There are at this point a vast majority of extensions to NMF, both generic and audio-specific, allowing for sparsity, more structured decompositions, temporal dependencies, a variety of error metrics, etc. Rather than list them all here, we note that the contribution of this paper is orthogonal to many of these extensions and could easily be combined with them to produce better results. For simplicity we focus on the most basic version.

The remainder of this paper is organized as follows. Section 2 covers basic NMF and a simple DOA estimator. Our main contribution is introduced in Section 3.1 and efficient implementation is explored in Sections 3.2 and 3.3. We show experimental results in Section 4 and close with conclusions in Section 5.

2 Background

This known material is presented to build toward describing and analyzing our model.

2.1 Nonnegative Matrix Factorization (NMF)

NMF is an analysis technique to approximately factor an elementwise nonnegative matrix $X \in \mathbb{R}_{\geq 0}^{F \times T}$ as $X \approx WH$ with $W \in \mathbb{R}_{\geq 0}^{F \times Z}$ and $H \in \mathbb{R}_{\geq 0}^{Z \times T}$. The inner dimension $Z \ll F, T$ is chosen to control the complexity of the learned model. This technique is often applied to a time-frequency representation $X$ (e.g. a magnitude spectrogram) of an audio signal [8].

To simplify what follows we assume $X$ has been normalized so $\sum_{f,t} X_{ft} = 1$ and we use probabilistic language. In place of $X$ we take a probability distribution $p_{\text{obs}}(f,t)$ as given
Figure 1: Directed graphical models summarizing the factorizations discussed in this paper. A joint distribution over the shaded variables is given. Unshaded nodes are unobserved.

seek to approximately decompose $p^{\text{obs}}(f, t) \approx \sum_z q(f, t, z)$, where $q$ is factored into any of the equivalent forms illustrated graphically in Figure 1(a):

$$q(f, t, z) := q(f, z)q(t \mid z) = q(f \mid z)q(t, z) = q(z)q(f \mid z)q(t \mid z).$$

(1)

The values of $z$ index a dictionary of prototype spectra $q(f \mid z)$ which combine according to the activations $q(t, z)$. We seek to minimize the Kullback-Leibler (KL) divergence

$$D_{KL} \left( p^{\text{obs}} \left\| \sum_z q(\cdot, \cdot, z) \right. \right) := \sum_{f, t} p^{\text{obs}}(f, t) \log \frac{p^{\text{obs}}(f, t)}{\sum_z q(f, t, z)}.$$

Since $p^{\text{obs}}$ is fixed, this is equivalent to maximizing the cross entropy

$$\alpha(q) := \sum_{f, t} p^{\text{obs}}(f, t) \log q(f, t) = \sum_{f, t} p^{\text{obs}}(f, t) \log \sum_z q(f, t, z).$$

We use Minorization-Maximization, a generalization of Expectation-Maximization, to locally optimize the cross entropy over factored distributions $q$. This is an iterative procedure; the essential step is to go from a factored distribution $q^0$ to a factored distribution $q^1$ with higher cross entropy. Typically one initializes $q^0$ randomly, computes $q^1$, then iterates with $q^1$ in place of $q^0$ to convergence.

Fix $q^0$. Applying Jensen’s inequality to the logarithm, we have:

$$\sum_z q^0(z \mid f, t) \log \frac{q(f, t, z)}{q^0(z \mid f, t)} \leq \log \sum_z q^0(z \mid f, t) \frac{q(f, t, z)}{q^0(z \mid f, t)} = \log \sum_z q(f, t, z)$$

for all $f, t$, and $q$. Furthermore we have equality when $q = q^0$, since all the terms being averaged are equal. Substituting into $\alpha(q)$ gives

$$\beta(q) := \sum_{f, t, z} p^{\text{obs}}(f, t) q^0(z \mid f, t) \log \frac{q(f, t, z)}{q^0(z \mid f, t)} \leq \alpha(q),$$
again with equality at \( q = q^0 \) (\( \beta \) is said to \textbf{minorize} \( \alpha \) at \( q^0 \)). Choosing \( q^1 \) to maximize \( \beta \) gives \( \alpha(q^1) \geq \beta(q^1) \geq \beta(q^0) = \alpha(q^0) \).

