UMONS
Université de Mons

Faculté Polytechnique

Tri-factorisation symétrique et positive de matrices et détection de communautés

Travail de Fin d'Études
présenté en vue de l'obtention du grade de
Master Ingénieur civil en Informatique et Gestion

Tom VANZEVEREN

POLYTECH MONS

Sous la direction de :
Professeur Nicolas GILLIS (promoteur)
Professeur Arnaud VANDAELE (co-promoteur)

Août 2023

pôle hainuyer

Remerciements

Je voudrais remercier vivement le promoteur de ce travail, le Professeur Nicolas Gillis, pour sa disponibilité, ses conseils avisés et son soutien dans la réalisation de ce travail. Je remercie également monsieur le Professeur Arnaud Vandaele, co-promoteur

Finalement, je tiens à remercier mon entourage proche pour leur soutien durant ces cinq années d'études, conclues par ce travail.

Résumé

L'analyse des réseaux complexes et la détection de communautés sont des défis cruciaux dans de nombreux domaines, de la sociologie à l'analyse de textes en passant par l'informatique. Dans ce contexte, la factorisation positive de matrices (*NMF*) et plus particulièrement la factorisation symétrique positive de matrices (*symNMF*) se placent comme des approches très efficaces pour extraire des structures cachées et significatives à partir de données de réseau. SymNMF exploite la symétrie des matrices d'adjacence pour capturer des interactions complexes entre les nœuds d'un réseau.

Ce travail de fin d'étude propose un nouveau modèle de factorisation positive pour la détection de communautés : la tri-factorisation symétrique positive hors diagonale - *ODtri-symNMF*. Il est la rencontre de la tri-facotrisation symétrique positive (*tri-symNMF*) et de la factorisation symétrique positive hors diagonale (*ODsymNMF*). Le modèle décrit parvient à pointer les liens cachés entre les nœuds d'un réseau pour les classer en communautés. Mais il est également conçu pour renseigner les interactions entre ces communautés. Après une définition mathématique du modèle et de ceux qui l'ont inspiré, nous présentons un algorithme basé sur la descente de coordonnées. Il est décliné en deux versions et une méthode d'initialisation supplémentaire à l'initialisation aléatoire classique est développée.

Pour évaluer l'efficacité de notre algorithme, nous l'appliquons à divers ensembles de données synthétiques d'abord et issues de jeux de données réelles provenant de domaines tels que la psychiatrie ou la sociologie ensuite. De cette manière, nous pouvons apprécier la qualité de l'approximation obtenue ainsi que la qualité de détection de communautés et des liens qui les caractérisent. Les résultats obtenus sont critiqués pour exposer les forces et faiblesses du modèle. Ils sont encourageants et soulignent la pertinence de notre modèle pour identifier la structure des réseaux.
Finalement, de nombreuses perspectives d'évolution de ce projet sont présentées.

mots clés : [Factorisation non-négative, Matrice, Détection de communautés, Analyse de réseaux, Factorisation symétrique]

Table des matières

Chapitre 1

Introduction

La factorisation positive de matrices *(Nonnegative Matrix Factorization - NMF)* est une méthode de détection de communauté très rependue.
La détection de communautés est une problématique fondamentale dans l'analyse de réseaux complexes. Ces réseaux, qu'ils soient sociaux ou de toute autre nature, présentent souvent une structure sous-jacente composée de groupes d'entités interconnectées, appelées communautés. La compréhension de ces structures intrinsèques est cruciale pour de nombreuses applications, telles la recherche dans les sciences sociales, la recommandation de contenu, l'analyse de textes, *etc.* Tout phénomène modélisable sous forme de matrice est un domaine d'application potentiel de la NMF.

La factorisation symétrique positive de matrices *(Symmetric Nonnegative Matrix Factorization - symNMF)* est une des nombreuses variantes de la NMF. Elle émerge comme une approche puissante pour la détection de communautés dans les réseaux. symNMF est particulièrement adaptée aux réseaux non dirigés, car elle prend en compte la symétrie des interactions entre les nœuds.

