Jacobi Algorithm For Nonnegative Matrix Factorization With Transform Learning

Herwig Wendt, Dylan Fagot and Cédric Févotte
IRIT, Université de Toulouse, CNRS
Toulouse, France
firstname.lastname@irit.fr

***Abstract*—Nonnegative matrix factorization (NMF) is the state-of-the-art approach to unsupervised audio source separation. It relies on the factorization of a given short-time frequency transform into a dictionary of spectral patterns and an activation matrix. Recently, we introduced transform learning for NMF (TL-NMF), in which the short-time transform is learnt together with the nonnegative factors. We imposed the transform to be orthogonal likewise the usual Fourier or Cosine transform. TL-NMF yields an original non-convex optimization problem over the manifold of orthogonal matrices, for which we proposed a projected gradient descent algorithm in our previous work. In this contribution we describe a new Jacobi approach in which the orthogonal matrix is represented as a randomly chosen product of elementary Givens matrices. The new approach performs favorably as compared to the gradient approach, in particular in terms of robustness with respect to initialization, as illustrated with synthetic and audio decomposition experiments.**

***Index Terms*—Nonnegative matrix factorization (NMF), transform learning, single-channel source separation.**

I. Introduction

Nonnegative matrix factorization (NMF) consists of decomposing nonnegative data $\mathbf{V} \in \mathbb{R}_+^{M\times N}$ into

$$\mathbf{V} \approx \mathbf{W}\mathbf{H}, \tag{1}$$

where $\mathbf{W} \in \mathbb{R}_+^{M\times K}$ and $\mathbf{H} \in \mathbb{R}_+^{K\times N}$ are two nonnegative factors referred to as *dictionary* and *activation matrix*, respectively. The order K of the factorization is usually chosen such that $K < \min(M, N)$ such that NMF provides a low-rank approximation. In audio signal processing, $\mathbf{V}$ is a spectrogram $|\mathbf{X}|^{\circ 2}$ or $|\mathbf{X}|$, where $\mathbf{X}$ is the short-time frequency transform of a sound signal y and where $|\cdot|$ and $\circ$ denote entry-wise absolute value and exponentiation, respectively. The transform $\mathbf{X}$ is typically computed as $\mathbf{\Phi}\mathbf{Y}$, where $\mathbf{Y} \in \mathbb{R}^{M\times N}$ contains windowed segments of the original signal y in its columns, M is the length of the analysis window and N is the resulting number of segments (aka frames). $\mathbf{\Phi}$ is a transform matrix of size $M \times M$ such as the complex-valued Fourier transform or the real-valued discrete cosine transform (DCT). The factorization of $\mathbf{V}$ then leads to a dictionary $\mathbf{W}$ which contains spectral patterns and an activation matrix $\mathbf{H}$ which encodes how these patterns are mixed in every frame. Finally, the factorization (1) can be used to reconstruct latent components of the original signal via Wiener filtering [1].

In this state-of-the-art procedure, the time-frequency transform used to compute the input matrix $\mathbf{V}$ is chosen off-the-shelf and may have a considerable impact on the quality of the decomposition. As such, we proposed in [2], to *learn* the transform $\mathbf{\Phi}$ together with the factors $\mathbf{W}$ and $\mathbf{H}$. Such transform-learning procedures have received increasing attention in signal processing, for example in the context of audio decomposition with neural architectures [3] or sparse coding of images [4], but their use with NMF is to the best of our knowledge new. In [2], we defined transform-learning NMF (TL-NMF) as the solution of

$$\min_{\mathbf{\Phi},\mathbf{W},\mathbf{H}} C(\mathbf{\Phi},\mathbf{W},\mathbf{H}) \overset{\text{def}}{=} D_{\text{IS}}(|\mathbf{\Phi}\mathbf{Y}|^{\circ 2}|\mathbf{W}\mathbf{H}) + \lambda||\mathbf{H}||_1 \tag{2}$$

$$\text{s.t. } \mathbf{H} \geq 0, \mathbf{W} \geq 0,\ \forall k, ||\mathbf{w}_k||_1 = 1, \mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{I}_M \tag{3}$$

where $D_{\text{IS}}(\cdot|\cdot)$ is the Itakura-Saito divergence and where the notation $\mathbf{A} \geq 0$ expresses nonnegativity of the entries of $\mathbf{A}$. The Itakura-Saito divergence is defined by $D_{\text{IS}}(\mathbf{A}|\mathbf{B}) = \sum_{ij}(a_{ij}/b_{ij} - \log(a_{ij}/b_{ij}) - 1)$ and is a common choice in spectral audio unmixing [5] (note that other divergences could be considered without loss of generality). The orthogonality constraint imposed on $\mathbf{\Phi}$ mimics the usual Fourier or DCT transform. We here consider a real-valued transform $\mathbf{\Phi} \in \mathbb{R}^{M\times M}$ for simplicity (like the DCT). The ℓ_1 penalty term in (2) enforces some degree of sparsity on $\mathbf{H}$ which indirectly adds structure to the rows of $\mathbf{\Phi}$ as demonstrated in [2].