The denominator in \( \beta \) only contributes an additive constant, so we can equivalently maximize the cross entropy \( \gamma \) between \( r(f, t, z) := p_{\text{obs}}(f, t)q^0(z \mid f, t) \) and \( q(f, t, z) \):

\[
\gamma(q) := \sum_{f,t,z} r(f, t, z) \log q(f, t, z) = \sum_{f,t,z} r(f, t, z) \log q(f \mid z) q(t, z) \\
= \sum_{z} r(z) \sum_{f} r(f \mid z) \log q(f \mid z) + \sum_{t,z} r(t, z) \log q(t, z).
\]

Though our original goal was to maximize a cross entropy, we have made progress in the senses that (a) there is no longer a sum inside the logarithm and (b) we have decoupled the terms involving \( q(f \mid z) \) and \( q(t, z) \). These can be chosen independently and arbitrarily while maintaining the factored form \( q(f, t, z) = q(f \mid z)q(t, z) \).

To complete the computation of \( q^1 \) we repeatedly invoke Gibbs’ inequality, which states that for any fixed discrete probability distribution \( \sigma \) the probability distribution \( \tau \) which optimizes the cross entropy \( \sum_{u} \sigma(u) \log \tau(u) \) is \( \tau = \sigma \). Therefore we optimize \( \gamma \) by choosing \( q^1(f \mid z) := r(f \mid z) \) and \( q^1(t, z) := r(t, z) \). Typically \( q^1(f, t, z) \neq r(f, t, z) \); rather \( q^1(f, t, z) \) is a product of a marginal and a conditional of \( r \), which itself might not factor.

These updates can be viewed as alternating projections. We seek a distribution \( q(f, t, z) \) which (a) factors and (b) has marginal \( q(f, t) \) close to \( p_{\text{obs}}(f, t) \). We begin with \( q^0(f, t, z) \) which satisfies (a) but not (b) and modify it to get \( r(f, t, z) = p_{\text{obs}}(f, t)q^0(z \mid f, t) \), which gives (b) exactly but destroys (a). Then we multiply a marginal and conditional of \( r(f, t, z) \) to get \( q^1(f, t, z) \) sacrificing (b) and satisfying (a), and repeat.

### 2.2 Multiplicative updates for NMF

The iteration derived in Section 2.1 can be computed efficiently, resulting in the celebrated multiplicative updates of Lee and Seung [4]. We must compute

\[
q^1(t, z) = \sum_{f} r(f, t, z) = \sum_{f} p_{\text{obs}}(f, t)q^0(z \mid f, t) = \sum_{f} p_{\text{obs}}(f, t) \frac{q^0(f \mid z)q^0(t, z)}{q^0(f, t)} \\
= q^0(t, z) \sum_{f} \frac{p_{\text{obs}}(f, t)}{q^0(f, t)} q^0(f \mid z) = q^0(t, z) \sum_{f} \rho(f, t)q^0(f \mid z).
\]

Matrix multiply to find \( q^0(f, t) = \sum_{z} q^0(f, z')q^0(t \mid z') \). Elementwise divide to get \( \rho(f, t) := p_{\text{obs}}(f, t)/q^0(f, t) \). Matrix multiply for \( \sum_{f} \rho(f, t)q^0(f \mid z) \) and elementwise multiply by \( q^0(t, z) \) to get \( q^1(t, z) \). Reuse \( \rho(f, t) \) to compute \( q^1(f, z) \) analogously and condition to get \( q^1(f \mid z) \).

In this way we avoid storing any \( F \times T \times Z \) arrays, such as \( r(f, t, z) \) or \( q^0(z \mid f, t) \). Indeed, this implementation uses \( \Theta(FT + FZ + TZ) \) memory (proportional to the sum of the sizes of the input and the output) total, but \( \Theta(FTZ) \) arithmetic operations per iteration.
2.3 Supervised NMF for single-channel audio source separation

To use NMF to separate $S$ audio sources, we posit that the mixture can be decomposed as a weighted sum over sources $s$:

$$ p_{\text{obs}}(f, t) \approx \sum_s q(s) \sum_z q(f \mid z, s)q(t, z \mid s), $$

as in Figure 1(b). Mathematically, this is equivalent to an NMF decomposition with inner dimension $ZS$; as yet there is no connection between the dictionary elements assigned to source $s$ making them relate to the same source in the recording.

