DISTRIBUTED DYADIC CYCLIC DESCENT FOR NON-NEGATIVE MATRIX FACTORIZATION

M.O. Ulfarsson[†], V. Solo[‡], J. Sigurdsson[†], and J.R. Sveinsson[†]

[†]University of Iceland, Dept. Electrical Eng., Reykjavik, ICELAND
[‡]University of New South Wales, School of Electrical Eng., Sydney, AUSTRALIA

ABSTRACT

Non-negative matrix factorization (NMF) has found use in fields such as remote sensing and computer vision where the signals of interest are usually non-negative. Data dimensions in these applications can be huge and traditional algorithms break down due to unachievable memory demands. One is then compelled to consider distributed algorithms. In this paper, we develop for the first time a distributed version of NMF using the alternating direction method of multipliers (ADMM) algorithm and dyadic cyclic descent. The algorithm is compared to well established variants of NMF using simulated data, and is also evaluated using real remote sensing hyperspectral data.

Index Terms— Non-negative matrix factorization, dyadic cyclic descent, alternating direction method of multipliers, optimization, distributed signal processing.

1. INTRODUCTION

Non-negative matrix factorization (NMF) is a standard method for finding a low-rank representation for non-negative signals. Although algorithms for performing NMF were developed earlier, it was the work of Lee and Seung [1, 2] that triggered a rapid growth in the use of NMF in areas such as image processing [3], document mining [4], community detection, [5] and remote sensing [6].

Due to the size of the data in the above-mentioned application, distributed storage and processing is often necessary. An important class of distributed algorithms is the alternative direction method of multipliers (ADMM) [7]. The aim of ADMM is to split a large global optimization problem into a number of small local subproblems that are fused to solve the global problem. Those small subproblems can be solved on different nodes in a computer cluster.

1.1. Related Work

A number of NMF algorithms have been proposed in the literature [8, 9, 1, 3, 10]. These algorithms include gradient descent, alternating least squares, and multiplicative algorithms. A distributed version of the Lee-Seung NMF algorithm was proposed in [11] for sparse web-scale data. In [12], ADMM was used to develop a distributed version of principal component analysis (PCA) with application to sensor networks.

1.2. Contribution

We develop, for the first time a distributed NMF algorithm. Our particular version is also aided by the use of the recently developed dyadic cyclic descent algorithm [13, 14] which has been used for sparse component analysis. Our motivation comes from the hyperspectral unmixing problem where NMF has been widely used. In this problem, the data consists of a datacube that represents a sequence of images each measured at a different spectral band. The image dimension is often very large so it is of interest to split the datacube into smaller datacubes that can be processed individually and then fused together to solve the overall NMF problem. This is precisely what our proposed algorithm does. To solve those smaller subproblems we employ a dyadic cyclic descent algorithm.

The paper is organized as follows. In Section 2, the model used is introduced and in Section 3, the distributed version is detailed. In Section 4, the algorithms are evaluated and compared to two variants of NMF. Finally, in Section 5, conclusions are drawn.

1.3. Notation

Matrices are denoted with upper case bold letters and vectors denoted with lower case bold letters. $\boldsymbol{I}_M$ is the $M \times M$ identity matrix. The tth row of vector $\boldsymbol{S}$ is denoted as $\boldsymbol{s}_t^T$, and the jth column vector of $\boldsymbol{S}$ is denoted as $\boldsymbol{s}_{(j)}$. The matrix $\boldsymbol{A}_{\text{-}j}$ is equal to $\boldsymbol{A}$ with its jth column removed. An estimate of $\boldsymbol{s}$ at iteration k is denoted as $\boldsymbol{s}^k$; $\text{tr}(\boldsymbol{A}) = \sum_i a_{ii}$.