In [2] we described a block-descent algorithm that updates the variables $\mathbf{\Phi}$, $\mathbf{W}$ and $\mathbf{H}$ in turn (as in Algorithm 1). Because the objective function $C(\mathbf{\Phi},\mathbf{W},\mathbf{H})$ is non-convex, the proposed algorithm is only guaranteed to return a stationary point which may not be a global solution. The update of $\mathbf{W}$ and $\mathbf{H}$ given $\mathbf{\Phi}$ are tantamount to standard NMF and are obtained using majorization-minimization [6], [7] (leading to so-called multiplicative updates). In [2], we proposed a projected gradient-descent update for $\mathbf{\Phi}$ using the method described in [8]. We propose in this contribution a new method for updating $\mathbf{\Phi}$, namely a Jacobi-like iterative approach that searches $\mathbf{\Phi}$ as a product of randomly chosen Givens rotations. The approach is described in Section II and illustrated with experiments in Section III. Section IV concludes.

II. Jacobi Algorithm for transform learning

A. Givens representation of orthogonal matrices

Let us denote by $\mathbb{O}^M$ the set of real-valued orthogonal matrices of size $M \times M$. Jacobi methods have a long history in numerical eigenvalue problems such as Schur decomposition [9] or joint-diagonalization [10]. They rely on the principle

ISBN 978-90-827970-1-5

Algorithm 1: TL-NMF-Jacobi

Input : $\mathbf{Y}$, τ, K , λ
Output: $\mathbf{\Phi}$, $\mathbf{W}$, $\mathbf{H}$

Initialize $\mathbf{\Phi} = \mathbf{\Phi}^{(0)}$, $\mathbf{W} = \mathbf{W}^{(0)}$, $\mathbf{H} = \mathbf{H}^{(0)}$ and set $l = 1$
while $\epsilon > \tau$ **do**
  $\mathbf{H}^{(l)} \leftarrow$ Update $\mathbf{H}$ as in [2] (U1)
  $\mathbf{W}^{(l)} \leftarrow$ Update $\mathbf{W}$ as in [2] (U2)
  $\mathbf{\Phi}^{(l)} \leftarrow$ Update $\mathbf{\Phi}$ using Algorithm 2 (U3)
  Normalize $\mathbf{\Phi}$ to remove sign ambiguity
  $\epsilon = \frac{C(\mathbf{\Phi}^{(l-1)},\mathbf{W}^{(l-1)},\mathbf{H}^{(l-1)}) - C(\mathbf{\Phi}^{(l)},\mathbf{W}^{(l)},\mathbf{H}^{(l)})}{|C(\mathbf{\Phi}^{(l)},\mathbf{W}^{(l)},\mathbf{H}^{(l)})|}$
  $l \leftarrow l + 1$
end

that any orthogonal matrix in $\mathbb{O}^M$ may be represented as a product of Givens matrices $\mathbf{R}_{pq}(\theta) \in \mathbb{O}^M$ defined by

$$\mathbf{R}_{pq}(\theta) = \begin{array}{c} \\ p \\ \\ q \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccc} p & & & q & \end{array} \\ \begin{pmatrix} \mathbf{I} & 0 & \dots & \dots & 0 \\ 0 & \cos\theta & 0 & -\sin\theta & \vdots \\ \vdots & 0 & \mathbf{I} & 0 & \vdots \\ \vdots & \sin\theta & 0 & \cos\theta & 0 \\ 0 & \dots & \dots & 0 & \mathbf{I} \end{pmatrix} \end{array}. \quad (4)$$