To address this we require training data $\hat{p}_{\text{obs}}(f, t)$ corresponding to recordings of sounds typical of each of the sources alone. We apply NMF to these for each $s$ separately, learning a representative dictionary $\hat{q}_s(f \mid z)$. The factor $\hat{q}_s(t, z)$ represents when and how active these dictionary elements are in the training data, so it is discarded as irrelevant for separation.

We apply NMF again to learn (2) with the twist that we fix $q(f \mid z, s) := \hat{q}_s(f \mid z)$ for all iterations. The argument given in Section 2.1 still shows that the KL divergence cannot increase at each step. After NMF $q(s \mid f, t)$ gives a measure of the contribution from source $s$ in each time-frequency bin. A common use case is when $p_{\text{obs}}(f, t)$ is a normalized spectrogram, computed as the magnitude of a Short-Time Fourier Transform (STFT). It is typical to approximately reconstruct the time-domain audio of a separated source $s$ by multiplying the magnitude component $p_{\text{obs}}(f, t)q(s \mid f, t)$ with the phase of the mixture STFT, then taking the inverse STFT. Considering the outputs to be the mask $q(s \mid f, t)$ or reconstructed time-domain audio, separation takes $\Theta(FTS + FZS + TZS)$ memory total and $\Theta(FTZS)$ arithmetic operations per iteration.

2.4 Direction of Arrival (DOA) estimation

Estimating DOA from measurements at an array of sensors is a well-studied problem and the output of essentially any algorithm for it can be plugged into the framework of Section 3. We will the least squares method (perhaps the simplest) for estimating a DOA at each time-frequency bin. We take as given the STFTs of audio signals recorded at each of $M$ microphones. The same procedure is applied to all bins, so we focus on a single bin and its STFT values $Y_1, \ldots, Y_M$.

Assume this bin is dominated a single point source far enough away to appear as a plane wave and the array is small enough that wrapping of the phases $\angle Y_i$ is not an issue. Letting $x_i$ denote the position of microphone $i$ and $k$ the wave vector, we have $\angle Y_i - \angle Y_1 = (x_i - x_1) \cdot k$.

We solve these linear equations for $k$ in a least squares sense. The direction of $k$ serves as a DOA estimate for the chosen bin. Note that the coefficients of $k$ are fixed by the geometry, so the least squares problems for all time-frequency bins can be solved by taking a single pseudoinverse at design time and a small matrix multiplication for each bin thereafter.

---

1. Since the STFTs of separate sources are complex, they do not strictly speaking combine additively. However this is a common modeling assumption which tends to perform well and can be motivated by the fact that individual sources tend to be sparse in the time-frequency plane and so not overlap too much.

2. In fact $\hat{q}_s(t, z)$ contains information about evolution of the activations in time, which basic NMF ignores.
3 Adding Directionality to NMF With Nonnegative Tensor Factorization (NTF)

3.1 The Main Idea

This section is parallel to Section 2.1, except instead of a matrix $p_{\text{obs}}(f,t)$ we take as given an array or tensor $p_{\text{obs}}(f,t,d)$, interpreted as a distribution over time, frequency, and DOA. We treat DOA as quantized to a finite domain of size $D$, with strictly positive marginal $p_{\text{obs}}(d)$. We fit $p_{\text{obs}}(f,t,d) \approx \sum_{s,z} q(f,t,d,z,s)$ for the factorization

$$q(f,t,d,z,s) := q(s)q(f \mid s,z)q(t,z \mid s)q(d \mid s) = q(d,s)q(f \mid s,z)q(t,z \mid s),$$

represented in Figure 1(c). We allow a distribution $q(d \mid s)$ over DOA rather than a fixed DOA per source to allow for noise, slight movements of sources, and modeling error.

The crucial difference between the factorization (2) used for source separation with NMF and (3) is that the additional term $q(d \mid s)$ does not depend on $z$. This ties together dictionary elements corresponding to the same source, requiring them to come from the same direction. We can then learn the model (3) from $p_{\text{obs}}(f,t,d)$ alone, without training data.