L'objectif de ce travail de fin d'étude est de développer et présenter un modèle de factorisation positive : la tri-factorisation symétrique positive hors diagonale de matrices *(Off-diagonal Symmetric Nonnegative Matrix Trifactorization - ODtri-symNMF).* Ce modèle est la combinaison de deux variantes de la symNMF : la tri-factorisation symétrique positive de matrices *(Symmetric Nonnegative Matrix Trifactorization - tri-symNMF)* et la factorisation symétrique positive hors diagonale de matrices *(Off-diagonal Symmetric Nonnegative Matrix Factorization - ODsymNMF).*

La structure de ce travail est la suivante : la prochaine section passe en revue les modèles de factorisation positives qui mènent au modèle ODtri-symNMF. La section suivante abordera la modélisation mathématique du problème ainsi que son implémentation informatique. Ensuite, les résultats de tests menés sur l'algorithme à l'aide de données synthétiques et réelles seront présentés et critiqués. Finalement, nous revenons sur l'objectif initial en guise de conclusion et discutons de perspectives de ce modèle.

Le code et les données utilisées sont disponibles sur github : https ://github.com/tom-vze/TFE-ODtri-symNMF/tree/main

Chapitre 2

Modèles de factorisations non-négatives de matrices

2.1 Factorisation non-négative de matrices

La factorisation non-négative de matrices, *Nonnegative Matrix Factorisation - NMF* en anglais, est une technique de décomposition de matrices largement utilisée en analyse de données. La NMF présente l'avantage d'imposer des contraintes de non-négativité, ce qui la rend utile lorsque les données ne peuvent pas être naturellement représentées par des valeurs négatives. De plus, elle peut fournir une représentation plus interprétable des données, car les facteurs extraits sont généralement plus facilement compréhensibles que ceux obtenus par d'autres méthodes de décomposition de matrices.

Étant donné une matrice non-négative $X \in \mathbb{R}_+^{m\times n}$, un rang de factorisation r et une distance $D(.,.)$ entre deux matrices, la NMF consiste à calculer deux matrices non-négatives $W \in \mathbb{R}_+^{m\times r}$ et $H \in \mathbb{R}_+^{r\times n}$ minimisant $D(X, WH)$.

$$\min_{W\in\mathbb{R}_+^{m\times r}, H\in\mathbb{R}_+^{r\times n}} D(X, WH) \tag{2.1}$$

Remarquons que le terme "approximation non-négative" de matrices serait plus approprié étant donné que nous ne cherchons pas l'égalité $WH = X$ mais une approximation. Pour la suite de ce travail, nous allons cependant nous en tenir à la dénomination usuelle de factorisation non-négative de matrices.

La distance,

$$D : \mathbb{R}_+^{m\times n} \times \mathbb{R}_+^{m\times n} \mapsto \mathbb{R}_+ \text{ tel que } (A, B) \mapsto D(A, B), \tag{2.2}$$

est la fonction modélisant l'objectif du problème et est également la mesure de l'erreur. Le choix de cette distance a une grande influence sur la solution trouvée.

Applications

L'extraction de connaissance, fouille de textes ou encore, en anglais, *text mining*, est une approche qui permet d'explorer et d'analyser de grandes quantités de données textuelles afin d'en extraire des informations exploitables.
Considérons la matrice X comme un ensemble de documents. Chaque colonne est une représentation des mots présents dans un même document. Les techniques de construction de cette matrice varient, mais le principe reste identique.

$$X(:,j) \approx \sum_{k=1}^{r} W(:,k)H(k,j) \tag{2.3}$$

On observe que les colonnes de W peuvent être considérées comme des vecteurs de mots, au même titre que le sont les colonnes de X. Plus précisément, chaque colonne regroupe les mots utilisés pour parler de sujets spécifiques. Les colonnes de H représentent l'importance de ces différents sujets au sein d'un document. Cette modélisation est illustrée par la figure 2.1.