The couple $(p, q) \in \{1, \dots, M\} \times \{1, \dots, M\}$ defines an axis of rotation while $\theta \in [0, 2\pi[$ represents the angle of the rotation. Given a current estimate $\mathbf{\Phi}^{(i)}$ and a couple (p, q), the Jacobi update is given by

$$\mathbf{\Phi}^{(i+1)} = \mathbf{R}_{pq}(\hat{\theta})\mathbf{\Phi}^{(i)} \quad (5)$$

$$\hat{\theta} = \arg\min_{\theta} J_{pq}(\theta) \stackrel{\text{def}}{=} D(|\mathbf{R}_{pq}(\theta)\mathbf{X}^{(i)}|^{\circ 2}|\mathbf{WH}), \quad (6)$$

and $\mathbf{X}^{(i)} = \mathbf{\Phi}^{(i)}\mathbf{Y}$ is the current transformed data. In this basic scenario, every iteration i involves the choice of a rotation axis (p, q). Every orthogonal matrix can be decomposed as a sequence product of Givens rotations, but the correct ordered sequence of rotation axes is unfortunately unknown. As such, the sequence pattern in which the axes are selected in Jacobi methods can have a dramatic impact on convergence and this has been the subject of many works for eigenvalue problems (see [9] and reference therein). The optimization of $J_{pq}(\theta)$ given (p, q) on the one hand and the sequential choice of axes (p, q) on the other hand are discussed in the next sections.

B. Optimization of $J_{pq}(\theta)$ for one axis (p, q)

By construction of Givens rotations, $\mathbf{R}_{pq}(\theta)\mathbf{X}^{(i)}$ is everywhere equal to $\mathbf{X}^{(i)}$ except for rows p and q. It follows that

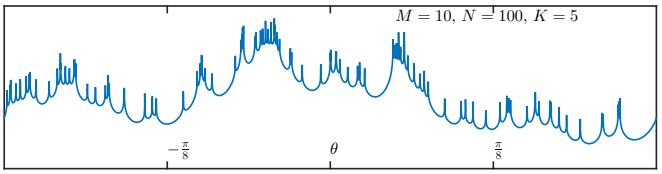

Fig. 1. Illustration of $J_{pq}(\theta)$ evaluated for 10^6 equally spaced points $\theta \in (-\frac{\pi}{4}, \frac{\pi}{4}]$ for randomly selected $\mathbf{X}$ and $\hat{\mathbf{V}}$.

$J_{pq}(\theta)$ can be expressed as

$$J_{pq}(\theta) = \sum_n \left[\frac{(\cos\theta x_{pn}^{(i)} - \sin\theta x_{qn}^{(i)})^2}{\hat{v}_{pn}} + \frac{(\sin\theta x_{pn}^{(i)} + \cos\theta x_{qn}^{(i)})^2}{\hat{v}_{qn}} - 2\log\left[(\cos\theta x_{pn}^{(i)} - \sin\theta x_{qn}^{(i)})(\sin\theta x_{pn}^{(i)} + \cos\theta x_{qn}^{(i)})\right] \right] + cst \quad (7)$$

where $\hat{v}_{mn} \stackrel{\text{def}}{=} [\mathbf{WH}]_{mn}$ and cst is a constant w.r.t. θ. $J_{pq}(\theta)$ is $\frac{\pi}{2}$ periodic and we may address its minimization over the domain $\theta \in (-\frac{\pi}{4}, \frac{\pi}{4}]$ only. Unfortunately $J_{pq}(\theta)$ does not appear to have a closed-form minimizer. Furthermore, $J_{pq}(\theta)$ is highly non-convex w.r.t to θ. Moreover, one can show that (7) is not everywhere smooth because it has N poles for $\theta \in (-\frac{\pi}{4}, \frac{\pi}{4}]$, see Fig. 1 for an illustration. For these reasons combined, gradient-descent based minimization proves, at best, highly inefficient.

To circumvent these difficulties, we propose to resort to an original randomized grid search procedure described next. The strategy consists in drawing at random N_{prop} proposals $\tilde{\theta} \in (-\frac{\alpha\pi}{4}, \frac{\alpha\pi}{4}]$, $\alpha \in (0, 1]$, which are used to approximate (6) as $\hat{\theta} \approx \arg\min_{\theta \in \{\tilde{\theta}_i\}_{i=1}^{N_{\text{prop}}}} J_{pq}(\theta)$. If the update of $\mathbf{\Phi}$ in (5) does not improve the objective function, the procedure could be repeated until a value for $\hat{\theta}$ is found for which the update yields an improvement. Here, we will instead move on to a different axis (p, q), see Section (II-C). If the update with $\hat{\theta}$ yields an improvement, then the approximation can be refined by repeating the procedure for smaller and smaller values of α. We will intertwine such refinements with sweeps over random couples of rows (p, q), as described next.