It is possible to simultaneously account for sound coming from multiple real sources in the environment within a single source $s$ of the model (3) by choosing the corresponding $q(d \mid s)$ to be an appropriate multimodal DOA distribution. This says that for each dictionary element of source $s$, a comparable amount of energy is coming from the DOAs corresponding to each of the modes, which is by assumption not true. So the KL divergence between the data $p_{\text{obs}}(f,t,d)$ and the marginal $q(f,t,d)$ of the model would be lower if each source $s$ in the model described only one real source in the environment, as desired.

We can derive a Minorization-Maximization algorithm to fit the model (3) to $p_{\text{obs}}(f,t,d)$ just as we did to fit the model (1) to $p_{\text{obs}}(f,t)$ in Section 2.1. The resulting algorithm has the same alternating projections flavor. We begin with a factored model

$$q^0(f,t,d,z,s) := q^0(d,s)q^0(f \mid s,z)q^0(t,z \mid s),$$

force the desired marginal to obtain

$$r(f,t,d,z,s) := p_{\text{obs}}(f,t,d)q^0(z,s \mid f,t,d),$$

and finally return to factored form by defining

$$q^1(d,s) := r(d,s), \quad q^1(f \mid s,z) := r(f \mid s,z), \quad \text{and} \quad q^1(t,z \mid s) := r(t,z \mid s).$$

After iterating the desired number of times, we compute $q(s \mid f,t)$ and use this as a soft mask to separate the sources as in Section 2.3. The only difference is that without training data the model is symmetric under permutations of the sources. Thus the correspondence between sources learned by the model and sources in the environment is unknown a priori. However, the learned factors $q(d \mid s)$ show where in the environment the learned sources are located. This information may be useful for source selection.
### 3.2 Implementation

As in Section 2.2 we can turn these equations into multiplicative updates and in the process reduce the resource requirements. For example, we can calculate

\[
q^1(d, s) = \sum_{f,t,z} r(f, t, d, z, s) = \sum_{f,t,z} p_{\text{obs}}(f, t, d) \frac{q^0(f, t, d, z, s)}{q^0(f, t, d)}
\]

\[
= q^0(d, s) \sum_{f,t} \sum_{s'} q^0(d, s') \sum_{z'} q^0(f \mid z', s') q^0(t, z' \mid s') \sum_{z} q^0(f \mid z, s) q^0(t \mid s)
\]

\[
= q^0(d, s) \sum_{f,t} \sum_{s'} p_{\text{obs}}(f, t, d) q^0(f, t \mid s) q^0(f \mid t, s) = q^0(d, s) \sum_{f,t} \rho(f, t, d) q^0(f, t \mid s),
\]

as follows. Compute \(q^0(f, t \mid s)\); for each \(s\) this is an \(F \times T\) times \(Z \times T\) matrix multiplication. Then compute the denominator \(q^0(f, t, d)\) as a \(D \times S\) times \(S \times FT\) matrix multiplication. Divide \(p_{\text{obs}}(f, t, d)\) elementwise by the result and call this \(\rho(f, t, d)\). Compute the remaining sum as a \(D \times FT\) times \(FT \times S\) matrix multiplication. Multiply by \(q^0(d, s)\) elementwise to get \(q^1(d, s)\).

Similar computations yield \(q^1(f \mid z, s)\) and \(q^1(t, z \mid s)\); again the intermediate factor \(\rho(f, t, d)\) can be reused. The total memory required is \(\Theta(FTS + FTD + FZS + TZS)\), again proportional to the memory required to store the input and output (both the factorization and the mask \(q(s \mid f, t)\) are here considered part of the output). The number of arithmetic operations used at each iteration is \(\Theta(FTS(D + Z))\). So in addition to never having to allocate memory for any of the size \(FTDZS\) arrays referred to in Section 3.1 we do not even have to explicitly compute all their elements.

