Non-Negative Tensor Factorisation for Sound Source Separation

Derry FitzGerald, Matt Cranitch[ϕ] **and Eugene Coyle***

[ϕ]*Dept. of Electronic Engineering, Cork Institute of Technology Rossa Avenue, Bishopstown, Cork, Ireland E-mail:* [ϕ] *derry.fitzgerald@cit.ie*

** Department of Electronic Engineering, Dublin Institute of Technology Kevin St, Dublin, Ireland E-mail: *eugene.coyle@dit.ie*

Abstract – **An algorithm for Non-negative Tensor Factorisation is introduced which extends current matrix factorisation techniques to deal with tensors. The effectiveness of the algorithm is then demonstrated through tests on synthetic data. The algorithm is then employed as a means of performing sound source separation on two channel mixtures, and the separation capabilities of the algorithm demonstrated on a two channel mixture containing saxophone, strings and bass guitar.**

Keywords – **Non-negative tensor factorisation, sound source separation.**

I MATRIX FACTORISATION TECHNIQUES

In recent years, sparse matrix factorisation techniques have been used to attempt sound source separation, with a focus on single channel sound source separation. Various methods of sparse matrix factorisation have been proposed for this task, including Independent Subspace Analysis (ISA) [1], Non-Negative Sparse Coding (SC) [2] and Non-negative Matrix Factorisation (NMF) [3]. These techniques typically take as input a magnitude or power spectrogram **X** of size *n* x *m*, and attempt to decompose **X** into matrix factors **A** and **S**, such that

$$\mathbf{X} \approx \hat{\mathbf{X}} = \mathbf{A}\mathbf{S}^T \qquad (1)$$

where $\hat{\mathbf{X}}$ is an approximation to **X**, **A** is an *n* x *k* matrix, **S** is an *m* x *k* matrix, with *r* smaller than *n* or *m*, where *k* is the chosen rank of the decomposition and ^{T} denotes matrix transpose. This results in a compressed version of the original spectrogram. While a factorisation such as described in equation 1 could be obtained by singular value decomposition (SVD), using SVD results in a factorisation where the energy of the factors is spread across the frequency range. This does not reflect the fact that most musical instruments are harmonic in nature and so can be represented efficiently as sparse factors. The use of sparseness as a factorisation criterion encourages the recovery of factors which reflect this situation. When *k* is chosen correctly, this can result in a factorisation where the elements of **A** and **S** reveal the underlying structure of the input spectrogram.

The principal difference between the above mentioned sparse matrix factorisation methods lies in how the decomposition is achieved. ISA makes use of SVD to obtain a reduced rank approximation to the original spectrogram, followed by Independent Component Analysis [4] to obtain a set of independent and sparse factors. SC attempts to balance the reconstruction of the spectrogram with the sparseness of the recovered factors, with additional constraints to ensure the non-negativity of the factorisation. NMF makes use of a generalised Kullback-Liebler divergence between the spectrogram and the reconstruction of the spectrogram, and uses multiplicative updates to ensure the non-negativity of the factorisation. The divergence used is:

$$D\left(\mathbf{X}\middle\|\hat{\mathbf{X}}\right) = \sum_{i,j} \mathbf{X}\log\frac{\mathbf{X}}{\hat{\mathbf{X}}} - \mathbf{X} + \hat{\mathbf{X}} \qquad (2)$$

where *i* and *j* index over the frequency bins and time frames of the spectrogram respectively. This cost function is equivalent to assuming a Poisson noise model for the input spectrogram. The addition of non-negativity to the factorisation is important in that, by its nature, a magnitude or power spectrogram contains only non-negative data, and so a non-negative factorisation is more likely to model accurately the data presented.

After factorisation, the columns of **A** contain frequency basis functions, while the rows of **S** contain a corresponding set of amplitude basis functions. In the case of pitched musical signals, it has been observed when *r* has been chosen properly, the basis functions correspond to notes or chords played by the instruments present [2], [3].