C. Updating $\mathbf{\Phi}$: selection of the rotation axes (p, q)

The Givens matrices (4) act on a total number of $M(M-1)/2$ different rotation axes, defined by couples (p, q). To perform the transform update step (U3) in Algorithm 1, we propose to compute Jacobi updates (5-6), as described in the previous paragraph, for a total of R couples (p, q) that are selected at random from the $M(M-1)/2$ possible ones. For computational efficiency, we make use of the fact that the rotation for a given (p, q) affects the rows p and q only. Therefore, $\frac{M}{2}$ mutually independent random couples (p, q), a so-called rotation set [9], can be updated in parallel. It is easy to see that a rotation set can be generated by drawing, without replacement, the values for p and q at random from the set $(1, \dots, M)$. This update for $\mathbf{\Phi}$ is summarized in Algorithm 2.

ISBN 978-90-827970-1-5 © EURASIP 2018

Algorithm 2: Jacobi update of $\mathbf{\Phi}$ at iteration l

Input : $\mathbf{\Phi}, \mathbf{X} = \mathbf{\Phi Y}, \hat{\mathbf{V}} = \mathbf{WH}, N_{\text{prop}}, R, l$
Output: $\mathbf{\Phi}$

for $k = 1, \ldots, \lfloor 2R/M \rfloor$ **do**
  Generate a random permutation of $(1, ..., M)$ in $\mathbf{u}$
  for $j = 1, \ldots, M/2$ **do**
    $(p, q) = (\mathbf{u}_j, \mathbf{u}_{j+\frac{M}{2}})$
    for $s = 1, \ldots, N_{\text{prop}}$ **do**
      Draw at random $\tilde{\theta} \in (-\frac{\alpha(k,l)\pi}{4}, \frac{\alpha(k,l)\pi}{4}]$
      Evaluate $J_{pq}(\tilde{\theta}) = D(|\mathbf{R}_{pq}(\tilde{\theta})\mathbf{X}|^{\circ 2} \big| \hat{\mathbf{V}})$
    end
    $\hat{\theta} \leftarrow \arg\min_{\tilde{\theta}} J_{pq}(\tilde{\theta})$
    if $D(|\mathbf{R}_{pq}(\hat{\theta})\mathbf{X}|^{\circ 2} \big| \hat{\mathbf{V}}) < D(|\mathbf{X}|^{\circ 2} \big| \hat{\mathbf{V}})$ **then**
      Update transform $\mathbf{\Phi} \leftarrow \mathbf{R}_{pq}(\hat{\theta})\mathbf{\Phi}$
    end
  end
end

Further, the refinement factor α in the optimization of $J_{pq}(\theta)$, which plays a role similar to a step size, is sequentially updated as

$$\alpha = \alpha(k, l) = l^{-a_1} k^{-a_2}, \tag{8}$$

where l is the number of the outer iteration for updating $\mathbf{W}, \mathbf{H}$ and $\mathbf{\Phi}$ (see Algorithm 1) and k is the inner iteration over the number of blocks of $\frac{M}{2}$ mutually independent random couples (p, q) in the update of $\mathbf{\Phi}$. The Jacobi algorithm for updating $\mathbf{\Phi}$ is summarized in Algorithm 2.

III. Experiments

A. Transform learning experiment with synthetic data

We begin with studying the performance of the proposed randomized Jacobi algorithm for finding a transform $\mathbf{\Phi}$ given $\hat{\mathbf{V}}$ (i.e., for update step (U3) in Algorithm 1). To this end, we let $\mathbf{Y} \in \mathbb{R}^{M \times N}$ and $\mathbf{\Phi}^* \in \mathbb{O}^M$ be two randomly generated matrices, and let $\mathbf{V}^* = |\mathbf{\Phi}^* \mathbf{Y}|^{\circ 2}$, and we study the minimization of

$$F(\mathbf{\Phi}) = D(|\mathbf{\Phi Y}|^{\circ 2} \big| \mathbf{V}^*) \quad \text{s.t.} \quad \mathbf{\Phi} \in \mathbb{O}^M, \tag{9}$$

i.e., Algorithm 1 with $\hat{\mathbf{V}} = \mathbf{V}^*$ and update steps (U1) and (U2) removed. The minimum of $F(\mathbf{\Phi})$ is here 0 by construction. We compare the Jacobi approach for minimizing (9) proposed in this paper with the gradient-descent approach described in [2], which consists of Algorithm 1 with step (U3) replaced by a gradient-descent step with Armijo step size selection. The algorithms are run with $\tau = 10^{-10}$, and $(a_1, a_2) = (0.75, 0)$ for this experiment. We initialize the algorithms with $\mathbf{\Phi} = \mathbf{\Phi}^{(0)}$ in the vicinity of the ground-truth solution $\mathbf{\Phi}^*$. To measure the closeness of the initialization to the ground-truth solution we define the Initialization Proximity Ratio (IPR) as

$$\text{IPR} = 10 \log_{10} \frac{||\mathbf{\Phi}^*||_F^2}{||\mathbf{\Phi}^{(0)} - \mathbf{\Phi}^*||_F^2}. \tag{10}$$

