

Factorisation nonnégative avec des fonctions rationnelles : partitions efficaces et méthodes hybrides

Cécile HAUTECOEUR¹, François GLINEUR¹

¹*ICTEAM & CORE institutes, UCLouvain Louvain-la-Neuve, Belgique*
 cecile.hautecoeur@uclouvain.be, francois.glineur@uclouvain.be

Résumé – La Factorisation Matricielle Nonnégative (NMF) est une méthode commune pour analyser des données matricielles nonnégatives. Pour améliorer la qualité de la factorisation, il peut être intéressant d'imposer une certaine structure aux facteurs outre la nonnégativité. Dans ce travail, afin de trouver des facteurs lisses, nous imposons à ceux-ci d'être la discrétisation de fonctions rationnelles. Ce nouveau problème peut être formulé comme un problème de moindre carrés non-contraint, ce qui permet de le résoudre à l'aide de méthodes usuelles. De plus, il est connu que partitionner les variables en blocs et utiliser une méthode de descente par blocs peut accélérer la convergence à une solution. Par conséquent, nous présentons trois partitions du problème, les comparons et montrons qu'associer deux partitions conduit à la résolution la plus efficace.

Abstract – Nonnegative Matrix Factorization (NMF) is a common method to analyze nonnegative matrix data. To improve the quality of the factorization, it can be interesting to impose additional structure on the factors besides the nonnegativity. In this work, to identify smooth factors, we require them to be discretization of rational functions. This new model can be formulated as an unconstrained least squares problem, allowing it to be solved with standard optimization solvers. Moreover, it is well known that partitioning variables into blocs and using bloc descend methods may accelerate convergence to a solution. Therefore, we present three partitions, compare them and show that combining two partitions leads to the most efficient solution.

NMF avec des fonctions rationnelles

La Factorisation Matricielle Nonnégative (NMF) est une méthode de décomposition linéaire de données permettant de compresser, filtrer le bruit, et extraire des facteurs caractéristiques. En pratique, la NMF exprime une matrice de données $Y \in \mathbb{R}^{m \times n}$ comme le produit de deux matrices nonnégatives avec un faible nombre de colonnes r , $A \in \mathbb{R}_+^{m \times r}$ et $X \in \mathbb{R}_+^{n \times r} : Y \simeq AX^\top$. Comme r est supposé plus petit que m et n , le produit AX^\top est de rang plus faible que Y , ce qui permet de filtrer le bruit. Si chaque colonne de Y est nonnégative et peut raisonnablement résulter de la somme d'un petit nombre de signaux de base nonnégatifs, imposer la nonnégativité permet de retrouver des facteurs A et X plus réalistes. Par exemple, dans des images hyperspectrales les colonne de Y contient la réponse spectrale d'un certain pixel, et cette réponse spectrale est la somme pondérée des réponses spectrales de tous les matériaux présent dans le pixel.

Définition 1. NMF : Soit une matrice de données $Y \in \mathbb{R}^{m \times n}$ et un rang r . Trouver deux matrices nonnégatives $A \in \mathbb{R}_+^{m \times r}$ et $X \in \mathbb{R}_+^{n \times r}$ telles que

$$A, X = \underset{A \in \mathbb{R}_+^{m \times r}, X \in \mathbb{R}_+^{n \times r}}{\operatorname{argmin}} \|Y - AX^\top\|_F^2. \quad (1)$$

Ce travail a été soutenu par le Fonds de la Recherche Scientifique - FNRS et le Fonds Wetenschappelijk Onderzoek - Vlaanderen sous le projet EOS n°30468160.

Supposons que chaque colonne de Y , $Y_{:j}$, peut être vue comme l'échantillonnage d'un signal y_j sur m points $\tau = \{\tau_i\}_{i=1}^m$, i.e. $Y_{:j} = y_j(\tau)$ et $Y_{ij} = y_j(\tau_i)$. C'est le cas par exemple des images hyperspectrales, où τ est l'ensemble des longueurs d'onde observées et y_j est le spectre de réflectance pour le pixel j . Si les signaux dans Y peuvent raisonnablement être décrits comme des fonctions rationnelles, on peut tenter d'inclure cette information dans la NMF. Ainsi, on peut imposer à A de contenir dans chacune de ses colonnes l'évaluation sur τ de fonctions rationnelles nonnégatives. Dans le cadre de ce travail, les degrés du numérateur et du dénominateur d_1 et d_2 des fonctions rationnelles sont fixés. Si $\mathcal{R}_+^{(d_1, d_2)}[\tau_{\min}, \tau_{\max}]$ est l'ensemble des fonctions rationnelles de degré (d_1, d_2) nonnégatives sur l'intervalle $[\tau_{\min}, \tau_{\max}]$, on impose donc que chaque colonne de A puisse s'écrire comme : $A_{:i} = f_i(\tau)$ où $f_i \in \mathcal{R}_+^{(d_1, d_2)}[\tau_{\min}, \tau_{\max}] \forall i$. Cette contrainte sera notée $A_{:i} \in \mathcal{R}_+$ dans ce qui suit pour alléger le texte. Ce problème a été présenté dans [3]

