Documentation for the MATLAB code from the book 'Nonnegative Matrix Factorization'

Nicolas Gillis*
Department of Mathematics and Operational Research
Faculté Polytechnique, Université de Mons
Rue de Houdain 9, 7000 Mons, Belgium

Abstract

This paper contains a high-level documentation for the MATLAB codes provided with the book 'Nonnegative Matrix Factorization' [9] and available from `https://gitlab.com/ngillis/nmfbook/`. These codes contain algorithms for various NMF models, with examples of applications which are described in the book. This documentation hopefully allows you to navigate more easily through the codes. For more details, we encourage the reader to use the help function (or look at the m-files directly) to have all the details about a particular algorithm. Also, we encourage the interested reader to read the book, where the models, algorithms and numerical examples are described thoroughly. Please report any bug to nicolas.gillis@umons.ac.be.

Contents

*Email: nicolas.gillis@umons.ac.be. NG acknowledges the support by the European Research Council (ERC starting grant no 679515), and by the Fonds de la Recherche Scientifique - FNRS and the Fonds Wetenschappelijk Onderzoek - Vlanderen (FWO) under EOS Project no O005318F-RG47.

1 Main folder 'NMFbook'

The main folder contains two files:

- `Install.m` that adds the paths of all subfolders. This allows you to run any code from this package (since some functions use other functions located in different folders).
- The License which is CC0 (Creative Commons Zero).

It contains 4 folders: data sets, algorithms, examples by chapter, and utils, described in the following four sections.

2 Folder 'data sets'

This folder contains 9 widely used data sets in the NMF literature which are used in the book for the numerical experiments.

- `CBCL.mat`: This is a 361×2429 pixel-by-image matrix whose columns are gray-scale facial images of size 19×19; see Figure 1 for a sample.

Figure 1: Sample images of the CBCL data set.

This is the famous data set that Lee and Seung used to examplify the ability of NMF to learn a part-based representation [16]; see also the introduction of the book and `CBCL.m` in Section 4.1.

- `classic.mat`: This is a 7094×2429 word count matrix, each column corresponds to a word and each row to a document [25]. This data set is used in Section 4.8 to compare NMF algorithms on sparse data sets.

- `karate.mat`: This is the famous Zachary karate club data set of a graph containing 34 vertices and 78 edges; see Figure 2. It is used in Section 4.5 to show the ability of symmetric NMF to

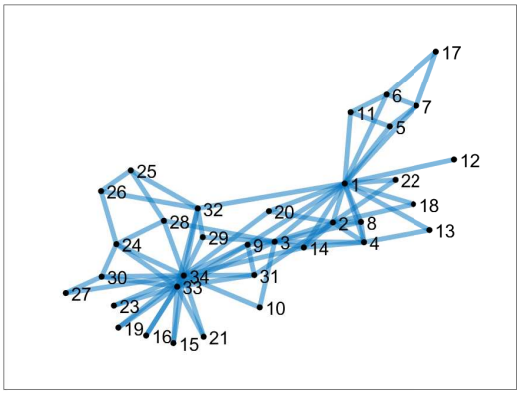

Figure 2: Zachary karate club graph.

identify communities in graphs; see `symNMF_karate.m`.

- `microarrayIFNbeta.mat`: This is a 52×189 gene-by-patient matrix whose entries are gene expression levels. It is described in the paper [3], and available from `https://doi.org/10.1371/journal.pbio.0030002.sd001`. It is used in Section 4.9 for gene expression analysis with NMF; see `Microarray.m`.

- `piano_Mary.mat`: This is 129×586 time-by-frequency matrix (a spectrogram) representing the song 'Mary had a little lamb' played at the piano. It is used in Section 4.1 to illustrate the ability of NMF to decompose audio signals (`Mary_piano.m`), and comes from the paper [17]; see Figure 3 for an illustration.