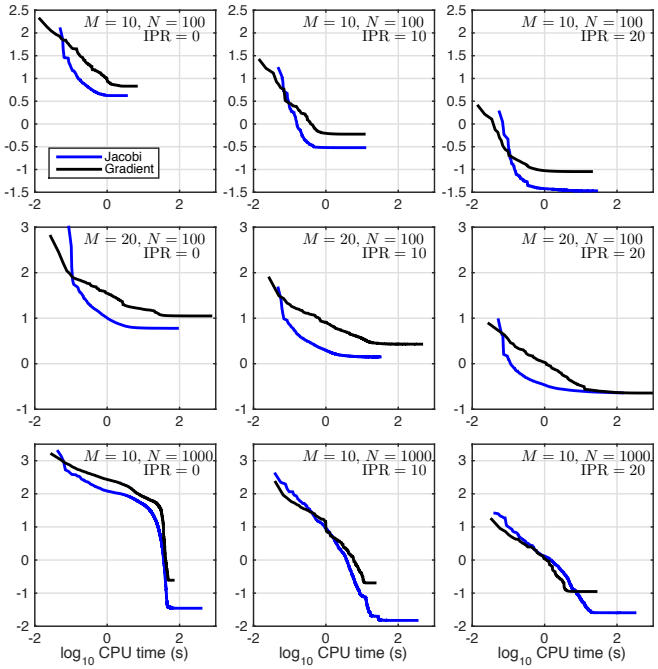

Fig. 2. Finding $\mathbf{\Phi}$ for fixed $\hat{\mathbf{V}}$. Objective function values $\log_{10}(F(\mathbf{\Phi}))$ as a function of CPU time for different values M, N (from top to bottom) and initializations for $\mathbf{\Phi}$ (increasingly close to the ground-truth $\mathbf{\Phi}^*$, from left to right): Jacobi (blue) and Gradient (black) algorithm, respectively.

A large IPR value means $\mathbf{\Phi}^{(0)}$ and $\mathbf{\Phi}^*$ are close; $\mathbf{\Phi}^{(0)}$ is generated as $\mathbf{\Phi}^{(0)} = \text{proj}_{\mathbb{O}^M}(\mathbf{\Phi}^* + \sigma \mathbf{B})$ where $\text{proj}_{\mathbb{O}^M}$ denotes the projection onto $\mathbb{O}^M$ [8], $\mathbf{B}$ is a standard normal random matrix and $\sigma > 0$ is set to meet the desired IPR value.

The objective function values $F(\mathbf{\Phi})$ obtained for the two algorithms are plotted in Fig. 2 for several values of M, N and IPR, as a function of CPU time. As expected, larger IPR values lead overall to solutions with smaller objective function value. A comparison of the two algorithms yields the following conclusions. First, with the proposed randomized Jacobi algorithm, a transform $\mathbf{\Phi}$ is obtained that corresponds with a local minimum of (9) that has a smaller objective function value than that returned by gradient descent. Second, the proposed algorithm is in general faster in finding a transform $\mathbf{\Phi}$ with objective function below a given value, as indicated by the fact that in most of the cases plotted in Fig. 2, the blue curve (Jacobi) is consistently below the black curve (Gradient). Overall, these findings clearly demonstrate the practical benefits of the proposed randomized Jacobi algorithm for updating $\mathbf{\Phi}$.

B. NMF with transform learning for audio data

We now study the full TL-NMF problem (2-3) with unknowns $(\mathbf{\Phi}, \mathbf{W}, \mathbf{H})$ for the decomposition of the toy piano sequence used in [5]. The audio sequence consists of four notes played all at once in the first measure and then by pairs in all possible combinations in the subsequent measures, see Fig. 3 (a) and (b). The audio signal was recorded live on a