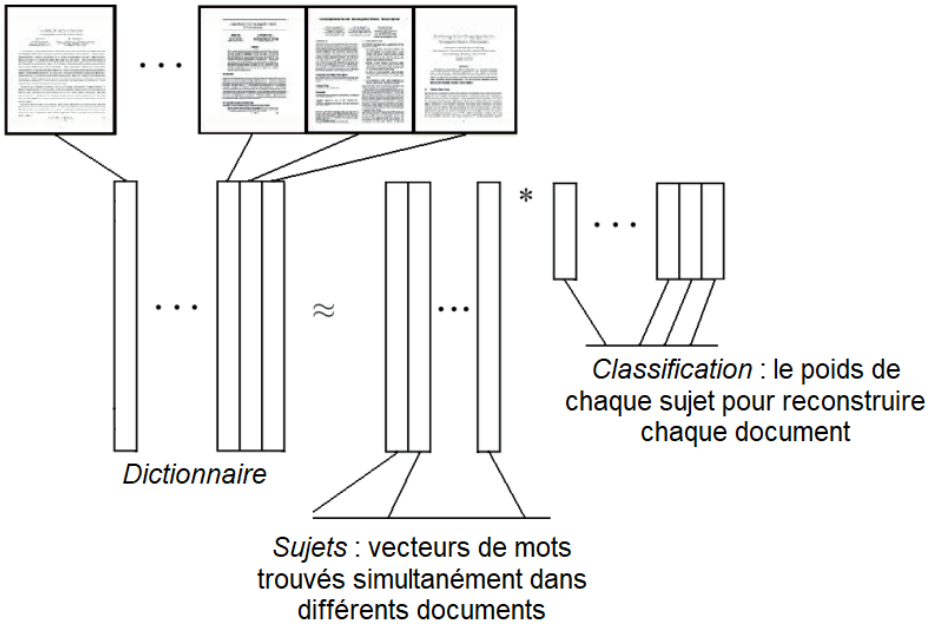


FIGURE 2.1 – Illustration de la NMF pour l'extraction de documents. Illustration tirée du livre *Nonnegative Matrix Factorization* [3].

Cette application illustre bien le sens concret que l'on peut donner aux matrices W et H. En effet, comme les entrées sont toutes positives, aucune annulation n'est possible dans la reconstruction de X.

De manière plus générale, W représente une base de caractéristiques et H représente les poids attribués à ces caractéristiques pour reconstruire X.

Nous pouvons également citer d'autres applications répandues :

— L'extraction de caractéristiques dans un ensemble d'images.
Prenons l'exemple d'une base de données de visages. Les colonnes de W sont alors des caractéristiques telles que les yeux, le nez ou la moustache. Les colonnes de H indiquent quelles caractéristiques sont présentes ou non sur le visage original.

— Démixage aveugle d'images hyperspectrales.
Une image hyperspectrale mesure l'intensité de la lumière dans une scène pour de nombreuses longueurs d'onde différentes. On parle de démixage aveugle, car il est effectué sans connaissance préalable des matériaux ou des sources présentes dans l'image. La matrice W représente les signatures spectrales des différents matériaux présents dans l'image, tandis que la matrice H contient les proportions ou les contributions de chaque matériau dans chaque pixel de l'image.

— Séparation des sources sonores.
Pour cette application, les colonnes de X représentent la signature fréquentielle de certains sons, généralement des notes de musique. Tandis que les colonnes de H indiquent quand ces notes sont actives.

Nous renvoyons les lecteurs intéressés vers le livre *Nonnegative Matrix Factorization* [3] et l'article [9].

2.2 Factorisation non-négative symétrique de matrices

La factorisation non-négative symétrique de matrices, *symNMF*, est une variante de la factorisation non-négative classique qui impose $W = H^T$. L'équation (2.3) modélisant le problème devient alors :

$$X \approx WW^T = \sum_{k=1}^{r} W(:,k)W(:,k)^T, \tag{2.4}$$

qui n'est possible que si X est une matrice symétrique.

