Source separation using NMF for mixtures of musical notes

Milind Kumar V[1]

Abstract— **Blind signal separation or source separation refers to the separation of each of the individual sources from a mixture of sources given little or no prior information about the source components. This has been a topic of great study and several techniques relying on matrix methods exist for this task. Non-negative matrix factorizatoin (NMF henceforth) is one such technique popularly used for its representational power and ease of implementation. In this work, NMF is introduced, the central algorithm based on multiplcative update rules is discussed and a comparison is drawn between local implementations of the algorithm and the implementation in scikit-learn. Following this, variants of the orignal algorithm utilising priors of sparseness and temporal continuity are used to perform source separation on mixtures of musical notes. The variation of the signal to noise ratio (SNR) is studied for different variations of NMF, different windows and window lengths and different weights of the sparseness and temporal continuity cost functions.**

I. INTRODUCTION

Source separation refers to the separation of components from a single signal containing a mixture of these. Typically, most source separation algorithms act on the audio represented in the frequency domain. Consequently, in the frequency domain, this involves the recovery of both the magnitude and phase spectra of the components constituting the given signal. While this work is mostly centered around the analysis of audio signals, images shall briefly be considered when considering the representational power of the algorithm under consideration. There exist several techniques from linear algebra that can be used for the purpose of source separation such as indpenedent component analysis, independent subspace analysis, principal component analysis, vector quantization and singular value decomposition. This work explores the use of non-negative matrix factorization (NMF) and its variants in source separation and attempts to draw a comparison between them. A brief introduction to NMF is presented in Section II. This work solely studies the multiplicative updates proposed by Lee and Seung in [1].

Source separation has several applications specially in audio signal processing such as the separation of audios corresponding to individual speakers from an audio of multiple speakers speaking at the same time i.e the cocktail party problem and speech enhancement by the removal of noise from signals. Further, NMF is extremely popular as a tool because of its simplicity, performance and ease of implementation. Further, the representational power of NMF discussed in Section II-B contributes to its usage in source separation.

[1]Milind Kumar V is a senior undergraduate student at IIT Madras, Chennai, Tamil Nadu, India. milind.blaze9@gmail.com

This work introduces NMF, discusses the reason for its popularity, presents the multiplicative update rule algorithm in [1] and Python code for the same. It also draws a comparison between the presented implementation and the implementation in standard Python libraries such as scikit-learn. Following this, some applications of NMF are discussed and some results on application of NMF to images are presented. A variant of this technique employing sparseness and temporal continuity priors [2] is then applied to mixtures of musical notes and the derivation of the update rules, the effects of hyperparameter tuning, the effect of the window used during data processing are analyzed and presented. Following this, other problems to be solved and some suggested approaches are presented.

II. NON-NEGATIVE MATRIX FACTORIZATION

Consider a matrix V. V is said to be non-negative if every element of V is non-negative. NMF refers to factorizing V approximately as the product of two matrices W and H, each of which is non-negative, i.e

$$V \approx WH \tag{1}$$

Such a factorization affords one the opportunity to interpret the columns of W as basis vectors and the columns of H as the corresponding weights. If the matrix V is the magnitude of the STFT of a given audio, the columns of W can be interpreted as underlying components and the elements of the j^{th} column of H as the weights of the components in the j^{th} time frame. Clearly, this matrix factorization is not unique. Further, metrics are required to determine the quality of the approximations. In this work, we shall look at the Frobenius norm defined as

$$||A - B||^2 = \sum_{ij} (A_{ij} - B_{ij})^2 \tag{2}$$

and the generalized KL divergence

$$D(A||B) = \sum_{ij} \left(A_{ij} log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij} \right) \tag{3}$$

These are loss functions that capture the differnce between the source and the reconstructed matrix and are lower for better reconstructions. These are exactly zero when the reconstructed matrix exactly matches the original matrix.