ISBN 978-90-827970-1-5

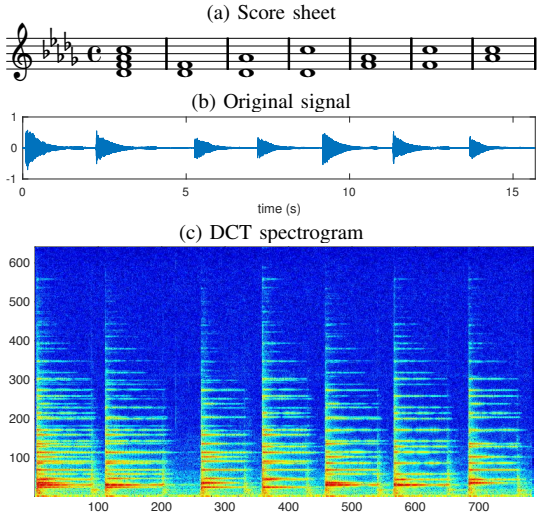

Fig. 3. Three representations of the piano data.

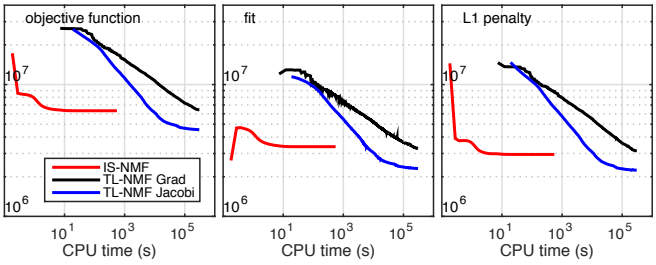

Fig. 4. Objective function $C(\mathbf{\Phi}, \mathbf{W}, \mathbf{H})$, data fitting term $D(|\mathbf{\Phi Y}|^{\circ 2}|\mathbf{WH})$ and sparsity term $\lambda\|\mathbf{H}\|_1$ with respect to CPU time.

real piano. The duration is 15.7 s with sampling frequency $f_s = 16$ kHz. The columns of matrix $\mathbf{Y}$ are composed of adjacent frames of size $M = 640$ (40 ms) with 50 % overlap, windowed with a sine-bell. This leads to a total of $N = 785$ frames.

We compare the performance and results of TL-NMF-Jacobi, TL-NMF-Gradient (the gradient descent method of [2]) and baseline (sparse) IS-NMF. The latter simply consists of running Algorithm 1 without step (U3) and with a fixed DCT transform $\mathbf{\Phi} = \mathbf{\Phi}_{\text{DCT}}$ defined as

$$[\mathbf{\Phi}_{\text{DCT}}]_{qm} = (2M)^{-\frac{1}{2}} \cos\left(\pi(q+1/2)(m+1/2)/M\right). \quad (11)$$

The DCT spectrogram of the audio data is plotted in Fig. 3 (c). The three algorithms are run with $K = 8$. IS-NMF was run with $\tau = 10^{-10}$, which ran in a few minutes on a personal computer. TL-NMF-Gradient and TL-NMF-Jacobi were run for several days, reaching $\tau \approx 4 \cdot 10^{-7}$. Further, the parameters for TL-NMF-Jacobi are set to $N_{\text{prop}} = 100$, $R = 6$ and $(a_1, a_2) = (0.3, 0.7)$. All algorithms are initialized with the same random factors $\mathbf{W}$ and $\mathbf{H}$, and TL-NMF-Jacobi and TL-NMF-Gradient are initialized with the same random orthogonal matrix $\mathbf{\Phi}$. The hyper-parameter λ was set to 10^6, like in [2].

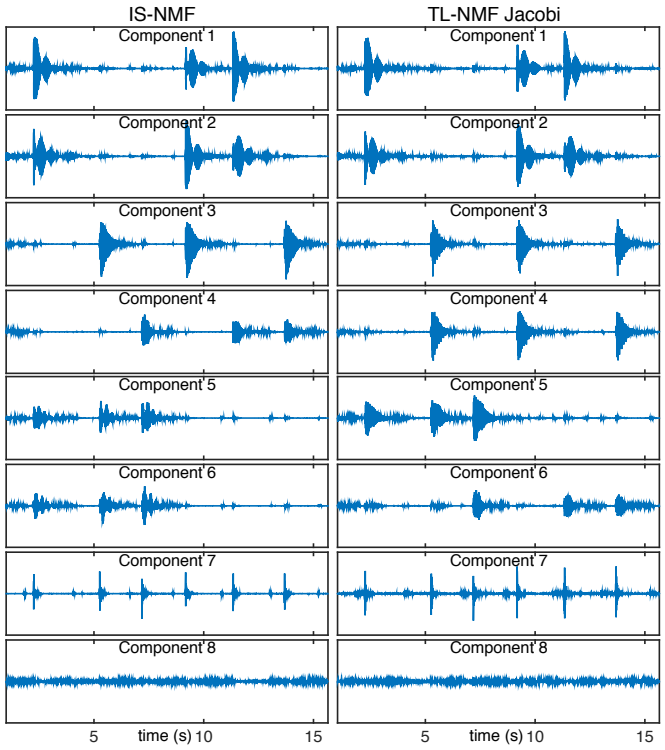

Fig. 5. Separated audio components using IS-NMF (left) and Jacobi TL-NMF (right), sorted by decreasing energy (from top to bottom). IS-NMF (resp., TL-NMF) splits the first (resp., second) and third notes over two components. In the two cases, component 7 extracts percussive sounds and component 8 contains residual noise.