Le problème d'optimisation (2.1) prend le plus souvent la forme suivante :

$$\min_{W \geq 0} \quad \|X - WW^T\|_F^2. \tag{2.5}$$

Une telle matrice X peut être considérée comme la matrice d'adjacence d'un graphe. C'est-à-dire que l'entrée $X(i,j) = X(j,i)$, si elle est binaire, représente la présence ou non d'un lien entre le nœud i et le nœud j du graphe. Si les valeurs ne sont pas binaires, elles représentent le poids de ce lien. L'enjeu est d'identifier des sous-ensembles de nœuds fortement liés, que l'on appelle généralement communautés.
La factorisation non-négative symétrique décompose X en une somme de matrices $W(:,k)W(:,k)^T$ de rang 1. Les entrées non-négatives de ces matrices forment des sous-matrices carrées qui caractérisent les connections dans des sous-groupes de nœuds du graphe initial. Autrement dit, ces matrices mettent en évidence les communautés.

Illustration

Le jeu de données *Zachary's karate club* est souvent utilisé pour illustrer des concepts et des analyses dans le domaine de la théorie des réseaux sociaux, tels que la détection de communautés, la centralité des nœuds, la résilience des réseaux, *etc.* Il a été largement référencé dans la littérature et est devenu un exemple classique pour comprendre les structures et les interactions sociales dans les réseaux.

Il a été recueilli par Wayne Zachary en 1977 lors d'une étude sur les dynamiques sociales au sein d'un club de karaté universitaire.

Le jeu de données décrit les relations sociales entre les 34 membres du club de karaté, en enregistrant les interactions observées et les liens d'amitié entre eux. Chaque membre est représenté par un nœud du réseau et les relations sont représentées par les arêtes entre les nœuds. L'arête entre deux nœuds indique une relation amicale ou une interaction entre les deux membres correspondants. On compte 78 liens entre les 34 membres.
Suite à un conflit interne, le groupe s'est divisé en deux. Près de la moitié des membres est restée avec l'administrateur tandis que l'autre moitié a suivi l'instructeur dans la création d'un nouveau club.

Si l'on considère X, la matrice d'adjacence du graphe décrit ci-dessus, en précisant que l'on a choisi $X(i,i) = 1$ pour $i = 1, 2, ..., 34$, on peut lui appliquer symNMF avec un rang de 2 pour tenter d'identifier les deux communautés. Cette décomposition est visible sur la figure 2.2.
En associant le nœud j au groupe 1 si $W(j,1) > W(j,2)$ et au groupe 2 sinon, on obtient la communauté de l'instructeur (nœud 1) :

$$K_I = \{1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 17, 18, 20, 22\}$$

et celle de l'administrateur (nœud 34) :

$$K_A = \{9, 10, 15, 16, 19, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34\}.$$

Certains nœuds semblent avoir une appartenance claire à une communauté ou l'autre. On a par exemple $W(4,1) = 0.86405 > W(4,2) = 0$ pour le nœud 4, ou évidement $d, W(1,1) = 1.37 > W(1,2) = 0.04$ pour l'instructeur et $W(34,1) = 0.08 < W(34,2) = 1.38$ pour l'administrateur. D'autres sont plus centraux, on a $W(9,1) = 0.32$ et $W(9,2) = 0.54$ pour le nœud 9. Il s'agit d'ailleurs du seul membre mal classé par la factorisation.

Nous renvoyons les lecteurs intéressés vers la section 5.4.7 du livre *Non-negative Matrix Factorization* [3] et le chapitre 7 de [6]