2. DYADIC CYCLIC DESCENT

There are two main types of NMF algorithms, one is based on the optimizing a Kullback-Leibler divergence, and the other on optimizing a least squares problem [1]. In this paper, the

This work was partly supported by the Research Fund of the University of Iceland and the Icelandic Research Fund (130635-051)

focus is on the least squares approach which involves solving

$$\hat{\boldsymbol{A}}, \hat{\boldsymbol{S}} = \arg\min \|\boldsymbol{Y} - \boldsymbol{S}\boldsymbol{A}^T\|_F^2 \quad \text{s.t. } \boldsymbol{S} \geq 0, \boldsymbol{A} \geq 0 \text{ and } \|\boldsymbol{a}_{(j)}\|^2 = 1,\ j = 1, ..., r \tag{1}$$

where $\boldsymbol{Y}$ is a $T \times M$ matrix, $\boldsymbol{A}$ is an $M \times r$ non-negative matrix, and $\boldsymbol{S}$ a $T \times r$ non-negative matrix. This formulation is unusual due to the unit norm constraint on the columns of $\boldsymbol{A}$ but there is no loss of generality in doing this since, as long as the $\boldsymbol{S}\boldsymbol{A}^T$ is kept fixed, $\boldsymbol{A}$ and $\boldsymbol{S}$ can always be re-normalized. The reason for the normalization is that it saves some computations in the algorithm below.

There is no closed form solution available for solving (1). We propose to use the dyadic cyclic descent (DCD) to solve this problem which is based on the dyadic expansion

$$\boldsymbol{S}\boldsymbol{A}^T = \sum_{j=1}^{r} \boldsymbol{s}_{(j)} \boldsymbol{a}_{(j)}^T. \tag{2}$$

Using (2), a cyclic descent procedure can be devised by iteratively updating the columns of $\boldsymbol{A}$ and $\boldsymbol{S}$. The algorithm has two steps that are iterated until convergence:

$\boldsymbol{a}$**-step** : Given $\boldsymbol{R}_j = \boldsymbol{Y} - \boldsymbol{S}_{\text{-}j}\boldsymbol{A}_{\text{-}j}^T$ and $\boldsymbol{s}_{(j)}^k$, get

$$\begin{aligned} \boldsymbol{a}_{(j)}^{k+1} &= \text{argmin}_{\boldsymbol{a}:\boldsymbol{a}\geq\boldsymbol{0},\|\boldsymbol{a}\|=1} \|\boldsymbol{R}_j - \boldsymbol{s}_{(j)}^k \boldsymbol{a}^T\|^2 \\ &= \max\left(\boldsymbol{0}, \frac{\boldsymbol{R}_j^T \boldsymbol{s}_{(j)}^k}{\|\boldsymbol{R}_j^T \boldsymbol{s}_{(j)}^k\|}\right). \end{aligned}$$

$\boldsymbol{s}$**-step** : Given $\boldsymbol{R}_j = \boldsymbol{Y} - \boldsymbol{S}_{\text{-}j}\boldsymbol{A}_{\text{-}j}^T$ and $\boldsymbol{a}_{(j)}^{k+1}$, get

$$\begin{aligned} \boldsymbol{s}_{(j)}^{k+1} &= \text{argmin}_{\boldsymbol{s}\geq\boldsymbol{0}} \|\boldsymbol{R}_j - \boldsymbol{s}{\boldsymbol{a}_{(j)}^{k+1}}^T\|^2 \\ &= \max(\boldsymbol{0}, \boldsymbol{R}_j \boldsymbol{a}_{(j)}^{k+1}). \end{aligned}$$

We call the resulting algorithm NMF-DCD.