### 3.3 Sparse Direction Data

Suppose all the mass in each time-frequency bin is assigned to a single direction \(d(f, t)\), so \(p_{\text{obs}}(f, t, d) = p_{\text{obs}}(f, t) \delta(d = d(f, t))\) in terms of the Kronecker \(\delta\). The size of the input is then \(\Theta(FT)\) and the implementation simplifies further.

Since \(r(f, t, d, z, s)\) is only nonzero when \(d = d(f, t)\), we only need to explicitly compute the denominator \(q^0(f, t, d)\) of (4) for \(d = d(f, t)\). To do this, we compute \(q^0(f, t \mid s)\) as before and then substitute \(d = d(f, t)\) into \(q^0(d, s)\), yielding another \(F \times T \times S\) tensor \(q^0(d(f, t), s)\). Summing the elementwise product \(q^0(f, t \mid s') q^0(d(f, t), s')\) over \(s'\) yields the \(F \times T\) array \(q^0(f, t, d(f, t))\).

Instead of defining \(\rho(f, t, d)\) as in Section 3.2 we define \(\rho(f, t) := \frac{p_{\text{obs}}(f, t, d)}{q^0(f, t, d(f, t))}\), so

\[
r(f, t, d, z, s) = \rho(f, t) \delta(d = d(f, t)) q^0(d, s) q^0(f \mid z, s) q^0(t \mid s).
\]
Mean and min over separated sources run time as

| Algorithm         | SDR in dB | SIR in dB | SAR in dB | % real time |
|-------------------|-----------|-----------|-----------|-------------|
| Ideal Ratio Mask  | 14.4      | 19.9      | 16.0      | 1.5 %       |
| Ideal Binary Mask | 15.2      | 24.5      | 15.9      | 1.5 %       |
| **Directional NTF** | 12.1      | 20.3      | 13.4      | 25.6 %      |
| Directional NMF   | 3.2       | 9.5       | 8.0       | 17.2 %      |
| Supervised NMF    | 2.6       | 5.0       | 9.1       | 11.6 %      |

Table 1: Results of experiments described in Section 4 averaged over 1000 random instances. A range of ±0.5 is an (at least) 95% confidence interval for all true average BSS_EVAL metrics. Some algorithms often successfully isolate one source but not the other, so we record both the mean and the minimum of the metrics over the two separated sources for each instance. Runtimes are from python code (available with the arXiv version of this paper) on a 2012 Macbook Pro with a 2.7 GHz Intel Core i7 processor and 16 GB of RAM.

Marginalizing, we get:

\[ q^1(d, s) = \sum_{f, t, z} r(f, t, d, z, s) = q^0(d, s) \sum_{f, t: d(f, t) = d} \rho(f, t) q^0(f, t | s), \]

which can now be computed naively in \( \Theta(FTS) \) memory and arithmetic operations. For

\[ q^1(f, z, s) = \sum_{t, d} r(f, t, d, z, s) = q^0(f | z, s) \sum_t \rho(f, t) q^0(d(f, t), s) q^0(t, z | s) \]

we multiply \( \rho(f, t) \) by \( q^0(d(f, t), s) \), which takes \( \Theta(FTS) \) memory and operations, then compute the sum over \( t \) as \( S \) matrix multiplications of size \( F \times T \) times \( T \times Z \). This takes \( \Theta(FZS) \) memory and \( \Theta(FTZS) \) operations. Then we multiply elementwise by \( q^0(f | z, s) \) and condition to get \( q^1(f | z, s) \). The computation of \( q^1(t, z | s) \) is similar.

Altogether the resource requirements are \( \Theta(FTS + FZS + TZS) \) memory total and \( \Theta(FTZS) \) arithmetic operations per iteration.\(^3\) In particular, these are within a constant factor of supervised NMF (Section 2.3). This is somewhat of an apples and oranges comparison since the two algorithms have the same goal but different inputs: one uses direction information and the other uses clean audio training data. Still, the resource costs for using directionality information in an NTF framework are minimal relative to traditional NMF.

### 4 Experiments

To demonstrate the advantages of the algorithm in Section 3.3, we provide results in Table 1 on separating random TIMIT sentences [3] evaluated using the mir_eval implementation [7].