As observed above, each basis function typically contains a note or chord played by a given instrument. This means that for instruments that play melodies some method of grouping the notes to their respective instruments is required for source separation to succeed. Grouping methods have been proposed by Casey [1] and Virtanen [2], but in many cases it is difficult to obtain a correct clustering for reasons described in [5].

As a result, extended methods have been proposed to try and overcome this problem, such as shifted NMF [6]. In shifted NMF, instruments are represented by a single frequency basis function, and notes played by an instrument are represented as translations of the instrument frequency basis function. It should be noted that this technique requires the use of logarithmic frequency resolution. Another technique is that of non-linear ISA which represents chord spectra as sums of note power spectra, and note spectra as sums of instrument dependent log-power spectra [7]. Unlike shifted NMF, non-linear ISA requires the use of trained instrument priors to separate the signals.

II Stereo Signal Model

As noted above, matrix factorisation techniques have been used for sound source separation of single channel mixtures. However, most musical recordings from the past 40 years are stereo, or two channel recordings. In most cases, these recordings have been created by recording each instrument individually. The recordings of the instruments are then summed and distributed (or panned) across the two channels. This results in a situation where for any individual instrument, the only difference between the 2 channels lies in the intensity of the instrument. This fact has been used for sound source separation by Barry et al. in [8]. The mixing model can be mathematically described, after Barry et al., as follows:

$$L(t)=\sum_{j=1}^{J} Gl_j S_j(t) \tag{3}$$

$$R(t)=\sum_{j=1}^{J} Gr_j S_j(t) \tag{4}$$

where S_j are the J independent sources, and where Gl_j and Gr_j are the gains for each source for the left and right channels, and L and R are the resulting channel mixtures.

As observed above, the only difference between the 2 channels for a given instrument lies in the intensity of each source. Therefore, the same frequency basis function could be used to describe a note or chord from a given instrument in either channel, the only difference lying in the gain of the basis function in each channel. Therefore, it is proposed to learn a single set of frequency basis functions which can be used to describe both channels of the input signal, a corresponding set of amplitude basis functions, and a set of corresponding gains which decide how loud a given pair of frequency and amplitude basis functions are in each channel. These gains can then be used to group the basis functions to their sources. The signal model can then be written as:

$$\mathcal{X} \approx \hat{\mathcal{X}} = \sum_{k=1}^{K} \mathbf{G}_{:k} \circ \mathbf{A}_{:k} \circ \mathbf{S}_{:k} \tag{5}$$

where $\mathcal{X}$ is a 2 x n x m tensor containing the spectrograms of the two channels, $\hat{\mathcal{X}}$ is an approximation to $\mathcal{X}$, **G** is a 2 x k matrix containing the gains of each factor in each channel, **A** is an n x k matrix containing the frequency basis functions, and **S** is an m x k matrix containing the amplitude basis functions, $\circ$ denotes outer product multiplication, and $:k$ denotes the kth column of a given matrix.

III Non-negative Tensor Factorisation

The signal model in equation 5 describes a tensor factorisation. Algorithms for tensor factorisation such as PARAFAC and multilinear SVD have been in existence for quite some time [9], [10]. However, these factorisation algorithms are analogous to SVD in the sense that the energy of the factors gets spread across the frequency range, and so are unlikely to recover meaningful factors. Further, these decompositions do not reflect the non-negativity of the spectrograms to be factorised. Therefore, it is proposed to perform a non-negative tensor factorisation on $\mathcal{X}$.