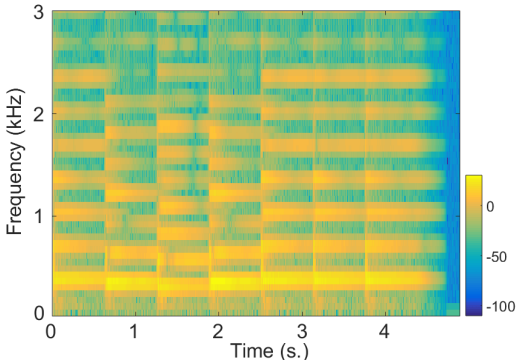

Figure 3: Spectrogram of the song 'Mary had a little lamb' in dB.

- `RamanSMCR.mat`: This is 2478×5 time-by-component matrix representing the Raman spectra measurements over time of a synthetic chemical reaction. It is used in Section 4.9 to illustrate the ability of NMF to extract the pure components automatically (`Raman.m`), and comes from the paper [18].

- `Swimmer.mat`: This is a 220×256 pixel-by-image matrix whose columns are gray-scale images of a swimmer whose limbs take each 4 different positions; see Figure 4. It was used in [4] to

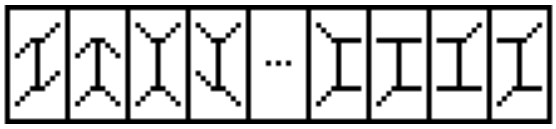

Figure 4: Sample images of the swimmer data set.

study the uniqueness of NMF. The file `Swimmer.m` in Section 4.1 compares various NMF models on this data set.

- `tdt2_top30.mat`: This is a 9394×19528 word count matrix, each column corresponds to a word and each row to a document [25]. This data set is used in Section 4.8 to compare NMF algorithms on sparse data sets.

- `Urban.mat`: This is a wavelength-by-pixel 162×94249 hyperspectral image of a Walmart in Copperas Cove (Texas); see Figure 5 for an illustration. It is used in many places in the book to compare various NMF models; see Sections 4.1, 4.4 and 4.5.

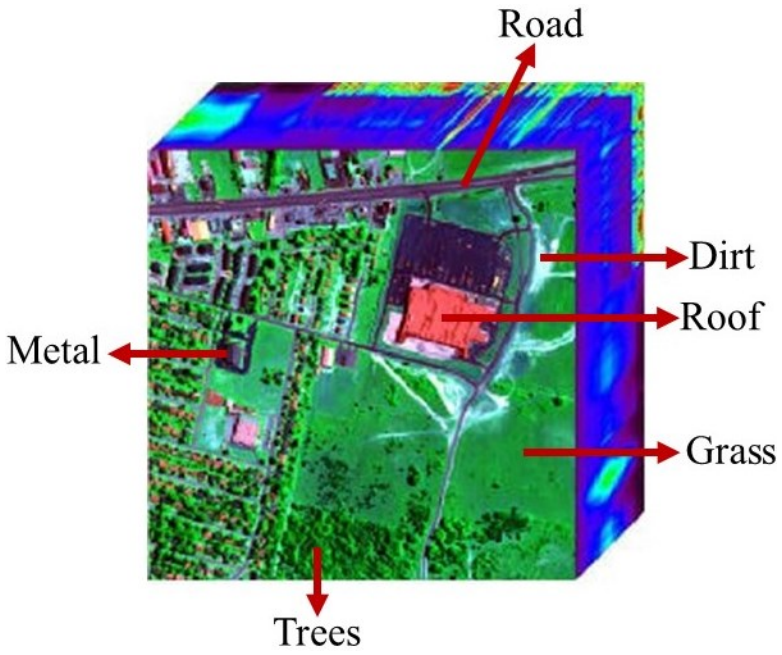

Figure 5: Urban hyperspectral image.

- *NEW* `Moffet.mat`: This is a wavelength-by-pixel 159×2500 hyperspectral image; see Figure 6 for an illustration. It is a new data set, not used in the book, that can be used to compare NMF algorithms; see Section 4.4.