Définition 2. NMF avec des fonctions rationnelles : Soit une matrice de données $Y \in \mathbb{R}^{m \times n}$, un rang r et l'ensemble des fonctions rationnelles nonnégatives \mathcal{R}_+ défini ci-dessus. Trouver deux matrices nonnégatives $A \in \mathbb{R}_+^{m \times r}$ et $X \in \mathbb{R}_+^{n \times r}$, avec A contenant dans chacune de ses colonnes un élément de \mathcal{R}_+ , telles que

$$A, X = \underset{A_{:i} \in \mathcal{R}_+ \forall i, X \in \mathbb{R}_+^{n \times r}}{\operatorname{argmin}} \|Y - AX^\top\|_F^2. \quad (2)$$

Fonctions rationnelles nonnégatives

Une fonction rationnelle est un quotient de deux polynômes : $r(t) = \frac{n(t)}{g(t)}$. Toute fonction rationnelle $r(t)$ nonnégative sur un intervalle I peut être décrite sans perte de généralité par un numérateur nonnégatif sur I et un dénominateur strictement positif sur I . Supposons que $I = [-1, 1]$ et que le degré de n est pair, $d_1 = 2d'_1$. Si $n(t)$ est nonnégatif sur I , on sait qu'il existe deux polynômes $a_1(t)$ et $a_2(t)$ de degré d'_1 et $d'_1 - 1$ tels que [2, 4] :

$$n(t) = a_1(t)^2 + (1 - t^2)a_2(t)^2. \quad (3)$$

Nous avons choisi d'exprimer les polynômes dans la base de Chebyshev, qui est plus stable que la base des monômes. Si on désire évaluer un polynôme sur les points $\boldsymbol{\tau}$, on peut construire la matrice de Vandermonde pour la base de Chebyshev de degré d_1 aux points $\boldsymbol{\tau}$, $V_{\boldsymbol{\tau}}^{d_1}$. Si les coefficients dans cette base sont $\mathbf{n} \in \mathbb{R}^{d_1}$ on a alors :

$$n(\boldsymbol{\tau}) = \{n(\tau_i)\}_{i=1}^m = V_{\boldsymbol{\tau}}^{d_1} \mathbf{n}. \quad (4)$$

Pour imposer la stricte positivité du dénominateur, nous le représentons sans perte de généralité, comme un polynôme nonnégatif auquel on ajoute $\epsilon > 0$ partout. Ainsi, une fonction rationnelle de degré $(2d'_1, 2d'_2)$, nonnégative sur I , évaluée aux point $\boldsymbol{\tau}$, peut être exprimée comme suit, avec $\mathbf{c} = \{\mathbf{a} \in \mathbb{R}^{d'_1+1}, \mathbf{b} \in \mathbb{R}^{d'_1}, \mathbf{d} \in \mathbb{R}^{d'_2+1}, \mathbf{e} \in \mathbb{R}^{d'_2}\}$:

$$r_{\boldsymbol{\tau}}(\mathbf{c}) = \frac{(V_{\boldsymbol{\tau}}^{d'_1} \mathbf{a})^2 + (1 - \boldsymbol{\tau}^2)(V_{\boldsymbol{\tau}}^{d'_1-1} \mathbf{b})^2}{(V_{\boldsymbol{\tau}}^{d'_2} \mathbf{d})^2 + (1 - \boldsymbol{\tau}^2)(V_{\boldsymbol{\tau}}^{d'_2-1} \mathbf{e})^2 + \epsilon}. \quad (5)$$

On impose également que le dénominateur soit monique à l'aide de la contrainte $\mathbf{b}_1[d'_2 + 1] = \frac{\sqrt{8+b_2[d'_2]}}{2}$. D'autres représentations similaires existent pour des degrés impairs et des intervalles quelconques.