For the remainder of the paper the following conventions are used. Tensors are denoted by upper case letters such as $\mathcal{X}$, and contracted tensor product multiplication is defined as follows. If $\mathcal{W}$ is a tensor of size I_1 x ··· x I_N x J_1 x ··· x J_M and $\mathcal{Y}$ is a tensor of size I_1 x ··· x I_N x K_1 x ··· x K_P then contracted tensor multiplication along the first N modes is given as:

$$\langle \mathcal{W}\mathcal{Y} \rangle_{\{1:N,1:N\}}(j_1,\ldots,j_M,k_1,\ldots,k_P) = \sum_{i_1=1}^{I_1}\cdots\sum_{i_N=1}^{I_N} \mathcal{W}(i_1,\ldots,i_N,j_1,\ldots,j_M)\; \mathcal{Y}(i_1,\ldots,i_N,k_1,\ldots,k_P) \tag{6}$$

where element indexing occurs within ( ) brackets, and where the modes to be multiplied are specified in the subscripts within the angle brackets. This is in line with the conventions adapted by Bader and Kolda in [11]

The concept of non-negative tensor factorisation was introduced by Shashua and Hazan in [12], where they presented an algorithm based on a least squares based factorisation of the input tensor. Here we present an algorithm derived from extending the cost function in equation 2 to tensors. The cost function is now given as:

$$D\left(\mathcal{X}\middle\|\hat{\mathcal{X}}\right)=\sum_{i,j,l}\mathcal{X}\log\frac{\mathcal{X}}{\hat{\mathcal{X}}}-\mathcal{X}+\hat{\mathcal{X}} \qquad (7)$$

where *l, i*, and *j* index over channel, frequency bin and time frame respectively. Eliminating terms in $\mathcal{X}$ which are constant, and substituting for equation 5 yields:

$$D\left(\mathcal{X}\middle\|\hat{\mathcal{X}}\right)=\sum_{i,j,l}-\mathcal{X}\log\left(\sum_{k=1}^{K}\mathbf{G}_{:k}\circ\mathbf{A}_{:k}\circ\mathbf{S}_{:k}\right) + \sum_{k=1}^{K}\mathbf{G}_{:k}\circ\mathbf{A}_{:k}\circ\mathbf{S}_{:k} \qquad (8)$$

Taking the gradient with respect to **G** yields the following update equation:

$$\mathbf{G}=\mathbf{G}+\lambda\left\lfloor\langle\mathcal{P}\mathcal{D}\rangle_{\{2:3,1:2\}}-\langle\mathcal{P}\mathcal{O}\rangle_{\{2:3,1:2\}}\right\rfloor \qquad (9)$$

where $\mathcal{P}$ is a tensor of size *n* x *m* x *k* and where

$$\mathcal{P}(:,:,k)=\mathbf{A}_{:k}\circ\mathbf{S}_{:k} \qquad (10)$$

$$\mathcal{D}=\mathcal{X}./\hat{\mathcal{X}} \qquad (11)$$

and where $\mathcal{O}$ is an all-ones tensor of size equal to $\mathcal{X}$. Elementwise division is denoted by ./ . Equation 9 can be converted into a multiplicative update rule by setting λ equal to:

$$\lambda=\mathbf{G}./\langle\mathcal{P}\mathcal{O}\rangle_{\{2:3,1:2\}} \qquad (12)$$

The multiplicative update rule is then given by:

$$\mathbf{G}=\mathbf{G}.*\left\lfloor\langle\mathcal{P}\mathcal{D}\rangle_{\{2:3,1:2\}}./\langle\mathcal{P}\mathcal{O}\rangle_{\{2:3,1:2\}}\right\rfloor \qquad (13)$$

where .* denotes elementwise multiplication. Update equations for **A** and **S** can be derived in a similar manner and are given by:

$$\mathbf{A}=\mathbf{A}.*\left\lfloor\langle\mathcal{Q}\mathcal{D}\rangle_{\{[1,3],1:2\}}./\langle\mathcal{Q}\mathcal{O}\rangle_{\{[1,3],1:2\}}\right\rfloor \qquad (14)$$