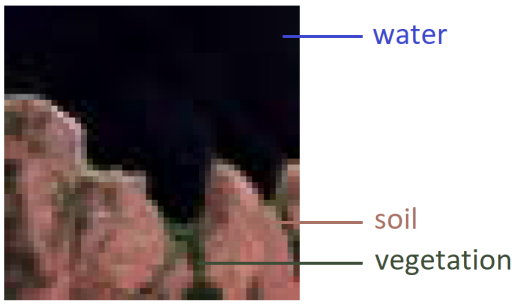

Figure 6: Moffet hyperspectral image.

3 Folder 'algorithms'

This folder contains 14 folders, each of them corresponding to a different NMF model described in the book and for which algorithms are provided. They are described in the following 14 sections.

3.1 beta-NMF with MU

The main function of this folder is `betaNMF.m` which contains multiplicative updates (MU) to solve NMF with any β-divergences (β-NMF), that is, to solve

$$\min_{W \geq 0, H \geq 0} D\left(X, WH\right),$$

where

$$D(X, WH) = \sum_{i,j} d_\beta\left(X_{i,j}, (WH)_{i,j}\right),$$

with

$$d_\beta(x|y) = \begin{cases} \frac{1}{\beta(\beta-1)}\left(x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}\right) & \text{if } \beta \in \mathbb{R} \setminus \{0,1\}, \\ x \log \frac{x}{y} - x + y & \text{if } \beta = 1, \\ \frac{x}{y} - \log \frac{x}{y} - 1 & \text{if } \beta = 0. \end{cases}$$

For $\beta = 2$, this is the Frobenius norm (for which much more efficient algorithms exist, see Section 3.3), for $\beta = 1$ the Kullblack-Leibler divergence, and for $\beta = 0$ the Itakura-Saito divergence.

This code is used in Section 4.5 to compute NMF solutions for various β-divergence to illustrate and interesting properties: for small β, β-NMF tends to overapproximate the input matrix while, for large β, it tends to underapproximate it; see `Example51.m`.

3.2 Exact NMF

The main function in this folder is `exactNMFheur.m` which attempts to compute an exact NMF of a given nonnegative input matrix X, such that $X = WH$. It is based on several heuristics (multiple random initializations, simulated annealing, a rank-by-rank heuritic). This code is copied from the paper of Vandaele et al. [22]; see also `https://sites.google.com/site/exactnmf/`.

This code is used for example in Section 4.2 to compute the exact NMF of the matrix corresponding to two nested hexagons; see `ExactNMF_nested_hexagons.m`.

3.3 Fro NMF

This folder contains algorithms for the most standard NMF model, namely

$$\min_{W \geq 0, H \geq 0} \|X - WH\|_F^2,$$

where $\|X-WH\|_F^2 = \sum_{i,j}(X-WH)_{i,j}^2$ is the Frobenius norm of $X-WH$. The main file is `FroNMF.m`.

This problem, referred to as Fro NMF, is solved via block coordinate descent (BCD) methods, where the subproblems are nonnegative least squares (NNLS) problems of the form

$$\min_{H \geq 0} \|X - WH\|_F^2.$$

Depending on the algorithm used to tackle the NNLS, we obtain various NMF algorithms.

The NNLS folder contains the implementation of five algorithms to tackle NNLS, namely: the alternating direction method of multipliers (ADMM), a fast projected gradient method (FPGM), a coordinate descent method (HALSupdt), the multiplicative updates (MU), and an active set method.

These algorithms are compared in Section 4.8 on the CBCL, classic and tdt2_top30 data sets; see `FroNMF_algo_comparison.m`.

3.4 hierarchical NMF

The main function is `hierclust2nmf.m` which computes a hierarchical rank-two NMF decomposition of a given nonnegative input matrix X. This code is from [12]. It is used in `Urban.m` in Section 4.1 on the Urban data set.

3.5 min-vol NMF

This folder contains the function `minvolNMF.m` to solve minimum-volume (min-vol) NMF

$$\min_{W \geq 0, H \geq 0} \|X - WH\|_F^2 + \lambda \operatorname{logdet}(W^\top W + \delta I),$$

with three possible constraints: (1) $H^\top e = e$, (2) $He = e$, or (3) $W^\top e = e$, leading to three different min-vol NMF models which are described in details in the book. The positive constants λ and δ are parameters.