La représentation (5) permet de reformuler le problème (2) de manière non-contrainte, ce qui sera présenté dans la section suivante. Ainsi, le problème pourra être résolu par des méthodes usuelles d'optimisation, comme par exemple un algorithme à régions de confiance. Il faut cependant noter que le problème n'est pas convexe, et l'algorithme peut donc converger vers une solution qui n'est que localement optimale, et non pas de manière globale.

Trois partitions du problème

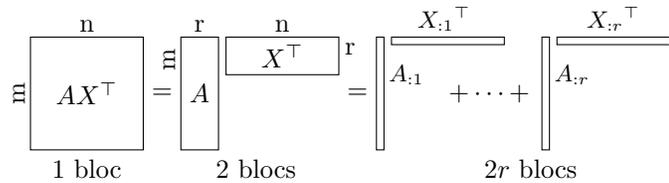


FIGURE 1 – Illustration des trois partitions considérées.

Partition en un seul bloc (LS_{rat}) On considère le problème (2) dans son entièreté, en le reformulant avec de nouvelles variables pour ne plus avoir de contraintes.

Les contraintes sur A , c'est-à-dire $A_{:i} \in \mathcal{R}_+$, peuvent être levées à l'aide de la représentation (5). De plus, chaque élément dans X , qui doit être nonnégatif, peut être exprimé comme étant un carré : $X = H^2$ où H représente le carré évalué élément par élément. On obtient ainsi le problème de moindre carrés non-contraint

$$\operatorname{argmin}_{\{\mathbf{c}_j\}_{j=1}^r, H \in \mathbb{R}^{n \times r}} \sum_{i=1}^n \left\| Y_{:i} - \sum_{j=1}^r r_{\boldsymbol{\tau}}(\mathbf{c}_j) H_{ij}^2 \right\|_F^2. \quad (6)$$

Partition en deux blocs (LS-2_{rat}) Les deux blocs considérés sont naturellement les facteurs A et X . Si A est fixé, le problème pour optimiser sur X est convexe et séparable sur chaque ligne de X puisque l'objectif est additif, et les contraintes sur chaque ligne de X sont indépendantes. Par conséquent, on peut résoudre le problème sur chaque ligne de X comme suit

$$X_{:i} \leftarrow \operatorname{argmin}_{X_{:i} \in \mathbb{R}^r} \left\| Y_{:i} - A X_{:i} \right\|_F^2 \text{ tel que } X_{ij} \geq 0 \forall j. \quad (7)$$

Par contre, les contraintes sur les lignes de A ne sont pas indépendantes. Par conséquent, quand X est fixé, il faut résoudre le problème en une fois sur l'ensemble de la matrice A . On a alors

$$A \leftarrow \operatorname{argmin}_{\{\mathbf{c}_j\}_{j=1}^r} \sum_{i=1}^n \left\| Y_{:i} - \sum_{j=1}^r r_{\boldsymbol{\tau}}(\mathbf{c}_j) X_{ij} \right\|_F^2. \quad (8)$$

Le problème de NMF est ensuite résolu de manière itérative en résolvant successivement (7) et (8).

Partition en 2r blocs (LS-2r_{rat}) Pour avoir $2r$ blocs, on considère les colonnes de A et X . Lorsqu'on considère toutes les variables comme fixes exceptée une colonne de X , $X_{:i}$, on peut optimiser analytiquement le problème en utilisant l'opérateur \max élément par élément [1] :

$$X_{:i} \leftarrow \max_{\mathbf{c}_i} \left(\frac{A_{:i}^\top Y - \sum_{j \neq i} A_{:i}^\top A_{:j} X_{:j}}{\|A_{:i}\|^2}, 0 \right), \quad (9)$$

Optimiser $A_{:i}$ en considérant que toutes les autres variables sont fixées n'est pas aussi facile, car la contrainte n'est pas séparable. On peut néanmoins trouver $A_{:i}$ en résolvant un problème de projection sur l'ensemble $\mathcal{R}_+(\boldsymbol{\tau})$ [3]

$$A_{:i} \leftarrow r_{\boldsymbol{\tau}} \left(\operatorname{argmin}_{\mathbf{c}_i} \left\| \frac{Y X_{:i} - \sum_{j \neq i} A_{:j} (X_{:j})^\top X_{:i}}{\|X_{:i}\|^2} - r_{\boldsymbol{\tau}}(\mathbf{c}_i) \right\|_F^2 \right).$$

La NMF est résolue de manière itérative en calculant (9) sur les r colonnes de X successivement et en résolvant le problème (10) sur les r colonnes de A successivement, puis en répétant ces $2r$ optimisations jusqu'à convergence.