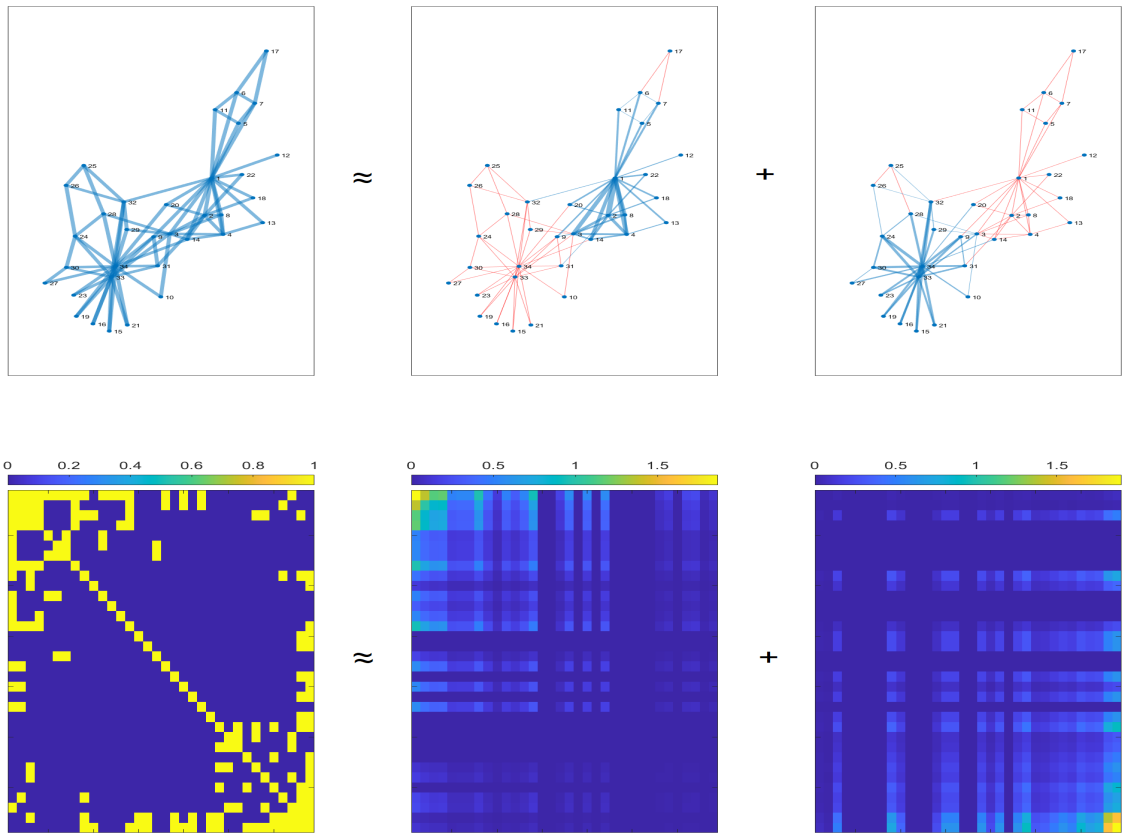


FIGURE 2.2 – Illustration de la symNMF sur le jeu de données *Zachary's karate club*. En haut : les graphes dont les poids des arêtes sont attribués par les matrices correspondantes. L'épaisseur de l'arête est proportionnelle à son poids, si ce dernier est inférieur à 0.1, l'arête est rouge. En bas et de gauche à droite : X, $W(:,1)W(:,1)^T$ et $W(:,2)W(:,2)^T$. Illustration réalisée avec le code de [3].

2.3 Factorisation non-négative symétrique hors diagonale de matrices

Off-diagonal symmetric nonnegative matrix factorization (ODsymNMF) est une variante récente de la symNMF qui ne prend pas en compte la diagonale de la matrice d'entrée. Ce modèle est proposé par Moutier, 2021 [7].

On fait évoluer l'expression (2.4) en

$$X_{ij} \approx (WW^T)_{ij} \quad \text{pour } i \neq j \tag{2.6}$$

et le problème d'optimisation (2.5) devient

$$\min_{W \geq 0} \quad \|X - WW^T\|_{OD,p}, \tag{2.7}$$

où

$$\|X - WW^T\|_{OD,p} = \left(\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left| X - WW^T \right|_{ij}^{p} \right)^{1/p}. \tag{2.8}$$

Différents algorithmes ont été développés pour les cas $p = 1$ et $p = 2$.
Il y a trois avantages principaux à la factorisation non-négative symétrique hors diagonale.

Le plus intuitif est que la diagonale d'une matrice symétrique représente la relation entre un élément et lui-même. C'est une information qui n'a pas beaucoup d'importance et qui détériore la capacité des algorithmes à correctement détecter les communautés.
Ensuite, ce modèle est théoriquement plus robuste, car il existe toujours une factorisation exacte de taille maximale $n(n-1)/2$ où n est la taille de la matrice d'entrée. Ce qui n'est pas le cas pour la factorisation non-négative symétrique.
Le troisième avantage est que le problème d'optimisation devient beaucoup plus simple à traiter. La technique la plus utilisée dans la factorisation de matrice et la descente de coordonnées. Avec la factorisation non-négative symétrique, le sous problème sur une variable est quartique alors qu'avec la factorisation hors diagonale, ce sous problème devient quadratique. De plus, dans le premier cas, la convergence n'est pas garantie étant donné que le minimum d'une fonction quartique n'est pas forcément unique. Tandis que dans le second cas, la convergence est garantie.