This code is used in Section 4.4 (`minvolNMF_Urban.m`) to compare these three models on the Urban data set.

3.6 nonnegative matrix underapproximation (NMU)

The MATLAB code `recursiveNMU.m` described in [10] computes an NMF sequentially, solving at each step a rank-one NMU problem of the form

$$\min_{w\geq 0, h\geq 0} \|X - wh^\top\|_F^2 \quad \text{such that} \quad wh^\top \leq X.$$

It is used to compute an NMF of the CBCL data set in Section 4.5; see `CBCL_NMU.m`.

3.7 orthogonal NMF

The MATLAB code `alternatingONMF.m` described in [20] solves orthogonal NMF (ONMF):

$$\min_{W\geq 0, H\geq 0} \|X - WH\|_F^2 \quad \text{such that} \quad HH^\top = I_r,$$

where I_r is the identity matrix of size r. This algorithm optimizes W and H alternatively: W has a closed-form solution, namely $XH^\top$, while H can be computed by assigning each data point to its closest (in terms of angle) column of W; this is done in `orthNNLS.m`.

It is used to compute an ONMF of the CBCL and Urban data sets in Section 4.4; see `CBCL_ONMF.m` and `CBCL_Urban.m`.

3.8 projective NMF

The MATLAB code `projectiveNMF.m` described in [24] solves projective NMF

$$\min_{W\geq 0} \|X - WW^\top X\|_F^2,$$

using MU.

It is applied on the CBCL data set in Section 4.5; see `CBCL_projectiveNMF.m`.

3.9 rank-two NMF

The MATLAB code `Rank2NMF.m` implements Algorithm 4.1 in the book and that solves the rank-two NMF problem. If the input matrix has rank two, it is guaranteed to provide an exact solution; otherwise, it constructs an approximation based on the truncated singular value decomposition.

A numerical example is provided in `test_Rank2NMF.m`.

3.10 semi-NMF

The MATLAB code `semiNMF.m` described in [13] solves semi-NMF:

$$\min_{W,H\geq 0} \|X - WH\|_F^2,$$

where W is not constrained to be nonnegative. The file `test_semiNMF.m` runs this algorithm on two simple examples.

The code `semiNMF.m` is applied on the CBCL data set in Section 4.5; see `CBCL_semiNMF.m`.

The code `seminonnegativerank.m` computes the semi-nonnegative rank of a matrix, that is, the smallest r such that an exact semi-NMF of size r exists.

3.11 separable NMF

Separable NMF is the following problem: given a matrix X and a factorization rank r, find an index set $\mathcal{K}$ of size r and a nonnegative matrix H such that

$$X \quad \approx \quad X(:, \mathcal{K})H.$$

This folder contains 4 widely used greedy algorithms for separable NMF, namely vertex component analysis (VCA) [19], the successive projection algorithm (SPA) [1], the successive nonnegative projection algorithm (SNPA) [8], and fast anchor words (FAW) [2]. It also contains preconditioning based on a minimum-volume ellipsoid problem that improves the performance of these algorithms [15]. These algorithms are compared on various data sets in Section 4.7; see `compare_greedy_sepNMFalgo.m`.

Separable NMF algorithms can also be used as an initialization technique for NMF (if the input matrix does not satisfy the above model); see for example Section 4.8 where SNPA is compared to random initialization in `FroNMF_init_randvsSNPA.m` on the CBCL data set.

This folder also contains `septrisymNMF.m` that solves separable symmetric nonnegative matrix tri-factorization (tri-symNMF):

$$X \approx WSW^\top,$$

where $X \geq 0$ is symmetric, $W \geq 0$ is separable (it contains the identity as a submatrix, up to scaling), and $S \geq 0$ is symmetric. The file `septrisymNMF_example.m` in Section 4.7 provides an example of using this algorithm on a noiseless synthetic data set.