Minimization of the objective function. Fig. 4 plots the values of the objective function $C(\mathbf{\Phi}, \mathbf{W}, \mathbf{H})$, data fitting term $D(|\mathbf{\Phi X}|^{\circ 2})$ and sparsity term $\lambda\|\mathbf{H}\|_1$ as a function of CPU time and yields the following conclusions. Firstly, IS-NMF is of course considerably faster because $\mathbf{\Phi}$ is fixed in its case. Then, TL-NMF-Gradient is much slower than TL-NMF-Jacobi; it has not been able to reach a stationary point within the experiment's runtime, and not even an objective function value below that of IS-NMF, despite the extra flexibility offered by learning $\mathbf{\Phi}$. In contrast, the proposed TL-NMF-Jacobi algorithm yields objective function values that are significantly below that reached by IS-NMF, already after a small fraction of the total runtime of the experiment. Finally, Fig. 4 shows that both the fit and the penalty term on $\mathbf{H}$ are decreased along the iterations, which confirms the mutual influence of the sparsity of $\mathbf{H}$ and the learnt transform and dictionary.

Decomposition. Fig. 5 displays the 8 latent components that can be reconstructed from the factorization returned by IS-NMF and by TL-NMF-Jacobi. The components have been reconstructed with standard Wiener filtering, inverse transformation and overlap-add [2], [5]. The set of components returned by the two methods are comparable and coherent: the piano notes, percussive sounds (hammer hitting the strings, sustain pedal release) and residual sounds are separated into distinct components. The corresponding audio

ISBN 978-90-827970-1-5 © EURASIP 2018

files are available online.[1] The audio quality of the components is satisfactory and comparable between the two methods, the components obtained with TL-NMF being a tiny bit fuzzy.

Learnt transform. We finally examine examples of the atoms returned by TL-NMF (rows $\{\boldsymbol{\phi}_m\}_m$ of $\boldsymbol{\Phi}$). Fig. 6 displays the 33 atoms which most contribute to the audio signal (i.e., with largest values of $\|\boldsymbol{\Phi}\mathbf{Y}\|_2$). As already observed in [2], the atoms adjust to the shape of the analysis window and come in *quadrature pairs*. As such, TL-NMF is able to learn a shift-invariant representation. Additionally, Fig. 6 shows that TL-NMF learns a variety of oscillatory patterns: regular tapered sine/cosine-like atoms (e.g., $\boldsymbol{\phi}_1$-$\boldsymbol{\phi}_8$), harmonic atoms exhibiting slow ($\boldsymbol{\phi}_9$, $\boldsymbol{\phi}_{12}$-$\boldsymbol{\phi}_{14}$) to fast ($\boldsymbol{\phi}_{27}$, $\boldsymbol{\phi}_{32}$) amplitude modulations, atoms with little regularity (e.g., $\boldsymbol{\phi}_{18}$, $\boldsymbol{\phi}_{22}$, $\boldsymbol{\phi}_{24}$) and atoms with uncommon structure ($\boldsymbol{\phi}_{28}$). Atoms $\boldsymbol{\phi}_1$ to $\boldsymbol{\phi}_8$ fit to the fundamental frequencies of the 4 individual notes while the other atoms adjust to the specificities of the data (such as partials of the notes). The previous paragraph shows that they make sense of the data in their ability to produce a meaningful decomposition that accurately describes the data. We feel that it is quite remarkable to be able to learn such a structured transform from a random initialization.

IV. Conclusion

In this paper, we proposed a novel Jacobi algorithm for TL-NMF. Experiments with synthetic and audio data have illustrated its superior performance with respect to our previous gradient-descent approach. In our setting it led to faster convergence and convergence to solutions with lesser objective values. We conjecture that the latter observation may be a consequence of the randomness element in the axes selection. The randomness is likely to improve the exploration of the non-convex landscape of the objective function. Despite TL-NMF-Jacobi being more efficient than TL-NMF-Gradient, it still runs very slow as it requires a few days to converge when applied to a few seconds audio signal (using a standard personal computer). Future work will look into faster optimization using for example incremental/stochastic variants.