Jusque-là, deux techniques d'initialisation de la matrice W étaient utilisées : une initialisation aléatoire ou une matrice nulle de la bonne dimension. En parallèle à la factorisation hors diagonale, Moutier propose également une initialisation gourmande. Il a observé que, pour un même nombre d'itérations, le coût en temps de calcul de cette initialisation est presque tout le temps plus faible que les algorithmes de factorisation qu'il a développés. Cette différence augmente avec le nombre d'itérations.

Résultats

Cette nouvelle approche a été testée dans un premier temps sur des données synthétiques afin de pouvoir mesurer précisément les performances des algorithmes.
Il en ressort que l'initialisation gourmande est meilleure que les deux précédentes méthodes d'initialisation. Sur des données binaires, l'algorithme résolvant le problème (2.7) pour $p = 2$ et l'algorithme de factorisation nonnégative symétrique classique ont des performances similaires, mais sont dépassés par l'algorithme résolvant le problème (2.7) pour $p = 1$.

Ensuite, les algorithmes ont été testés sur des documents réels.
Comme sur les données synthétiques, la factorisation hors diagonale avec $p = 2$ et la factorisation symétrique sont proches. Il est important de rappeler que la factorisation hors diagonale est moins coûteuse en temps de calcul et procure une garantie de convergence. Contrairement aux données synthétiques toujours binaires, la factorisation hors diagonale avec $p = 1$ est tantôt plus efficace, tantôt moins efficace que les deux autres méthodes. Cela est dû au fait que les exemples réels ne sont pas binaires.

2.4 Trifactorisation non-négative symétrique de matrices

Commençons par la trifactorisation non-négative. Cette variante consiste, comme son nom l'indique, à approximer la matrice initiale X par trois matrices W, S et H comme suit :

$$X \approx WSH = \sum_{k=1}^{r_1} \sum_{j=1}^{r_2} W(:,k)S(k,j)H(j,:). \tag{2.9}$$

Considérons que les colonnes de la matrice X sont des films, ses lignes des utilisateurs et que $X(i,j) = 1$ si l'utilisateur i a visionné le film j. Les colonnes de W identifient r_1 sous-groupes de films regardés ensemble, les colonnes de H identifient r_2 sous-groupes d'utilisateurs dont le comportement est similaire et $S(k,j) > 0$ indique que le sous-groupe j d'utilisateurs regardent les films du sous-groupe k.

Intéressons-nous désormais à la trifactorisation non-négative symétrique.

$$X(i,j) \approx W(i,:)SW(j,:)^T = \sum_{k=1}^{r} \sum_{p=1}^{r} W(i,k)S(k,p)W(j,p) \tag{2.10}$$

$X(i,j)$ est toujours une matrice d'adjacence à un graphe. On a $W(i,k)$ représentant l'appartenance de i à la communauté k et $S(k,p)$ l'intensité de la connexion entre les communautés k et p. Par exemple, si la communauté k est "politique" et la communauté p "économie", on s'attend à une valeur $S(k,p)$ élevée, car ces deux sujets sont fortement liés.
Ce modèle va plus loin que les précédents, car il identifie les communautés, mais il donne également des informations sur leurs interactions. De plus, le résultat est facilement compréhensible grâce au sens physique évident des matrices W et S.

Nous renvoyons les lecteurs intéressés vers les sections 5.4.8 et 5.4.9 du livre *Nonnegative Matrix Factorization* [3] et les articles [10] et [8].

Illustration

Pour illustrer l'intérêt de la trifactorisation, nous pouvons faire évoluer l'exemple du club de karaté évoqué au point 2.2.

Si l'on reprend la matrice W_{symNMF} calculée avec la factorisation symétrique positive, on trouve que la matrice S qui minimise $\|X - W_{symNMF}SW_{symNMF}^T\|_F$ est égale à la matrice identité. Ce résultat était prévisible étant donné que $(W_{symNMF}W_{symNMF}^T)$ est déjà très proche de X sans l'intervention de S.