where $\mathcal{Q}$ is a tensor of size 2 x *m* x *k* and where

$$\mathcal{Q}(:,:,k)=\mathbf{G}_{:k}\circ\mathbf{S}_{:k} \qquad (15)$$

$$\mathbf{S}=\mathbf{S}.*\left\lfloor\langle\mathcal{R}\mathcal{D}\rangle_{\{1:2,1:2\}}./\langle\mathcal{R}\mathcal{O}\rangle_{\{1:2,1:2\}}\right\rfloor \qquad (16)$$

where $\mathcal{R}$ is a tensor of size 2 x *n* x *k* and where

$$\mathcal{R}(:,:,k)=\mathbf{G}_{:k}\circ\mathbf{A}_{:k} \qquad (17)$$

The use of multiplicative updates ensures that once **G**, **A**, and **S** are randomly initialised to positive values the factorisation will be non-negative. Although the convergence proofs used for NMF (see [13]) no longer apply, in practice it has been observed that the algorithm converges reliably. Though the derivation presented is for a 3-D tensor, it can be easily extended to deal with higher order tensors. Though it is proposed to use the algorithm for the purposes of sound source separation, the algorithm is potentially useful in other areas such as image analysis and chemometrics, and in the processing of multivariate data in general.

The NTF algorithm was implemented in Matlab using the Matlab Tensor Classes developed by Bader and Kolda, which are available from [14]. To demonstrate the effectiveness of NTF algorithm, a tensor of size 2 x 264 x 100 was created from a set of factors with *K* = 2. Figure 1 shows the normalised factors used to create the tensor, while Figure 2 shows the factors recovered by the NTF algorithm. **G**, **A**, and **S** were all randomly initialised to positive values. The algorithm converged after 100 iterations. It can be seen that the algorithm has successfully recovered the underlying factors used to generate the data.

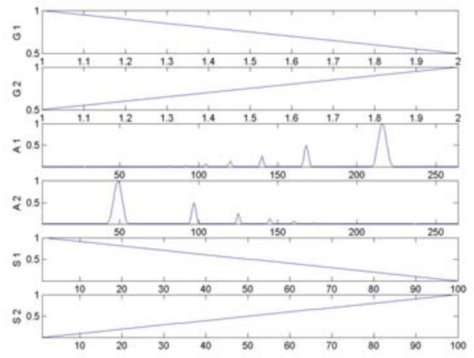

Figure 1: Factors used to create synthetic tensor.

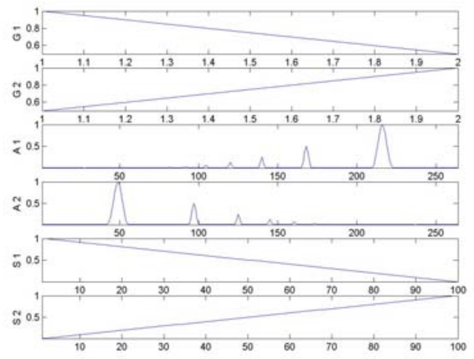

Figure 2: Factors recovered by the NTF algorithm.

IV SOUND SOURCE SEPARATION USING NTF

Having demonstrated that the NTF algorithm performs as intended, there remains to show its effectiveness as a means of sound source separation. To enable automatic separation of the sources requires the addition of a number of additional steps, namely the clustering of the gain values obtained, and the creation of source spectrograms based on the results of the clustering. As the NTF algorithm works on magnitude or power spectrograms, a set of phase information must be generated to invert the spectrograms to time-domain waveforms. To this end, the original phase information of the spectrogram in which the source is dominant is used. This has proved to be the most effective method of obtaining phase information for inverting magnitude spectrograms of this nature [15].

The algorithm proposed for automatically separating the sources is as follows:

1. Obtain a spectrogram for each of the two input channels.
2. Combine the two spectrograms into a tensor and perform NTF on the tensor.
3. Determine log intensity ratio of the factors:
   $$H = \log\left[\mathbf{G}(1,:)/\mathbf{G}(2,:)\right] \qquad (18)$$
4. Cluster H into J clusters, where J is the number of sources.
5. Estimate source spectrogram from:
   $$\mathbf{S} = \mathbf{A}(:,a)\mathbf{S}(:,a)^T \qquad (19)$$
   where a is a vector containing the indices of the factors associated with the jth source.
6. Apply phase information from the spectrogram where the source is dominant to $\mathbf{S}$.
7. Invert the spectrogram to obtain the time domain waveform.
8. Repeat steps 5-7 for each of the J sources.