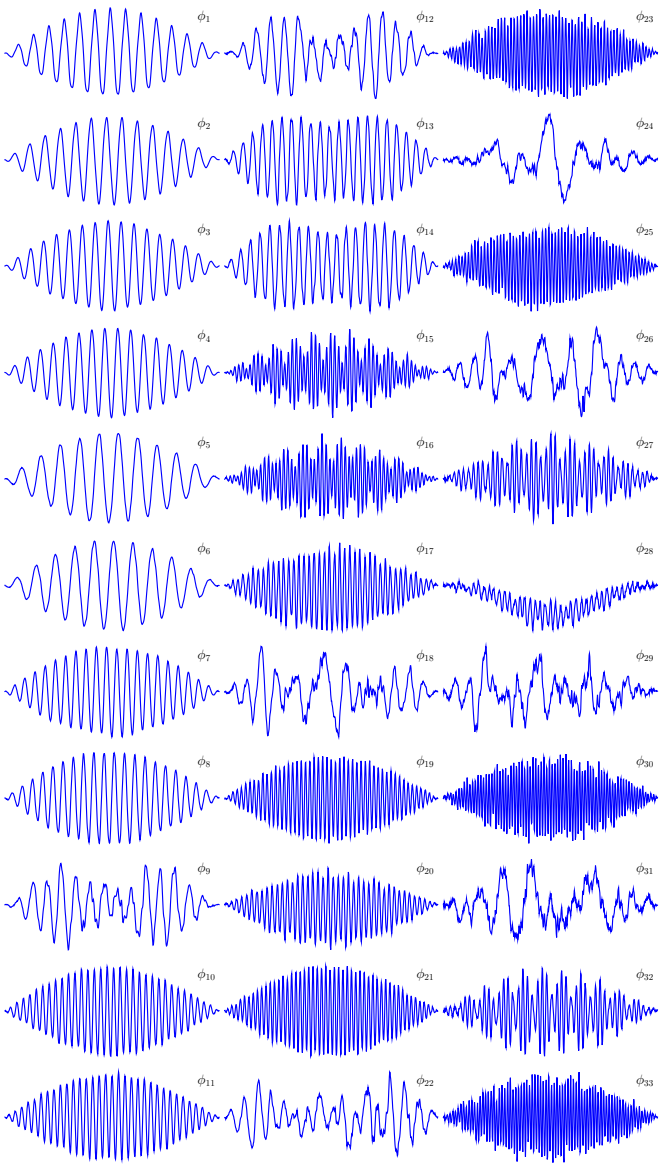

Fig. 6. The 33 most energy-significant atoms learnt with TL-NMF-Jacobi.

Acknowledgment

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 681839 (project FACTORY).

References

[1] P. Smaragdis, C. Févotte, G. Mysore, N. Mohammadiha, and M. Hoffman, "Static and dynamic source separation using nonnegative factorizations: A unified view," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 66–75, May 2014.

[2] D. Fagot, H. Wendt, and C. Févotte, "Nonnegative matrix factorization with transform learning," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[3] S. Venkataramani and P. Smaragdis, "End-to-end source separation with adaptive front-ends," arXiv:1705.02514, Tech. Rep., 2017.

[4] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms," *IEEE Transactions on Signal Processing*, vol. 61, no. 5, pp. 1072–1086, 2013.

[5] C. Févotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the Itakura-Saito divergence. With application to music analysis," *Neural Computation*, vol. 21, no. 3, pp. 793–830, Mar. 2009.

[6] C. Févotte and J. Idier, "Algorithms for nonnegative matrix factorization with the β-divergence," *Neural Computation*, vol. 23, no. 9, pp. 2421–2456, 2011.

[7] A. Lefèvre, F. Bach, and C. Févotte, "Itakura-Saito nonnegative matrix factorization with group sparsity," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011.

[8] J. H. Manton, "Optimization algorithms exploiting unitary constraints," *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 635–650, 2002.

[9] G. H. Golub and C. F. Van Loan, *Matrix computations*, 3rd ed. The Johns Hopkins University Press, 1996.

[10] J.-F. Cardoso and A. Souloumiac, "Jacobi angles for simultaneous diagonalization," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 1, pp. 161–164, 1996.

[1] https://www.irit.fr/~Cedric.Fevotte/extras/eusipco2018/audio.zip