A. Update rules

[1] presents an elegant derivation minimising auxiliary functions that form upper bounds for the functions in Equations 3 and 2 to arrive at the multiplicative update rules. For Frobenius norm, these rules can be written as:

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^T V)_{a\mu}}{(W^T W H)_{a\mu}} \tag{4}$$

$$W_{ia} \leftarrow W_{ia} \frac{(V H^T)_{ia}}{(W H H^T)_{ia}} \tag{5}$$

These can be vecotrised easily as shown below and consequently makes for highly efficient code[1]

$$H \leftarrow H \odot \frac{W^T V}{W^T (W H)} \tag{6}$$

$$W \leftarrow W \odot \frac{V H^T}{(W H) H^T} \tag{7}$$

Similarly, the update rules for KL divergence too can be vectorised as below

$$H \leftarrow H \odot \frac{W^T \frac{V}{WH}}{W^T \mathbf{1}} \tag{8}$$

$$W \leftarrow W \odot \frac{\frac{V}{WH} H^T}{\mathbf{1} H^T} \tag{9}$$

where $\mathbf{1}$ is a matrix of all ones of the same size as the matrix V. The code[1] however uses a slightly diffwerent implementation of the update rules. These rules are iterative update rules and need to be applied consecutively in an iteration, i.e W is updated first followed by H. The loss function was seen to diverge in a few cases when both the matrices were updated simultaneously. More implementation details and a discussion of the initialization to be used, different NMF algorithms and applications of NMF can be found in [3]. [1] provides theoretical justification for the use of these rules. The validity of these rules was further verified using by performing NMF on a matrix containing entries drawn from a uniform distribution between 0 and 1. The resulting loss curves are presented in Figures 1 and 2.

There is no perceptible difference between the performance of the direct python implementation of equations and the implementation in a standard library such as scikit-learn. Errors due to division by zero were avoided by adding a fraction of the machine epsilon to each term in the non-negative matrix to be factorized. This was tested against scikit-learn and addition of a thousandth of the smallest value of the input matrix and did not produce any significant difference in the output produced or the behaviour of the loss function with number of iterations.

[1] https://github.com/Milind-Blaze/source_separation/blob/master/sourcesep/sourcesep.py