Performances des méthodes

On résout la NMF avec des fonctions rationnelles en utilisant un algorithme de descente par bloc, et on compare les performances selon la partition utilisée. On compare

également à la NMF usuelle, sans contrainte sur les facteurs autre que leur nonnégativité, résolue avec $2r$ blocs (LS-2r_{vec}) [1]. Les données synthétiques utilisées pour nos tests sont obtenues à partir de fonctions rationnelles, il est donc théoriquement possible de les factoriser exactement.

Données de test Les données utilisées sont créées comme un produit de deux facteurs $Y = AX^\top$. La matrice $A \in \mathbb{R}^{200 \times 5}$, contient dans chacune de ses colonnes un élément de \mathcal{R}_+ , l'ensemble des fonctions rationnelles nonnégatives sur $[-1, 1]$, de degré (10, 10), évaluées sur 200 points équidistants sur $[-1, 1]$. Les colonnes de A sont la projection sur \mathcal{R}_+ d'un polynôme nonnégatif de degré 10 perturbé par une fonction rationnelle de degré (1,10). La matrice $X \in \mathbb{R}^{100 \times 5}$ est construite aléatoirement en suivant une distribution de Dirichlet de paramètre $\alpha = 1/5$. On ajoute parfois un bruit Gaussien N aux données avec un rapport signal sur bruit de 20dB, dans ce cas $Y = AX^\top + N$. Une fois les facteurs A^k, X^k obtenus à l'itération k , l'algorithme est arrêté si $s^k < 10^{-12}$ avec

$$s^k = \frac{\|Y - A^k X^k\|_F - \|Y - A^{k-1} X^{k-1}\|_F}{\|Y - A^k X^k\|_F}. \quad (10)$$

Pour éviter de trop long tests, le temps maximal alloué aux algorithmes est également limité. Cette limitation est fixée en fonction de la complexité de l'algorithme : les algorithmes avec $2r$ blocs résolvent de nombreux petits problèmes simples et sont limités à 200 secondes. Les algorithmes utilisant un ou deux bloc(s) doivent résoudre de plus gros problèmes avec plus de variables à chaque itération. Ils sont limités à 1000 secondes. On verra dans les expériences que ces limites de temps sont justifiées.

Pour évaluer la qualité de la factorisation, nous comparons aux facteurs originaux A et X , que l'algorithme ne connaît pas. On considère deux critères dont le résidu relatif $\|AX - A^k X^k\|_F / \|AX\|_F$ qui mesure la qualité de la reconstruction de la matrice de données Y . Le deuxième critère analyse la qualité des signaux retrouvés dans les colonnes de A . Comme ces signaux peuvent être normalisés et permutés sans perte de généralité, on trouve d'abord la meilleure permutation et normalisation de A^k par rapport à A, \tilde{A}^k , puis on calcule le rapport signal sur interférence (SIR) entre A et \tilde{A}^k . De plus, pour estimer au mieux le temps qui a été réellement nécessaire à un algorithme pour converger, on considère le temps pris pour atteindre la première itération k pour laquelle $\frac{s^k - s^n}{s^k} < 10^{-2} \forall n \geq k$.

Chaque méthode est testée sur dix initialisations différentes. Les graphes présentent en abscisse le résidu relatif et en ordonnée le SIR, les meilleurs points se trouvent donc en bas à gauche. La valeur moyenne de ces deux critères est présentée à l'aide d'un marqueur, et un trait est également tracé entre les valeurs extrêmes de chaque critère. La légende contient entre parenthèse le temps moyen nécessaire pour converger.

Comparaison des méthodes Si $L(J)$ est la complexité de l'optimiseur de moindres carré, avec J la taille de la Jacobienne du problème, et si I est le nombre d'itérations, on peut calculer la complexité de nos méthodes, voir Table 1. On s'attend donc à ce que l'ordre de rapidité (du plus rapide au plus lent) soit le suivant : LS-2r_{vec} ; LS-2r_{rat} ; LS-2_{rat} ; LS_{rat}. En pratique cet ordre est assez respecté, même si LS-2r_{rat} est parfois plus rapide que LS-2r_{vec}.