3. DISTRIBUTED DYADIC CYCLIC DESCENT

The main idea in developing the distributed dyadic cyclic descent for NMF is to split it into N subproblems, i.e.,

$$\hat{\boldsymbol{A}}, \hat{\boldsymbol{S}} = \begin{array}{cc} \text{argmin}_{\boldsymbol{A}\geq\boldsymbol{0},\boldsymbol{S}\geq\boldsymbol{0}} & \frac{1}{2}\sum_{i=1}^{N} \|\boldsymbol{Y}_i - \boldsymbol{S}_i\boldsymbol{A}_i^T\|_F^2 \\ \text{s.t.} & \boldsymbol{A}_i - \boldsymbol{A} = \boldsymbol{0} \\ & \boldsymbol{A}_i \geq \boldsymbol{0}, \quad i = 1, ..N \end{array} \tag{3}$$

where $\boldsymbol{Y} = [\boldsymbol{Y}_1^T, ..., \boldsymbol{Y}_N^T]^T$ and $\boldsymbol{S} = [\boldsymbol{S}_1^T, ..., \boldsymbol{S}_N^T]^T$. To solve (3) we use the ADMM algorithm, which uses the following augmented Lagrangian function

$$\begin{aligned} L_\rho &= \sum_{i=1}^{N} \left(\frac{1}{2}\|\boldsymbol{Y}_i - \boldsymbol{S}_i\boldsymbol{A}_i^T\|_F^2 + \text{tr}\left(\boldsymbol{X}_i^T(\boldsymbol{A}_i - \boldsymbol{A})\right) \right. \\ &+ \left. \frac{\rho}{2}\|\boldsymbol{A}_i - \boldsymbol{A}\|_F^2 \right). \end{aligned}$$

The ADMM algorithm for solving (3) is given by

$$\begin{aligned} \boldsymbol{S}_i^{k+1}, \boldsymbol{A}_i^{k+1} &= \arg \min_{\boldsymbol{A}_i\geq\boldsymbol{0},\boldsymbol{S}_i\geq\boldsymbol{0}} \frac{1}{2}\|\boldsymbol{Y}_i - \boldsymbol{S}_i\boldsymbol{A}_i^T\|_F^2 \\ &+ \text{tr}(\boldsymbol{X}^k)^T(\boldsymbol{A}_i - \boldsymbol{A}^k) \\ &+ \frac{\rho^k}{2}\|\boldsymbol{A}_i - \boldsymbol{A}^k\|_F^2 \\ \boldsymbol{A}^{k+1} &= \max\left(\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{A}_i^{k+1} + \frac{1}{\rho^k}\boldsymbol{X}_i^k), 0\right) \\ \boldsymbol{X}_i^{k+1} &= \boldsymbol{X}_i^k + \rho^k(\boldsymbol{A}_i^{k+1} - \boldsymbol{A}^{k+1}) \end{aligned}$$

where the max operator operates elementwise. Note that we allow the penalty parameter ρ^k to vary with iteration number which has been shown to improve the convergence rate [7]. To estimate $\boldsymbol{A}_i$ and $\boldsymbol{S}_i$ we use DCD which in this case involves performing the following two steps iteratively

$\boldsymbol{a}$**-step** : Given $\boldsymbol{R}_{ij}$,$\boldsymbol{s}_{i,(j)}^k$, $\mathbf{x}_{i,(j)}^k$, and $\boldsymbol{a}_{(j)}^k$, get

$$\boldsymbol{a}_{i,(j)}^{k+1} = \max\left(0, \frac{\boldsymbol{R}_{ij}^T\boldsymbol{s}_{i,(j)}^k - \mathbf{x}_{i,(j)}^k + \rho^k\boldsymbol{a}_{(j)}^k}{\|\boldsymbol{R}_{ij}^T\boldsymbol{s}_{i,(j)}^k - \mathbf{x}_{i,(j)}^k + \rho^k\boldsymbol{a}_{(j)}^k\|}\right).$$

with $\boldsymbol{R}_{ij} = \boldsymbol{Y}_i - \boldsymbol{S}_{i,\text{-}j}\,\boldsymbol{A}_{i,\text{-}j}^T$.

$\boldsymbol{s}$**-step** : Given $\boldsymbol{R}_{ij}$, and $\boldsymbol{a}_{i,(j)}^{k+1}$, get

$$\boldsymbol{s}_{i,(j)}^{k+1} = \max(0, \boldsymbol{R}_{ij}\boldsymbol{a}_{i,(j)}^{k+1}).$$

We call the resulting algorithm dNMF-DCD and for easy reference it is listed in Algorithm 1.