In this case, the clusters were created using a k-Nearest Neighbours approach, though other approaches could also be used. To demonstrate the use of the algorithm for sound source separation, a two channel mixture containing strings, saxophone, and bass guitar was created. The strings had an intensity ratio of 2:1 between the two channels, the bass guitar, a ratio of 1:1 and the saxophone had a ratio of 1:2. The number of factors to be recovered was set to 14, and the algorithm had again converged after 100 iterations. Figure 3 shows the two channel mixture input to the algorithm, while figure 4 shows the original waveforms for bass, saxophone and strings used to create the two channel mixture. Figure 5 shows the separated waveforms obtained from the separation algorithm.

It can be seen that the sources have been separated reasonably well, with the main characteristics of the sources having been captured. On listening to the separated waveforms, traces of the strings can be heard in the bass guitar, but the bass predominates. Similarly traces of the bass guitar can be heard on both the separated saxophone and string waveforms, but again the audio separation is quite good, with the respective instruments predominating in all cases. It should be noted that the best separation in terms of audio quality occurred for the saxophone, followed by bass guitar, and then strings, which is in line with what is visible in figures 4 and 5. This demonstrates that the proposed algorithm can perform automatic sound source separation when the underlying assumptions of the algorithm have been met.

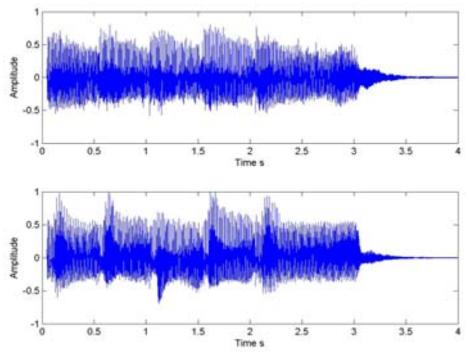

Figure 3: Input two channel mixture of strings, bass guitar, and saxophone.

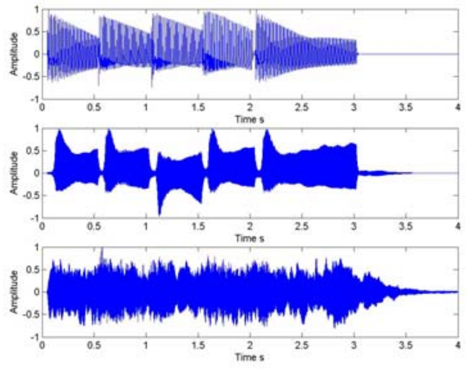

Figure 4: Original waveforms for bass guitar, saxophone and strings respectively.

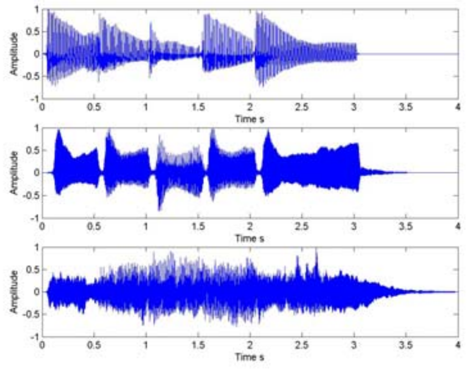

Figure 5: Separated waveforms for bass guitar, saxophone and strings respectively.

The principal problem with the algorithm as presently implemented is that the number of factors K has to be set by hand. At present, there is no method for automatically estimating the number of factors required. This is a problem for all factorisation algorithms at present.