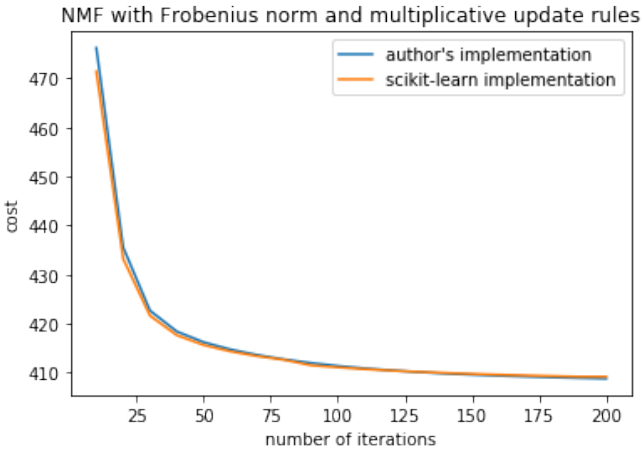


Fig. 1: Variation of loss using Equations 6 and 7

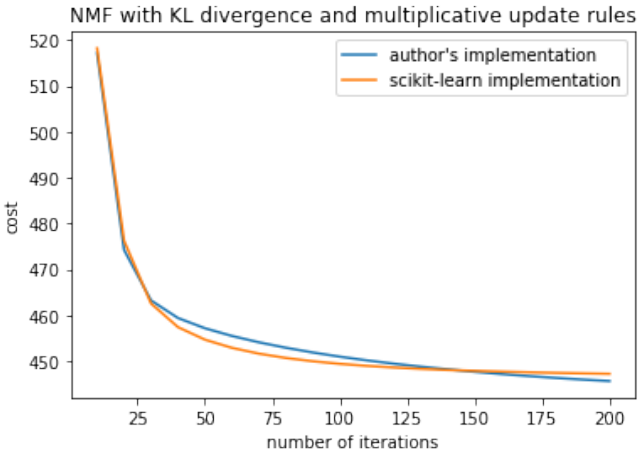


Fig. 2: Variation of loss using Equations 8 and 9

B. Representational power

One of the reasons why NMF is so popular is that it produces highly interpretable factorizations. Algorithms like vector quantization take a very hard winner take all approach and approximate each source signal with at most one signal. Algorithms like PCA produced distributed representations but these representations are not interpretable. A detailed study of the representational capabilities of these three techniques is presented in [4]. As a part of this work, the applicability of NMF in learning parts that can be used to constitute full signals is studied by decomposing the images present in the CBCL dataset, MIT Center For Biological and Computation Learning as done in [3] and [4]. These imaages are 19×19 each nine of which are presented in 3. These are flattened out into vectors of dimension 361 and are arranged in a matrix. The component vectors obtained when NMF is performed on this can then be reshaped and treated as constituent images as shown in Figure 4. The features thus plotted clearly possess discernible facial features- such as a smiling mouth, a nose, eyes and a jaw and so forth. In contrast, features learnt by PCA as shown in [4] do not form

Fig. 3: Images from the CBCL face dataset

Fig. 5: Images reconstructed after factorization

any discernible features.Further, the reconstructed images to can be plotted similarly. Figure 5 presents the reconstructed images corresponding to those present in Figure 3.

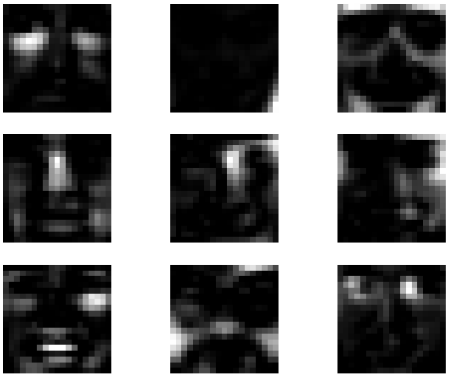


Fig. 4: Column vectors of the matrix W reshaped to form basis images

III. Source separation for musical audios

This portion of the work shall primarily focus on applying NMF to mixtures of musical notes. NMF with both the loss functions is applied to the audios under consideration followed by the algorithm presented in [2]. Much of the subsequent work is empirical and revolves around the use of temporal continuity and sparseness criteria as presented in [2] (referred to henceforth as original work) to improve the quality of separation.

A. Data

The data contains 7s musical audios created by mixing pitched sounds and drum beats. All audios are sampled at 44100 HZ. At the time of writing, several of the audios sources used by the author of the original work were unavailable for usage. Consequently, only the audios supplied at http://www.cs.tut.fi/ tuomasv/demopage.html were used as a part of this work. Much of the preliminary work presented utilises only one of the four supplied audios. Since the original work employed a large number of audios, it is reasonable to expect the results produced to be different. However, the general trend in the behaviour of the algorithm is assumed to be the same.

B. Processing

The considered audio is split into segments, windowed and the complex spectrogram is extracted for each audio considered. The overlap between successive segements is 50% for all the experiments performed and is not varied. On the other hand, the number of components (columns of the matrix W, r henceforth) and the window used are extensively experimented with. This is then separated into the magnitude and phase spectra. Much of this work deals only with the magnitude spectrum V of the audio. NMF is then performed on V to obtain the matrices W and H. The magnitude spectrogram of the i^{th} component is reconstructed as the outer product of the i^{th} column of W and the i^{th} row of H. The quality of this reconstruction can be improved by using sparsity and temporal continuity criteria on top of vanilla NMF. The complex spectrogram of the component is obtained by performing the Hadamard product of the magnitude spectrogram with phase of the orignial signal. The audio can be reconstructed by considering the ISTFT of this product. Components are clustered together using the original sources used to create the mixture. Each component is allocated to that source with which it has the highest SNR as given by Equation 10.

$$\text{SNR} = \frac{\sum_{k,t}[Y]_{k,t}^2}{\sum_{k,t}([Y]_{k,t} - [\hat{Y}]_{k,t})^2} \tag{10}$$

Here, Y is the original (reference) spectrogram and $\hat{Y}$ is the reconstructed spectrogram. The same metric is used to evaluate the quality of the reconstructed audio with respect to the original audio. The implementation [2] uses the librosa package from Python.