\(^3\)While it may be somewhat surprising that \( D \) does not appear in these bounds, the assumption that all \( D \) directions have positive probability along with sparsity implies \( D < FT \).
of the BSS.EVAL metrics \cite{10}. Each instance of the experiment was constructed as follows. We randomly selected two sentences from different speakers in the TIMIT test data set. A simple three-microphone array was simulated by mixing the two test files instantaneously and then with each file delayed by one sample relative to the other. This corresponds to an array with microphones at \((0, 0), (\Delta, 0)\) and \((0, \Delta)\) with \(\Delta = \frac{340.29 \text{m/s}}{16000 \text{Hz}} \approx 2.13 \text{cm}\) and far-field sources coming from the \(-x\) and \(-y\) axes.

We compare three source-separation algorithms: Directional NTF as in Section \textbf{3.3}; a less structured version we call Directional NMF, which consists of factoring \(p_{\text{obs}}(f, t, d) \approx \sum_s q(f, t \mid s)q(d, s)\); and Supervised NMF as in Section \textbf{2.3}. The first two methods receive only the three channels of mixed audio, while the third receives one channel of mixed audio and a different clean TIMIT sentence of training data for each speaker. All algorithms are set to extract two sources. Directional NTF and Supervised NMF each model sources with \(Z = 20\) dictionary elements, Directional NMF has no such parameter. Both directional methods receive a least-squares estimated DOA for each time-frequency bin (Section \textbf{2.4}) quantized to \(D = 24\) levels. For an upper bound we also compare two ideal oracle-based methods of computing masks.

5 Conclusions

Directional NTF is better than the other (non-oracular) algorithms according to all BSS.EVAL metrics (Table 1). We close with directions for future work.

First, this method fits naturally into the basic NMF / NTF framework. As such it should be extensible using any of the many improvements to these methods available in the literature.

Second, when separating speech from background noise using Directional NTF, how should we determine which source is speech? In some applications one may be able to infer this from the centers of the learned distributions \(q(d \mid s)\) and prior information about the location of the speaker. In other applications one may expect diffuse noise and call the source with \(q(d \mid s)\) more tightly peaked the speaker. Is there a more generally-applicable method?

Third, the results of Table 1 show that this method works well in one specific case. Our informal trials on real reverberant array recordings suggest the same perceptual improvements, but the results are harder to quantify. We leave a rigorous analysis of performance as a function of geometry and reverberation for future work. In particular, what are the fundamental limits on separating closely-spaced sources using this method?

Acknowledgements

The author would like to thank Paris Smaragdis, Johannes Traa, Théo Weber, and David Wingate for their helpful suggestions regarding this work.
References

[1] J. Benesty, J. Chen, and Y. Huang. *Microphone Array Signal Processing*, volume 1 of *Topics in Signal Processing*. Springer, 2008.

[2] Derry FitzGerald, Matt Cranitch, and Eugene Coyle. Non-negative tensor factorisation for sound source separation. In *Proceedings of the Irish Signals and Systems Conference*, September 2005.

[3] John Garofolo et al. TIMIT acoustic-phonetic continuous speech corpus LDC93S1. Web Download. Philadelphia: Linguistic Data Consortium, 1993.

[4] Daniel D. Lee and H. Sebastian Seung. Algorithms for Non-negative Matrix Factorisation. In *Advances in Neural Information Processing Systems*, volume 13, pages 556–562. MIT Press, 2000.

[5] Seokjin Lee, Sang Ha Park, and Koeng-Mo Sung. Beamspace-domain multichannel nonnegative matrix factorization for audio source separation. *IEEE Signal Processing Letters*, 19(1):43–46, January 2012.

[6] Alexey Ozerov and Cédric Févotte. Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):550–563, March 2010.

[7] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawan Liang, and Daniel P. W. Ellis. mir_eval: A transparent implementation of common MIR metrics. In *Proceedings of the 15th International Conference on Music Information Retrieval*, 2014.

[8] Bhikshya Raj and Paris Smaragdis. Latent variable decomposition of spectrograms for single channel speaker separation. In *Proceedings of WASPAA*, 2005.

[9] Johannes Traa. Multichannel source separation and tracking with phase differences by random sample consensus. Master’s thesis, University of Illinois, 2013.

[10] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1462–1469, 2006.