The algorithm presented works well when each factor recovered represents a single note of a given instrument. However, setting K too low can result in a factorisation where multiple notes from different instruments are approximated by a single factor, which in turn leads to incorrect separation. Setting K too high results in a situation where elements of notes are spread across several factors, and this can cause problems at the grouping stage. The best results are obtained when K is set approximately equal to the sum of the number of different notes played by all the instruments. A further problem is that clustering becomes more difficult as the number of sources increases. Nevertheless, the method presented does represent a new way of attempting sound source separation from two channel mixtures.

V CONCLUSIONS

An algorithm for performing Non-negative Tensor Factorisation was presented which extends present matrix factorisation techniques to deal with tensors. The effectiveness of the NTF algorithm was demonstrated using synthetic test data. The NTF algorithm was then incorporated into a sound source separation algorithm which was shown to be capable of separation mixtures of instruments from a two channel mixture. It is intended to improve the performance of the algorithm through the use of perceptual weighting, which has been shown to improve the performance of matrix factorisation techniques when used for sound source separation [16], and by investigating ways of automatically estimating the number of factors required.

REFERENCES

[1] Casey, M.A. & Westner, A., “Separation of Mixed Audio Sources By Independent Subspace Analysis” in Proc. Of ICMC 2000, pp. 154-161, Berlin, Germany.

[2] T. Virtanen, “Sound Source Separation Using Sparse Coding with Temporal Continuity Objective”, Proc. of International Computer Music Conference (ICMC2003), Singapore, 2003.

[3] P. Smaragdis, J.C. Brown, "Non-negative Matrix Factorization for Polyphonic Music Transcription", IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pp. 177-180, October 2003.

[4] A. Hyvärinen and E. Oja. “Independent Component Analysis: Algorithms and Applications”. *Neural Networks*, 13(4-5): pp 411-430, 2000.

[5] D. FitzGerald, “Automatic Drum Transcription and Source Separation”, PhD. Thesis, Dublin Institute of Technology, 2004.

[6] D. FitzGerald, M. Cranitch, and E. Coyle, “Shifted Non-negative Matrix Factorisation for Sound Source Separation”, Statistics in Signal Processing Conference, Bordeaux, France, 2005.

[7] E. Vincent and X. Rodet, “Music transcription with ISA and HMM.” In Proceedings of ICA , 2004. September 22-24, 2004, Granada, Spain, pages 478-485.

[8] Barry, D., Lawlor, R. and Coyle E., “Sound Source Separation: Azimuth Discrimination and Resynthesis”, Proc. 7th International Conference on Digital Audio Effects, DAFX 04, Naples, Italy, 2004.

[9] Richard A. Harshman. “Foundations of the PARAFAC procedure: Models and conditions for an “explanatory” multi-modal factor analysis.” UCLA working papers in phonetics, 16:1–84, 1970.

[10] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. “A multilinear singular value decomposition.” SIAM J. Matrix Anal. Appl., 21(4):1253–1278, 2000.

[11] B. Bader and T. Kolda, “MATLAB Tensor Classes for Fast Algorithm Prototyping”, Technical Report SAND2004-5187, Sandia National Laboratories, Livermore, California, 2004.

[12] A. Shashua and T. Hazan. “Non-Negative Tensor Factorization with Applications to Statistics and Computer Vision.” International Conference of Machine Learning (ICML), Aug. 2005.

[13] D. Lee, and H. Seung, “Algorithms for non-negative matrix factorization.” Adv. Neural Info. Proc. Syst. 13, 556-562 (2001).

[14] Tensor Classes for Matlab, available at http://csmr.ca.sandia.gov/ tgkolda/

[15] Barry, D., Lawlor, R. and Coyle E., “Comparison of Signal Reconstruction Methods for the Azimuth Discrimination and Resynthesis Algorithm”, 118th AES Convention Barcelona, Spain, 2005.

[16] T. Virtanen, “Separation of sound sources by convolutive sparse coding”, Proceedings of the ISCA Tutorial and Research Workshop on Statistical and Perceptual Audio Processing, 2004.