C. Adding temporal continuity and sparseness priors [2]

Section III-E presents the performance of plain NMF for source separation in the given audio. However, the algorithm presented in the original work can be shown to outperform

[2] https://tinyurl.com/y9ldovvq

vanilla NMF. This algorithm adds two additional terms to the KL divergence loss function that help make the outputs sparse and gains of an audio over different time frames more continuous. Divergence is used as the reconstruction error term c_r as it is linear in scaling (see Equation 11 and thus is sensitive to observations at lower energies.

$$D(ap||aq) = aD(p||q) \tag{11}$$

The temporal continuity cost function is defined as below:

$$c_t = \sum_{j=1}^{J} \frac{1}{\sigma_j^2} \sum_{t=2}^{T} \left(h_{t,j} - h_{t-1,j}\right)^2 \tag{12}$$

where σ_j is defined as

$$\sigma_j = \sqrt{\frac{1}{T} \sum_{t=1}^{T} h_{t,j}^2} \tag{13}$$

Minimising C_t ensures that the gains of the j^{th} component do not vary wildly over consecutive time frames and that the variation is relatively smooth. Further a sparseness cost function is also added which is defined as follows

$$c_s = \sum_{j=1}^{J} \sum_{t=1}^{T} \left| \frac{h_{t,j}}{\sigma_j} \right| \tag{14}$$

The update rule for the matrix W remains the same as Equation 9. However, the update rules for the matrix H are obtained by considering the ratio of the positive and negative terms in the gradients of the cost function given by Equations 15 - 20. It is worth noting that this gradient computation is only a technique that works well in this case and unfortunately has not theoretical backing.

$$\nabla c_r^+(W,H) = W^T 1 \tag{15}$$

$$\nabla c_r^-(W,H) = W^T \frac{V}{WH} \tag{16}$$

$$[\nabla c_t^+(H)]_{j,t} = \frac{4T h_{j,t}}{\sum_{i=1}^{T} h_{j,i}^2} \tag{17}$$

$$[\nabla c_t^-(H)]_{j,t} = 2T \frac{h_{j,t-1} + h_{j,t+1}}{\sum_{i=1}^{T} h_{j,i}^2} + \frac{2T h_{j,t} \sum_{i=2}^{T} (h_{j,i} - h_{j,i-1})^2}{\sum_{i=1}^{T} h_{j,i}^2} \tag{18}$$

$$[\nabla c_s^+(H)]_{j,t} = \frac{1}{\sqrt{\frac{1}{T} \sum_{i=1}^{T} h_{j,i}^2}} \tag{19}$$

$$[\nabla c_s^-(H)]_{j,t} = \frac{h_{j,t} \sqrt{T} \sum_{i=1}^{T} h_{j,i}}{\left(\sum_{i=1}^{T} h_{j,i}^2\right)^2} \tag{20}$$

The loss function is defined as

$$c = c_r + \alpha c_t + \beta c_s \tag{21}$$

The update is given by

$$H \leftarrow H \odot \frac{\nabla c_r^- + \alpha \nabla c_t^- + \beta \nabla c_s^-}{\nabla c_r^+ + \alpha \nabla c_t^+ + \beta \nabla c_s^+} \tag{22}$$

Some implementation details worth mentioning are that does not consider the cases when $t = 1$ or $t = T$ which require the 0^{th} and $T + 1^{th}$ column of the matrix H. A rigorous derivation yields that Equation III-C holds if the 0^{th} column is assumed to be equal to the first and the $T + 1^{th}$ column is set equal to T^{th}.

D. Experiments

Basic implementation[2] can provide an illustration of the effects of applying NMF optimising both KL divergence and Frobenius norm to the audio. Three sets of experiments were performed to validate the results presented in [2]. For each of the experiments, the SNR value obtained was averaged over all the sources.

1) Varying r: The first experiment involved varying the value of r i.e the number of components obtained after factorization and determining the effect of the same on the SNR. The number r was varied from the minimum number of sources present to an arbitrarily chosen maximum of 40. Each run of the experiment produced a comparison of the three different algorithms discussed thus far. This experiment was run several times by varying the values of α and β. The overlap percentage was kept fixed at 50%, number of iterations at 1000, frame size and FFT size of 40ms and the window used was Hann.