3.12 sparse NMF

This folder contains the file `sparseNMF.m` that solves a particular sparse NMF model, namely

$$\min_{W \geq 0, H \geq 0} \|X - WH\|_F^2 \quad \text{such that} \quad \text{sp}(W) \geq s_W \text{ and } \text{sp}(H^\top) \geq s_H,$$

where $\text{sp}(W)$ is the average of the Hoyer sparsity of the columns of W, while $s_W, s_H \in [0, 1]$ are given parameters that control the sparsity of W and H. Recall that the Hoyer sparsity of a vector $x \in \mathbb{R}^n$ is given by

$$\text{sp}(x) = \frac{\sqrt{n} - \frac{\|x\|_1}{\|x\|_2}}{\sqrt{n} - 1} \in [0, 1].$$

This code is based on accelerated projected gradient descent and is described in [14].

It is applied on the CBCL data set in Section 4.5 with $s_W = 0.85$ and $s_H = 0$ (no sparsity constraint on H); see `CBCL_sparseNMF.m`.

3.13 symmetric NMF

This folder contains the file `symNMF.m` that solves symmetric NMF: given a symmetric $X \geq 0$ and a factorization rank r, solve

$$\min_{H \geq 0} \|X - HH^T\|_F^2,$$

where H has r columns. This code is based on coordinate descent and is described in [23].

It is used in Section 4.5 to cluster the vertices of the Zachary karate club data set; see `symNMF_karate.m`.

3.14 weighted low-rank approximations (WLRA)

The code `WLRA.m` solves the weighted low-rank approximations (WLRA) problem: Given an input matrix X (not necessarily nonnegative), a weight matrix $P \geq 0$, and a factorization rank r, it solves

$$\min_{W,H} \sum_{i,j} P_{i,j}(X - WH)_{i,j}^2,$$

where (W, H) is a rank-r factorization. Note that the code can handle the problem with nonnegativity constraints, and is based on coordinate descent.

It is used in Section 4.9 to factor a small 6-by-5 synthetic matrix; see `RecomSys.m`.

This folder also contains `Example.m` that solves WLRA on randomly generated synthetic data sets with 50% missing entries.

4 Folder 'examples by chapter'

This folder contains all the numerical examples presented in the book, organized chapter by chapter. It also contains some algorithms that are discussed in the corresponding chapters but that are not NMF algorithms. For example, Chapter 2 contains a function that allows to display the geometric problem corresponding to the NMF of a rank-3 matrix, Chapter 3 contains functions that compute lower bounds for the nonnegative rank, and Chapter 4 contains a function that allows one to check the sufficiently scattered condition and a necessary condition for it.

4.1 Chapter 1 - Introduction

It contains the application of NMF on five data sets from Section 2:

1. `CBCL.m`: facial feature extraction on the CBCL data set using standard NMF.
2. `Mary_piano.m`: audio source separation on the piano piece 'Mary had a little lamb' using β-NMF with $\beta = 1$.
3. `Swimmer.m`: feature extraction on the swimmer data sets (the features are the body and limbs of the swimmer) using standard NMF, separable NMF, ONMF, min-vol NMF, and NMU.
4. `tdt2_top30.m`: topic extraction on the TDT2 (top 30) data set. NMF is able to extract automatically subset of words that correspond to topics. Note that this example is not presented in the book.
5. `Urban.m`: blind hyperspectral image unmixing on the Urban data set using hierarchical NMF.

4.2 Chapter 2 - Exact NMF

This folder contains two files:

1. `ExactNMF_nested_hexagons.m`: it factorizes exactly the 6-by-6 matrix

$$X_a = \frac{1}{6a}\begin{pmatrix} 1 & a & 2a-1 & 2a-1 & a & 1 \\ 1 & 1 & a & 2a-1 & 2a-1 & a \\ a & 1 & 1 & a & 2a-1 & 2a-1 \\ 2a-1 & a & 1 & 1 & a & 2a-1 \\ 2a-1 & 2a-1 & a & 1 & 1 & a \\ a & 2a-1 & 2a-1 & a & 1 & 1 \end{pmatrix},$$

   that depends on the parameter $a > 1$. It also displays the geometric interpretation: as explained in the book, this Exact NMF problem corresponds to the geometric problem of finding a polytope nested between two regular hexagons. To do so, it uses the function described below.