Algorithm 1: dNMF-DCD algorithm.

Input: $\boldsymbol{Y}$, r, N,
Initialization:
Split $\boldsymbol{Y}$ into N matrices, $\boldsymbol{Y}_1, \boldsymbol{Y}_2, ..., \boldsymbol{Y}_N$.
Initialize $\boldsymbol{A}_i^0,\ \boldsymbol{S}_i^0,\ i = 1, ..., N$
Set $\boldsymbol{A}^0 = \boldsymbol{0}, \boldsymbol{X}^0 = \boldsymbol{0}$
for *k=0...* **do**
  update ρ^k
  for *i=1...N* **do**
    Estimate $\boldsymbol{A}_i^{k+1}$ and $\boldsymbol{S}_i^{k+1}$
  Calculate $\boldsymbol{A}^{k+1}$ and $\boldsymbol{X}^{k+1}$
Output: $\hat{\boldsymbol{A}} = \hat{\boldsymbol{A}}_i, \hat{\boldsymbol{S}}_i,\ i = 1, ..., N$

4. EXAMPLES

4.1. Simulated Data

In this section, we compare the NMF-DCD and the dNMF-DCD against two NMF algorithms [1]. The first is the Lee-Seung algorithm [1] which we call NMF-L&S, and the second

[1]The Matlab nnmf implementation is used for these two algorithms.

algorithm is the alternating least squares algorithm for NMF [8] which we refer to as NMF-ALS.

To evaluate these algorithms we use a simulated data set, generated according to

$$\boldsymbol{Y} = \boldsymbol{S}\boldsymbol{A}^T + \boldsymbol{N}$$

with $M = 100$, $r = 6$, and $T = 1000$. The elements of $\boldsymbol{A}$ and $\boldsymbol{S}$ are drawn from the standard uniform distribution on the open interval $(0, 1)$, and the elements of $\boldsymbol{N}$ are drawn from the Gaussian distribution with zero mean and variance σ^2. The signal to noise ratio (SNR) is set to

$$\text{SNR} = 10\log_{10}\left(\frac{\|\boldsymbol{S}\boldsymbol{A}^T\|_F^2}{TM\sigma^2}\right) = 14.01\text{dB} \qquad (4)$$

by choosing $\sigma^2 = 0.1$. In these tests, we assume that the rank (r) is known.

The algorithms are run 100 times using the simulated data. In each run, different random values are used to initialize the algorithms. The algorithms are terminated when the relative change in $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{S}}$ is less than 10^{-4}. The NMF-ALS algorithm also needs to be terminated if the differences of the residuals between each iteration is less that a very small values (10^{-12}). If this is not done, the NMF-ALS algorithm converges to a degenerate solution.

For the dNMF-DCD algorithm, the data matrix $\boldsymbol{Y}$ is split into $N = 4$ matrices, i.e., $\boldsymbol{Y} = [\boldsymbol{Y}_1^T, \boldsymbol{Y}_2^T, \boldsymbol{Y}_3^T, \boldsymbol{Y}_4^T]^T$. The dNMF-DCD algorithm is run for 60 iterations, and the ρ^k parameter is increased in each iteration according to $\rho^k = e^{0.307k} - 1$. By using this method of changing ρ^k, the algorithm is able to adapt to each region in the image and in the final iterations, all $\hat{\boldsymbol{A}}_i$ matrices are forced to converge into the same matrix. The values of ρ^k in each iteration are shown in Fig. 1.

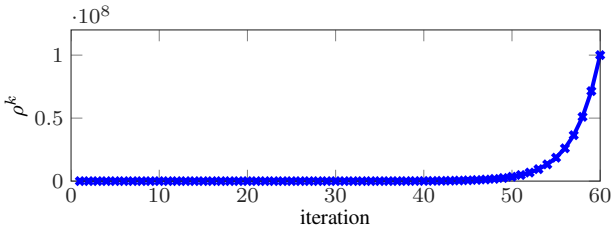

Fig. 1. The value of ρ^k in each iteration.