2) Varying alhpa and beta: This experiment studies the variation of the SNR of the reconstructed magnitude spectrogram with the variation in the hyperparameters α and β. The effect of one parameter is studied while the other is set to zero. Consequently, only the loss function in [2] is used. This experiment is run several times by varying r. Further, the overlap percentage was kept fixed at 50%, number of iterations at 1000, frame size and FFT size of 40ms and the window used was Hann.

3) Varying the window used: This experiment studied the effect of window used on the SNR of the algorithm. Windows tested include the Hann window, Hamming window and Blackman-Harris window. The experiment was run several times varying the number of components r. The value of α was fixed at 10 and that of β at 0.1 as these were found to be the best combination from Section III-D.2. Further, each experiment was run over several frame sizes varying from 39ms to 99ms in steps of 3 except for a run at 40ms which is reported in the original work as the used frame size. The number of components was varied from the number of constituent sources to an arbitrarily chosen maximum of 40. Further, this experiment was run for four different audios and the results recorded and averaged to obtain the variation of SNR with frame size averaged over all sources from all the audios for different windows. Following this, the Chebshev window [5] was also used to determine the impact of side lobe attenuation and main lobe width of the windows on SNR.

E. Results

It is worth noting that the algorithm proposed in [2] is not guaranteed to converge. For certain settings of α and β, it can diverge. This can be seen in Figure 6. Results of

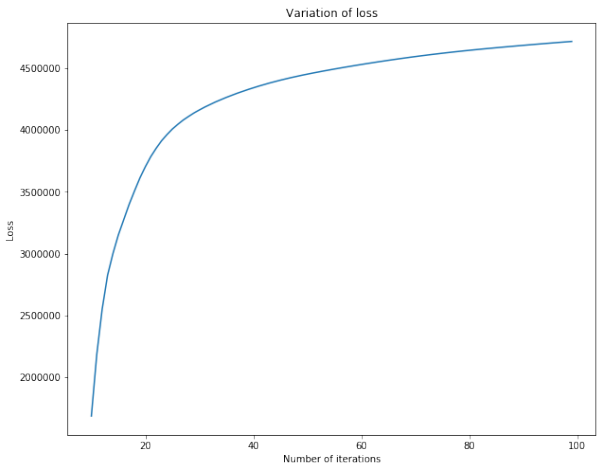


Fig. 6: Loss function increases during initial iterations for r = 300, $\alpha = 1$, $\beta = 0$

the experiment described in Section III-D.1 are presented in Figure 7. The objective of the experiment[3] was to compare the relative performance of the algorithm from the original work with vanilla NMF. In most cases, the performance of the algorithm was comparable if not superior to that of plain NMf. However, with $\alpha = 10$ and $\beta = 0.1$ the proposed algorithm clearly outperforms plain NMf. The curve displays highly erratic as only one audio is used to perform the experiment.