2. `NPPrank3matrix.m`: this codes represents geometrically, in two dimensions, the nested polytope (NPP) instance corresponding to the restricted Exact NMF problem of a rank-three matrix X; see Theorem 2.11 in the book. This code is adapted from the code presented in [11].

4.3 Chapter 3 - Nonnegative rank

The nonnegative rank of a nonnegative matrix X, denoted $\text{rank}_+(X)$, is the smallest r such that there exists $W \geq 0$ with r columns and $H \geq 0$ with r rows such that $X = WH$.

This folder contains five numerical examples:

- The first three, `bound_nnr_linEDM.m`, `bound_nnr_octagon.m` and `bound_nnr_Thomas.m`, provide the different lower bounds described in Section 4.3.1 below for linear Euclidean distance matrices (EDMs) of the form $X(i,j) = (i-j)^2$ for all i, j, the slack matrix of the regular octagon, and the 4-by-4 matrix of Thomas, respectively.

- `rec_cov_linEDM.m`: it computes and displays the rectangle covering of linear EDMs.

- `UDISJ.m`: it generates a 2^n-by-2^n unique disjointness matrix (UDISJ), and tries to compute an Exact NMF of size $2^n - 1$ using `exactNMFheur.m`; this is most likely not possible as it is conjectured that $\text{rank}_+(X) = 2^n$.

4.3.1 Lower bounds

This folder contains 5 algorithms that allow you to compute lower bounds for the nonnegative rank:

1. `geometric_bound.m`: given the restricted nonnegative rank of the matrix, it computes the geometric lower bound from [11].

2. `hyperplane_separation_bound.m`: given $Z \in \mathbb{R}^{m \times n}$, it computes

$$\alpha(Z) \quad = \quad \max_{R \in \{0,1\}^{m \times n}} \langle Z, R \rangle \text{ such that } \text{rank}(R) = 1.$$

This allows you to compute a lower bound for the nonnegative rank of any m-by-n matrix since

$$\text{rank}_+(X) \geq \frac{\langle Z, X \rangle}{\alpha(Z)\|X\|_\infty},$$

see for example [21].

3. `nonneg_nuclear_norm_bound.m` computes a lower bound described in [5] based on the so-called nonnegative nuclear norm, and that relies on solving a semidefinite program. You need to install CVX `http://cvxr.com/cvx/` to run this code.

4. `rec_cov_bound.m` computes the rectangle covering bound; see for example [7]. You need the MATLAB optimization package and in particular `intlinprog.m` to run this code.

5. `self_scaled_bound.m` computes a lower bound described in [6] that relies on sum of squares and semidefinite programming. You need to install CVX `http://cvxr.com/cvx/` to run this code.

It also contains the file `lowerbounds_nnr.m` that computes all these lower bounds at once. For the hyperplane separation bound that depends on the choice of Z, it is chosen as follows

$$Z(i,j) = \begin{cases} -1000 & \text{for } X(i,i) = 0, \\ X(i,j) - 0.5 & \text{otherwise }, \end{cases}$$

which seems to work relatively well for the integer matrices considered in the book.

4.4 Chapter 4 - Identifiability

This folder contains two algorithms along with two numerical examples:

1. `isSSC.m` checks whether a given matrix satisfies the sufficiently scattered condition (SSC) using a brute-force approach. This code is tested on a small 4-by-6 matrix in `checkSSC_H46_2sparse.m`, namely,

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

2. `SSC1_nec_cond.m` checks whether a given input matrix satisfies the necessary condition for the SSC; see Algorithm 4.2 and Theorem 4.28 in the book. The file `SSC1_nec_cond_illus.m` uses this code to generate phase transition plots that show the dependence between this necessary condition and the sparsity and the size of the input matrix (Figure 4.6 in the book).

It also contains three other numerical examples:

1. `minvolNMF_Urban.m` compares three min-vol NMF models on the Urban data set; see Section 3.3.