Cependant, l'algorithme développé lors de ce projet cherche à minimiser une distance différente qui correspond à l'équation (2.8) adaptée au cas de la tri-factorisation symétrique. On observe qu'une matrice S_{opti} qui minimise cette distance est très proche de la matrice identité, pour la même raison qu'évoquée précédemment.

$$S_{opti} = \begin{bmatrix} 0.9935 & 0 \\ 0 & 1.1034 \end{bmatrix}$$

L'erreur [1], calculée avec W_{symNMF} et S_{opti} vaut 94.303. Cette solution est un minimum local du problème. En effet, en repartant de celle-ci, avec l'algorithme de ODtri-symNMF développé dans ce projet, on peut trouver une nouvelle solution dont l'erreur est de 61.361.

Il n'est pas avisé de comparer les performances de la symNMF et de la ODtri-symNMF sur base de cette expérience car les deux méthodes n'ont pas le même objectif. La première minimise une norme de Frobenius et la seconde minimise la norme $\ell 1$ entre chaque élément de deux matrices. De plus, la seconde ne tient pas compte de la diagonale contrairement à la première.

Si la comparaison des performances n'a pas lieu d'être, cette section a pour objectif de montrer l'intérêt de la trifactorisation par le prisme de sa matrice S. Allons plus loin dans notre exemple.

On modifie la matrice W_{symNMF} pour ne garder que la valeur maximale de chaque ligne. Grâce à cela, nous avons deux communautés disjointes. Voici la matrice S_{opti}, une solution de ce nouveau problème.

$$S_{opti} = \begin{bmatrix} 1.0094 & 0.1596 \\ 0.1596 & 1.0079 \end{bmatrix}$$

On observe que les deux communautés sont fortement liées avec elles-mêmes (éléments diagonaux) mais on voit également apparaître le faible lien qui lie les deux communautés entre elles. L'erreur est aussi légèrement plus faible. La figure 2.3 illustre cela graphiquement. Les deux communautés sont les mêmes que celles identifiées par la symNMF mais elles sont bien séparées. On le voit sur les deux matrices $W(:,1)W(:,1)^T$et $W(:,2)W(:,2)^T$: les entrées non nulles ne se superposent pas, une personne ne peut être dans les deux communautés en même temps. La matrice S informe de la connexion entre ces deux communautés disjointes.