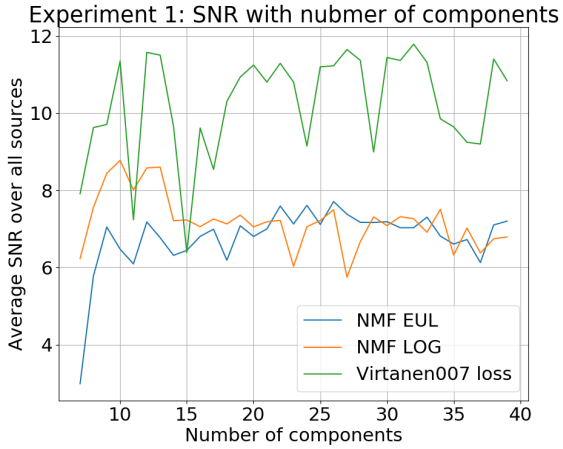
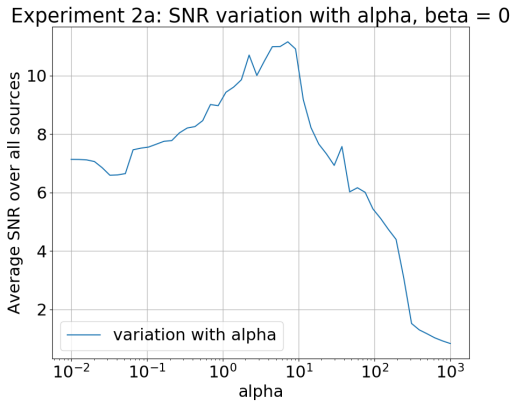
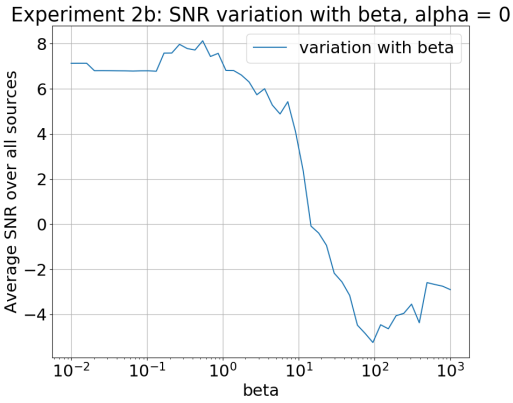


Fig. 7: Experiment 1: Variation of SNR averaged over all sources with the number of components for three different loss functions

Results from the experiment described in Section III-D.2 are presented in Figure 8. These results are very similar to those presented in [2]. However, the use of a single component does add some noise to the graphs. Further, it can be seen from Figure 8 that the highlest values of SNR occur at $\alpha = 10$ and $\beta = 0.1$. This can be seen even for other values of r[4] and hence is used as the default setting of α and β for all further experiments.



(a)



(b)

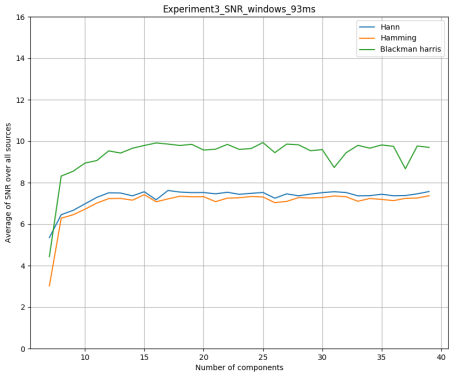
Fig. 8: Variaton of SNR with number of components when (a) $\beta = 0$ (b) $\alpha = 0$

Varying the windows as described in Section III-D.3 at a frame size of 40 ms produced results shown in Figure 9. It is quite evident that the Hamming and Hann windows outperform Blackaman-Harris at low frame sizes and are beaten by Blackman-Harris at higher frame sizes.

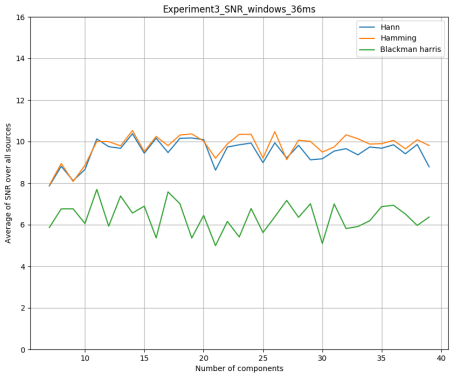
This is even more evident in the case of Figure 10 which shows that one obtains the highest SNR when frame size is set to 40ms as suggestd by the author of the original paper. All the three windows performs nearly the same at 40ms. In fact, the Blackman-Harris window outperforms the others only at sizes higher than this frame size. Further, it can be seen from Figure 11 that the Chebyshev window does not perform as well as the other three windows at 40ms. This experiment was run by setting the attenuation to the same as Hamming window i.e 43dB and letting the Chebyshev

[3] https://tinyurl.com/y35kxqc7

[4] https://tinyurl.com/yxl8fh39



(a)



(b)

Fig. 9: Variaton of SNR with number of components when (a) frame size = 36ms (b) frame size = 93ms