2. `minvolNMF_Moffet.m` compares two min-vol NMF models on the Moffet data set; see Section 3.3.

3. `ONMF_CBCL.m` applies ONMF on the CBCL data sets, initialized with SPA.

4. `ONMF_Urban.m` applies ONMF on the Urban data sets, initialized with SPA.

4.5 Chapter 5 - NMF models

In this section, various NMF models are applied to various data sets:

- The CBCL data set is decomposed using NMU (Section 3.6), projective NMF (Section 3.8), semi-NMF (Section 3.10), sparse NMF (Section 3.12), and nonnegative matrix tri-factorization via sparse NMF.
- `symNMF_karate.m` decomposes the Zachary karate club data set with symmetric NMF (Section 3.13).
- `Example51.m` applies β-NMF on sparse input matrices to show that, for small β (namely, $\beta = 0$), β-NMF tends to overapproximate the input matrix and, for larger β (namely, $\beta = 2$), β-NMF tends to underapproximate the input matrix.
- Projective NMF is applied to the Urban data set in `ProjectiveNMF_Urban.m`.

4.6 Chapter 6 - Complexity

This folder is empty as no numerical example is provided in Chapter 6.

4.7 Chapter 7 - Separable NMF

This folder contains four files:

1. `CBCL_sepNMF.m` applies NMF on the transpose of the CBCL data set to extract important pixels across the facial images.
2. `compare_greedy_sepNMFalgo.m` compares greedy separable NMF algorithms on various synthetic data sets.
3. `compare_LDR_MVESPA.m` compares three linear dimensionality reduction techniques applied before separable NMF on an ill-conditioned matrix.
4. `septrisymNMF_example.m` shows an example of applying separable tri-symNMF on a synthetic example.

4.8 Chapter 8 - NMF algorithms

This folder contains three files:

1. `FroNMF_algo_comparison.m` compares Fro NMF algorithms (Section 3.3) on various data sets.
2. `FroNMF_init_randvsSNPA.m` compares random initialization vs. SNPA on the CBCL data set.
3. `MU_vs_modifiedMU.m` shows that using a small lower bound $\epsilon > 0$ in the MU improves the performance, preventing the zero locking phenomenon.

4.9 Chapter 9 - Applications

This folder contains three files:

1. `Microarray.m` applies standard NMF on the microarrayIFNbeta data set.
2. `Raman.m` displays the decomposition (W, H) of the RamanSMCR data set.
3. `RecomSys.m` applies weighted NMF on the matrix

$$X = \begin{pmatrix} 2 & 3 & 2 & ? & ? \\ ? & 1 & ? & 3 & 2 \\ 1 & ? & 4 & 1 & ? \\ 5 & 4 & ? & 3 & 2 \\ ? & 1 & 2 & ? & 4 \\ 1 & ? & 3 & 4 & 3 \end{pmatrix}.$$

5 Folder 'utils'

This folder contains several useful functions; for example `affichage.m` that displays the columns of a matrix as images.

References

[1] Araújo, U., Saldanha, B., Galvão, R., Yoneyama, T., Chame, H., Visani, V.: The successive projections algorithm for variable selection in spectroscopic multicomponent analysis. Chemometrics and Intelligent Laboratory Systems **57**(2), 65–73 (2001)

[2] Arora, S., Ge, R., Halpern, Y., Mimno, D., Moitra, A., Sontag, D., Wu, Y., Zhu, M.: A practical algorithm for topic modeling with provable guarantees. In: Proceedings of the 30th International Conference on Machine Learning, pp. 280–288 (2013)

[3] Baranzini, S.E., Mousavi, P., Rio, J., Caillier, S.J., Stillman, A., Villoslada, P., Wyatt, M.M., Comabella, M., Greller, L.D., Somogyi, R., et al.: Transcription-based prediction of response to IFNβ using supervised computational methods. PLoS Biology **3**(1), e2 (2004)

[4] Donoho, D., Stodden, V.: When does non-negative matrix factorization give a correct decomposition into parts? In: Advances in Neural Information Processing Systems (NIPS), pp. 1141–1148 (2004)