In Table. 1, we compare all the algorithms used by using the normalized mean square error (nMSE), defined as

$$\text{nMSE} = \frac{1}{100}\sum_{n=1}^{100}\frac{\|\hat{\boldsymbol{S}}_n\hat{\boldsymbol{A}}_n^T - \boldsymbol{S}\boldsymbol{A}^T\|_F^2}{\|\boldsymbol{S}\boldsymbol{A}^T\|_F^2}, \qquad (5)$$

where $\hat{\boldsymbol{A}}_n$ and $\hat{\boldsymbol{S}}_n$ denote the estimates of $\boldsymbol{A}$ and $\boldsymbol{S}$, respectively, for the nth simulation. The NMF-DCD and dNMF-DCD yield, the lowest, almost identical nMSE, with a low standard deviation.

Table 1. The nMSE ($\pm$ one standard deviation) for the simulated data.

Algorithm	nMSE
NMF-L&S	$0.00135 \pm 5 \cdot 10^{-5}$
NMF-ALS	$0.00134 \pm 2.4 \cdot 10^{-9}$
NMF-DCD	$0.00131 \pm 7.5 \cdot 10^{-7}$
dNMF-DCD	$0.00131 \pm 3.5 \cdot 10^{-7}$

4.2. Real Data

Fig. 2. The generated RGB image of the Urban data set. The data is split into 6 regions, shown here and marked 1-6.

Here the dNMF-DCD algorithm is evaluated using a real hyperspectral data set. We will compare the results obtained by the dNMF-DCD algorithm to the results obtained using NMF-DCD algorithm.

The hyperspectral image used here is a HYDICE hyperspectral image[2] of an urban landscape. The number of spectral bands in this data set is 210 and covers the 400-2500nm spectral range.

This image is 307×307 pixels and the whole image is used. Bands numbered [89, 90, 103-109, 130-152, 204-210] are identified as water absorption or low SNR bands and are removed, resulting in 171 usable bands. The $\boldsymbol{Y}$ matrix is thus of dimensions $307^2 \times 171$. An RGB image, generated using the hyperspectral data, is shown in Fig. 2. The RGB image is created by using specific spectral bands from the data set, to represent the red, green, and blue channels of the RGB image. The image is split into $N = 6$ regions, marked 1-6 in the RGB image. The data matrix, $\boldsymbol{Y}$ is thus split into 6 matrices.

The dNMF-DCD algorithm is run for 60 iterations, using the same values of ρ^k, as in the previous subsection.

Using the algorithm proposed in [15], the rank is estimated to be $r = 8$. The algorithms are initialized randomly

[2] http://www.agc.army.mil/hypercube/

in the same manner as was done in the previous subsection.

In Fig. 3, two column in the $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{S}}$ are shown, using NMF-DCD and dNMF-DCD, respectively. The columns of $\hat{\boldsymbol{A}}$ are shown as a plot and the columns of $\hat{\boldsymbol{S}}$ are shown as intensity maps.

It can be seen that the solutions found by dNMF-DCD and NMF-DCD are very similar but not identical. There are some minor differences between the two solutions. There are no discontinuities visible in the intensity maps at the locations where the data was split up (vertical yellow lines in Fig. 2).