LS-2r _{vec}	LS-2r _{rat}	LS-2 _{rat}	LS _{rat}
$O(mnrI)$	$O(rL(md)I)$	$O(rL(rmdn)I)$	$O(L(rmn^2))$

TABLE 1 – Complexité des méthodes analysées. On suppose que $d \ll m, n$ et $O(mnrI) < O(rL(md)I)$.

LS-2r_{vec} a beaucoup plus de degrés de liberté que les méthodes utilisant des fonctions rationnelles, mais n'a pas d'a priori sur la forme des signaux à retrouver. Cela s'observe dans la Figure 2, puisque cette méthode obtient le meilleur résidu, mais un SIR souvent moins bon. En effet, LS-2r_{rat} et LS-2_{rat} peuvent obtenir, selon l'initialisation, un SIR jusqu'à deux fois meilleur. Comme il n'y a pas de bruit à filtrer, retrouver les signaux d'origine, et donc obtenir un bon SIR, est plus proche du but recherché, et on recommandera donc ces deux méthodes usant des fonctions rationnelles. Néanmoins, il y a une très grande variabilité des résultats lorsqu'on utilise des fonctions rationnelles et, en moyenne, utiliser la NMF classique avec des vecteurs est le plus intéressant. Enfin, on ne peut recommander LS_{rat} qui est la seule méthode qui n'arrive pas à retrouver les signaux originaux dans aucun des 10 tests réalisés, en plus d'être la méthode la plus lente.

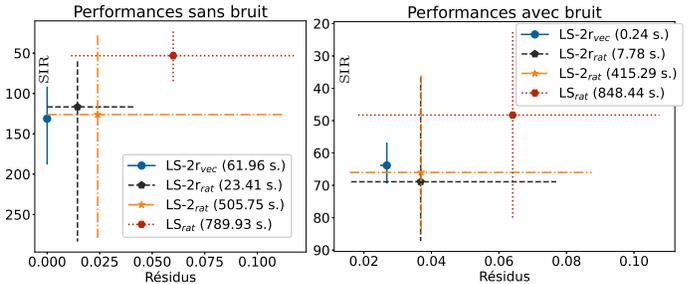


FIGURE 2 – Performances des algorithmes dans le cas non bruité (gauche) et dans le cas bruité (droite).

Dans le cas bruité, toutes les méthodes basées sur des fonctions rationnelles ont des résultats comparables dans le meilleur cas, mais ont toujours des résultats très variables. A cause du bruit, LS-2r_{vec} n'arrive plus à obtenir un résidu nul, et n'arrive pas à obtenir d'aussi bon résultats que les autres dans le meilleur des cas, mais reste une méthode intéressante en moyenne. Les méthodes utilisant des fonctions rationnelles peuvent donc obtenir de très bon résultats, avec un résidu faible et un SIR élevé, mais sont très dépendantes de leur initialisation. Nous allons examiner si associer ces méthodes lors de la résolution ne permet pas de diminuer ce problème.

Associer les différentes partitions Nous allons associer les méthodes par deux de la manière suivante : la première méthode est exécutée jusqu'à atteindre une tolérance $s^k < 10^{-2}$ (10). Ensuite, la seconde méthode est initialisée avec les facteurs obtenus par la première, et exécutée jusqu'à convergence. On utilise donc une partition pour en initialiser une autre. Ces méthodes hybrides sont nommées "méthode 1 - méthode 2".

On observe dans la Table 2 qu'associer les méthodes permet d'améliorer au moins 2 critères sur 3 (à savoir le temps, le résidu et le SIR) dans plus de 40% des cas, et même les méthodes les moins faciles à aider sont améliorées sur au moins deux critères dans plus de 30% des cas. Mieux, on voit dans les Tables 3 et 4 que les méthodes les plus précises sont toujours des méthodes hybrides. Associer deux méthodes peut donc aider à améliorer les performances de l'algorithme, et permet d'obtenir les meilleurs résultats en terme de résidu et de SIR. Par contre, il n'y a pas d'effet particulier de ces associations sur la variabilité des méthodes, ce qui est illustré dans la Figure 3 (les méthodes les plus précises peuvent également obtenir de mauvais résultats dans certains tests).

	Initialiser avec une autre méthode est meilleur pour au moins		
	1 critère	2 critères	3 critères
LS-2r _{vec}	60%	31%	5 %
LS-2r _{rat}	56%	37%	3 %
LS-2 _{rat}	76%	44%	17 %
LS _{rat}	86%	55%	41 %
Toutes méthodes	70%	42 %	15%

TABLE 2 – Pourcentage de cas où initialiser avec une autre méthode améliore un ou plusieurs des trois critères d'efficacité.