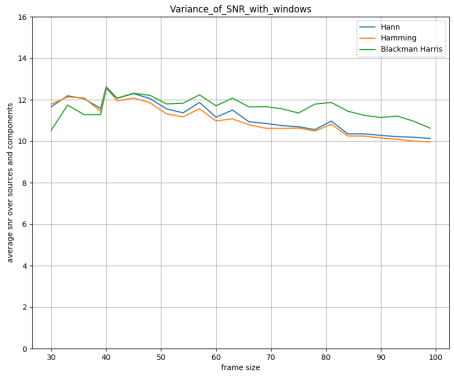


Fig. 10: Variation of SNR with different frame sizes averaged over all sources from four audios

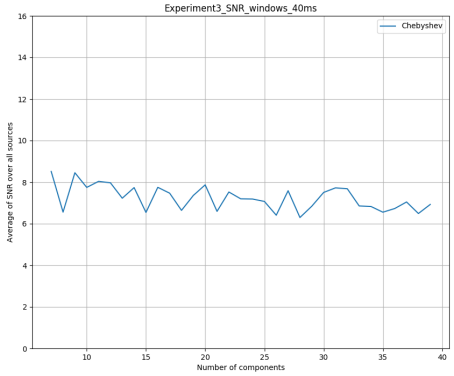


Fig. 11: Variation of SNR with number of components for a Chebyshev window at 40ms

window optimise for the smallest main lobe width. The average SNR does not cross 9dB which is lesser than that of the other three windows at 40ms.

F. Discussion and future work

Most of the results obtain closely follow the trends presented in the original paper. However, it can be seen that most of the graphs display a high degree of variance. Further, they tend to smoothen out when more audios and sources are used as is seen in Figure 10. Therefore, it is evident that more sources and audios need to be used to produce results identical to those produced in the original work. At the time of this work, the audio samples used in [2] were no longer available. However, equivalent datasets that supply musical notes can be used to produce more mixtures and data. Further, in experiment 3, only the effect of decreasing mainlobe width was tested and it was seen that it decreased performance. However, an alternative is to retain the same mainlobe width as Hamming and then increase the attenuation for the same framesize. This would give one a clearer idea of which of main lobe width and side lobe attenuation has greater impact on the SNR. While the author of the original paper observes that the sparseness criteria did not have any major impact on the factorization, this lack of impact was not observed in Experiment 2.It is also worth noting that all components were allocated to a source in this approach in contrast to the allocation of a single component with the highest SNR to a source as was done in the original paper.

While most of this work deals with the audio in the frequency domain, quality of separation in the time domain can be determined by performing listening tets. It can be seen that the third algorithm produces better results than plain NMF. In this work, the original phase of the audio has been used to reconstruct sources. However, the dependence between the phase and magnitude of a causal signal can perhaps be used to reconstruct the phase of the components better. Further better clustering algorithms that allocate components to a source without using the original sources need to be developed.

IV. Conclusion

This work provides a brief introduction to NMf, its applications, its use and related algorithms. In particular, the multiplicative update rules have been presented and their validity

emprically verified. This is followed by the verification of results present in previous work regarding source separation. In a careful survey of some implementation details, different window functions are applied to the task and it is found that all the tried windows produce nearly the same performance for the given settings of hyperparameters. Finally, a few remarks are made regarding work to be done in this direction and the need for better phase reconstrucion and clustering algorithms is highlighted.

ACKNOWLEDGMENT

I am extremely grateful to Prof. C. S. Ramalingam for his invaluable support and guidance. I am also grateful to Prof. Harishankar Ramachandran and Prof. Tuomas Virtanen for their insights and illuminating discussions.

REFERENCES

[1] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, pp. 556–562, 2001.

[2] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE transactions on audio, speech, and language processing*, vol. 15, no. 3, pp. 1066–1074, 2007.

[3] N. Gillis, "The why and how of nonnegative matrix factorization," *Regularization, Optimization, Kernels, and Support Vector Machines*, vol. 12, no. 257, 2014.

[4] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, p. 788, 1999.

[5] "Chebyshev window." https://ccrma.stanford.edu/ jos/sasp/Dolph_Chebyshev_Window_Definition.html. Accessed: 2019-05-17.