[5] Fawzi, H., Parrilo, P.A.: Lower bounds on nonnegative rank via nonnegative nuclear norms. Mathematical Programming **153**(1), 41–66 (2015)

[6] Fawzi, H., Parrilo, P.A.: Self-scaled bounds for atomic cone ranks: applications to nonnegative rank and cp-rank. Mathematical Programming **158**(1-2), 417–465 (2016)

[7] Fiorini, S., Kaibel, V., Pashkovich, K., Theis, D.O.: Combinatorial bounds on nonnegative rank and extended formulations. Discrete Mathematics **313**(1), 67–83 (2013)

[8] Gillis, N.: Successive nonnegative projection algorithm for robust nonnegative blind source separation. SIAM Journal on Imaging Sciences **7**(2), 1420–1450 (2014)

[9] Gillis, N.: Nonnegative Matrix Factorization. SIAM (2020). URL `https://my.siam.org/Store/Product/viewproduct/?ProductId=32898069`

[10] Gillis, N., Glineur, F.: Using underapproximations for sparse nonnegative matrix factorization. Pattern Recognition **43**(4), 1676–1687 (2010)

[11] Gillis, N., Glineur, F.: On the geometric interpretation of the nonnegative rank. Linear Algebra and its Applications **437**(11), 2685–2712 (2012)

[12] Gillis, N., Kuang, D., Park, H.: Hierarchical clustering of hyperspectral images using rank-two nonnegative matrix factorization. IEEE Transactions on Geoscience and Remote Sensing **53**(4), 2066–2078 (2014)

[13] Gillis, N., Kumar, A.: Exact and heuristic algorithms for semi-nonnegative matrix factorization. SIAM Journal on Matrix Analysis and Applications **36**(4), 1404–1424 (2015)

[14] Gillis, N., Ohib, R., Plis, S., Potluru, V.: Grouped sparse projection. arXiv preprint arXiv:1912.03896 (2019)

[15] Gillis, N., Vavasis, S.A.: Semidefinite programming based preconditioning for more robust near-separable nonnegative matrix factorization. SIAM Journal on Optimization **25**(1), 677–698 (2015)

[16] Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature **401**, 788–791 (1999)

[17] Leplat, V., Gillis, N., Ang, M.S.: Blind audio source separation with minimum-volume beta-divergence NMF. IEEE Transactions on Signal Processing **68**, 3400–3410 (2020)

[18] Luce, R., Hildebrandt, P., Kuhlmann, U., Liesen, J.: Using separable nonnegative matrix factorization techniques for the analysis of time-resolved Raman spectra. Applied spectroscopy **70**(9), 1464–1475 (2016)

[19] Nascimento, J.M., Dias, J.M.: Vertex component analysis: A fast algorithm to unmix hyperspectral data. IEEE transactions on Geoscience and Remote Sensing **43**(4), 898–910 (2005)

[20] Pompili, F., Gillis, N., Absil, P.A., Glineur, F.: Two algorithms for orthogonal nonnegative matrix factorization with application to clustering. Neurocomputing **141**, 15–25 (2014)

[21] Rothvoss, T.: The matching polytope has exponential extension complexity. In: Proceedings of the forty-sixth annual ACM symposium on Theory of computing, pp. 263–272. ACM (2014)

[22] Vandaele, A., Gillis, N., Glineur, F., Tuyttens, D.: Heuristics for exact nonnegative matrix factorization. Journal of Global Optimization **65**(2), 369–400 (2016)

[23] Vandaele, A., Gillis, N., Lei, Q., Zhong, K., Dhillon, I.: Efficient and non-convex coordinate descent for symmetric nonnegative matrix factorization. IEEE Transactions on Signal Processing **64**(21), 5571–5584 (2016)

[24] Yang, Z., Oja, E.: Linear and nonlinear projective nonnegative matrix factorization. IEEE Transactions on Neural Networks **21**, 734–749 (2010)

[25] Zhong, S., Ghosh, J.: Generative model-based document clustering: a comparative study. Knowledge and Information Systems **8**(3), 374–384 (2005)