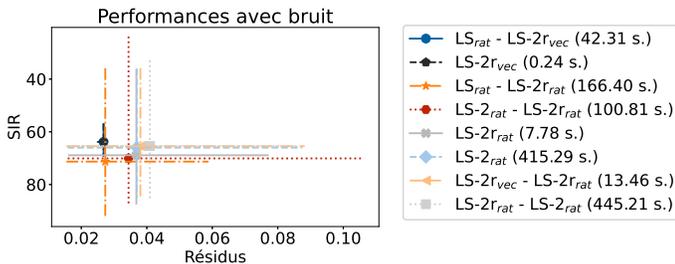


FIGURE 3 – Performances de quelques méthodes hybrides dans le cas bruité.

Il est également intéressant de noter que, dans le cas bruité, initialiser les méthodes avec LS_{rat} donne les résultats les plus précis (Table 4). Pourtant, dans la Figure 2 on a vu que cette méthode utilisée seule est la moins précise des quatre en moyenne. Cela est un peu contre-intuitif, puisqu'on pourrait supposer qu'il est plus intéressant d'initialiser avec une méthode rapide qui résout le problème localement, puis affiner avec la méthode plus lente qui prend en compte l'ensemble du problème, mais c'est ici l'inverse qui est le plus efficace dans le cas bruité. D'un autre côté

on observe que les méthodes hybrides terminant avec LS-2r_{vec} ou LS-2r_{rat} sont plus rapides que les autres, ce qui est assez logique. En effet, obtenir une tolérance de 10^{-2} est assez facile et ne nécessite que quelques itérations, la complexité est donc dominée par la deuxième méthode.

Meilleur	En moyenne	Parmi tous les tests
temps	LS-2r _{vec} - LS-2r _{rat} (20 sec)	LS-2r _{vec} - LS-2r _{rat} (2.57 sec)
résidu	LS-2r _{rat} - LS-2r _{vec} ($9.5 \cdot 10^{-6}$)	LS-2r _{rat} - LS-2r _{rat} ($1.9 \cdot 10^{-10}$)
SIR	LS-2r _{rat} - LS-2 _{rat} (162.66)	LS-2r _{rat} - LS-2 _{rat} (406.61)

TABLE 3 – Meilleures méthodes dans le cas non-bruité.

Meilleur	En moyenne	Parmi tous les tests
temps	LS-2r _{vec} (0.24 sec)	LS-2r _{vec} (0.05 sec)
résidu	LS _{rat} - LS-2r _{vec} (0.027)	LS _{rat} - LS-2 _{rat} (0.015)
SIR	LS _{rat} - LS-2r _{rat} (71.3)	LS _{rat} - LS-2r _{rat} (92.51)

TABLE 4 – Meilleures méthodes dans le cas bruité.

Dans les tests effectués, la méthode la plus précise dans le cas non-bruité est l'association LS-2r_{rat} - LS-2_{rat}, et dans le cas bruité LS_{rat} - LS-2_{rat} (cette méthode obtient un résidu proche de LS_{rat} - LS-2r_{vec}, mais un SIR bien plus élevé). Il est important de garder en tête que les méthodes utilisant des fonctions rationnelles obtiennent des résultats très variables selon le problème à résoudre, et même selon l'initialisation, et que ces méthodes ne seront donc pas forcément les meilleures dans d'autres situations. Une conclusion assez robuste demeure : associer les différentes partitions dans une méthode hybride permet souvent d'améliorer les performances de l'algorithme.

Références

- [1] A. Cichocki, R. Zdunek, and S.-i. Amari. Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization. In *International Conference on Independent Component Analysis and Signal Separation*, pages 169–176. Springer, 2007.
- [2] O. Debals, M. Van Barel, and L. De Lathauwer. Nonnegative matrix factorization using nonnegative polynomial approximations. *IEEE Signal Processing Letters*, 24(7) :948–952, 2017.
- [3] C. Hautecoeur, F. Glineur, and L. De Lathauwer. Hierarchical alternating nonlinear least squares for nonnegative matrix factorization using rational functions. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 1045–1049. IEEE, 2021.
- [4] V. Powers and B. Reznick. Polynomials that are positive on an interval. *Transactions of the American Mathematical Society*, 352(10) :4677–4692, 2000.