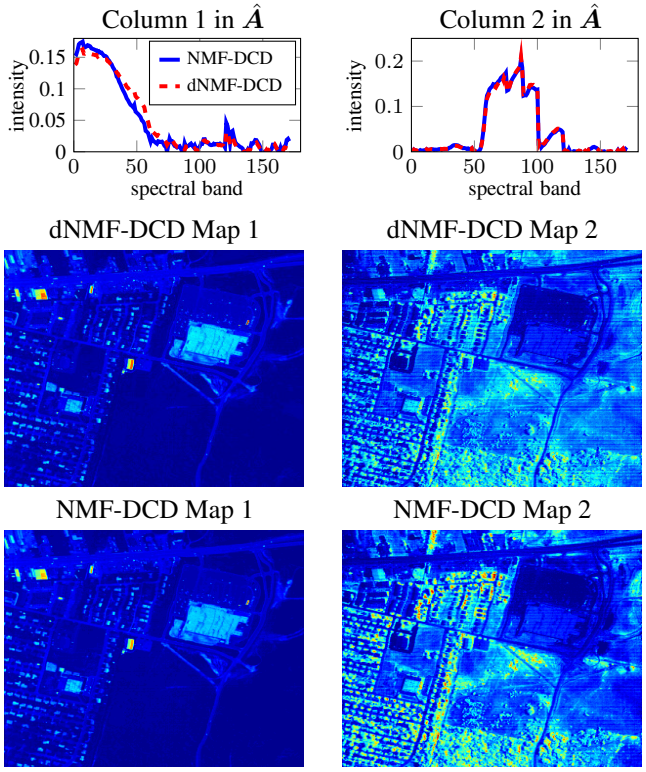

Fig. 3. The top row shows two columns of the $\hat{\boldsymbol{A}}$ matrix, estimates by NMF-DCD and dNMF-DCD, respectively. The middle row shows the corresponding columns in the $\hat{\boldsymbol{S}}$ (dNMF-DCD) matrix reshaped into a 307×307 intensity map. The bottom row shows the same column in in the $\hat{\boldsymbol{S}}$ (NMF-DCD) matrix.

5. CONCLUSIONS

In this paper, we developed for the first time a distributed version of NMF using ADMM, and the dyadic cyclic descent algorithm. Using simulated data, the algorithms are evaluated and compared to two variants of NMF. The proposed algorithms outperformed both of these NMF variants. In addition, the algorithm was also illustrated using a real remote sensing hyperspectral image. It is important to note that the new procedure could be used on very large problems whereas the standard algorithm could not be implemented due to memory problems.

6. REFERENCES

[1] D. D. Lee and S.H. Seung, "Algorithms for Non-Negative Matrix Factorization," *Advances in Neural Information Processing System*, vol. 13, pp. 556–562, 2001.

[2] D. D. Lee and H. Sebastian Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct 1999.

[3] Patrik O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *J. Mach. Learn. Res.*, vol. 5, pp. 1457–1469, Dec. 2004.

[4] A. Owen and P. Perry, "Bi-cross-validation of the SVD and the nonnegative matrix factorization," *The annals of applied statistics*, vol. 3, no. 2, pp. 564–594, 2009.

[5] I. Psorakis, S. Roberts, M. Ebden, and B. Sheldon, "Overlapping community detection using bayesian non-negative matrix factorization," *Phys. Rev. E*, vol. 83, no. 6, 2011.

[6] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 2, pp. 354 –379, April 2012.

[7] S. Boyd, N. Parikh, E. Chu, and B. Peleato, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundation and trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[8] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R J. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," in *Computational Statistics and Data Analysis*, 2006, pp. 155–173.

[9] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, June 1994.

[10] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Comput.*, vol. 19, no. 10, pp. 2756–2779, Oct. 2007.

[11] C. Liu, H. c. Yang, J. Fan, L.-W. He, and Y.-M. Wang, "Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce," *Proceedings of the 19th International World Wide Web Conference*, April 2010.

[12] A. Aduroja, I.D. Schizas, and V. Maroulas, "Distributed principal component analysis in sensor networks," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'13)*, Vancouver, Canada, 2013.

[13] M. Ulfarsson and V. Solo, "Sparse component analysis via dyadic cyclic descent," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'14)*, Florence, Italy, 2014.

[14] M. Sedeghi, M. Babaie-Zadeh, and C. Jutten, "Learning overcomplete dictionary based on atom-by-atom updating," *IEEE Trans. Signal Proc.*, vol. 62, no. 4, 2014.

[15] M.O. Ulfarsson and V. Solo, "Tuning parameter selection for nonnegative matrix factorization," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'13)*, Vancouver, Canada, 2013.