1. *cfr.* section 3.3 pour plus de détails sur le calcul de l'erreur

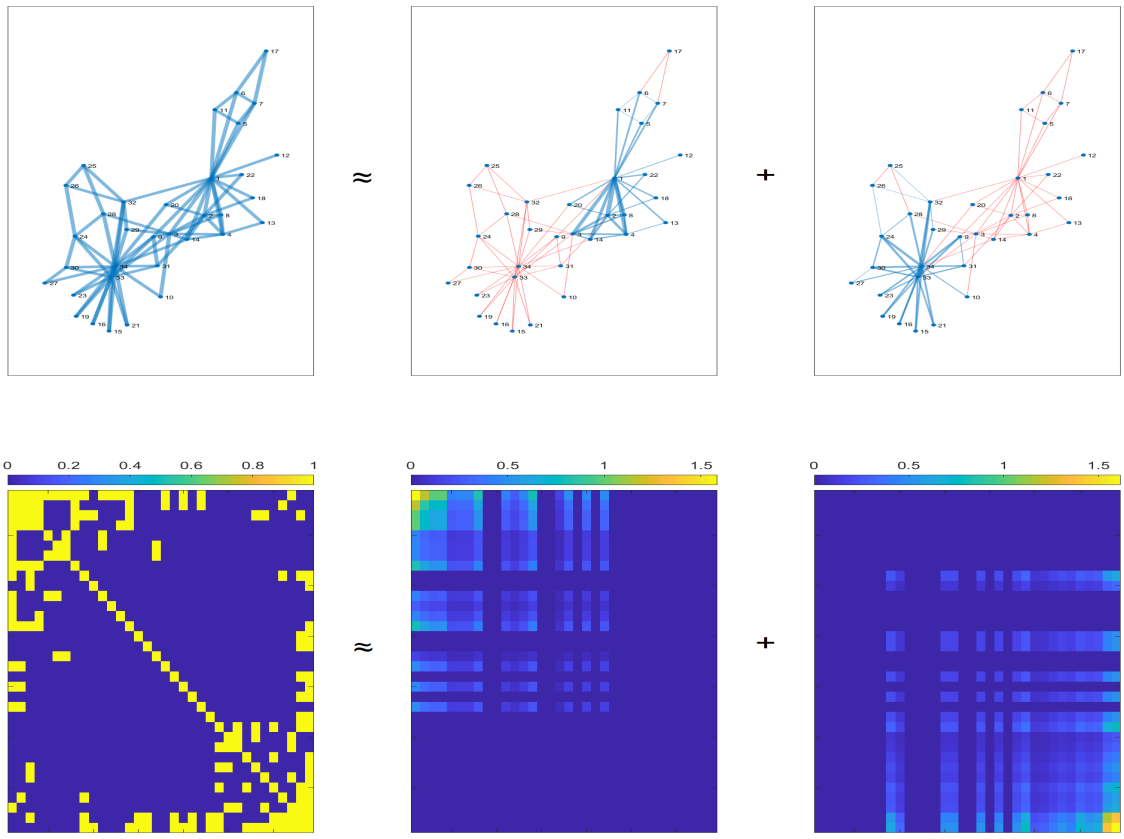


FIGURE 2.3 – Illustration de la ODtri-symNMF sur le jeu de données *Zachary's karate club*. En haut : les graphes dont les poids des arêtes sont attribués par les matrices correspondantes. L'épaisseur de l'arête est proportionnelle à son poids, si ce dernier est inférieur à 0.1, l'arête est rouge. En bas et de gauche à droite : X, $W(:,1)W(:,1)^T$ et $W(:,2)W(:,2)^T$. Illustration réalisée avec le code de [3].

Chapitre 3

Tri-factorisation non-négative symétrique hors diagonale de matrices

3.1 Motivation

L'objectif de ce modèle et de combiner les avantages des modèles tri-symNMF et ODsymNMF. On veut l'introduction d'une matrice S renseignant les connexions entre les communautés dans une modèle qui ne tient pas compte de la diagonale de la matrice X qui n'apporte généralement pas d'information utile.

3.2 Définition

Nous reprenons l'approximation (2.10) à la différence que nous ne considérons pas les éléments de la diagonale.
Pour $i \neq j$,

$$X(i,j) \approx W(i,:)SW(j,:)^T = \sum_{k=1}^{r}\sum_{p=1}^{r} W(i,k)S(k,p)W(j,p). \tag{3.1}$$

Le problème de minimisation qui en découle est le suivant :

$$\min_{W,S\geq 0} \quad \|X - WSW^T\|_{OD,1}, \tag{3.2}$$

où

$$X - WSW^T\|_{OD,1} = \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} \left| X(i,j) - \sum_{k=1}^{r}\sum_{p=1}^{r} W(i,k)S(k,p)W(j,p) \right|. \tag{3.3}$$

3.3 Implémentation

Le problème (3.2) cherche à minimiser deux variables, W et S. Une première version de l'algorithme met à jour W puis S. Dans une seconde version, ces deux étapes sont inversées afin de permettre à S d'évoluer plus significativement par rapport à sa forme initiale. En effet, dans la première version, l'optimisation complète de W avant de toucher à S rend les mises à jour de ses valeurs très faibles. La matrice renvoyée est souvent une matrice diagonale, n'apportant aucune information sur l'interaction entre les communautés.
Une troisième version intermédiaire, qui alternait la mise à jour d'un élément de W et d'un élément de S a rapidement été abandonnée, car l'erreur était beaucoup plus importante que celles des autres versions. Et les matrices W et S obtenues n'étaient pas significativement meilleures que celles rénvoyées par la seconde version de l'algorithme.

Nous allons utiliser la technique de descente de coordonnées. Cette technique est très connue et intuitive. À chaque itération, on cherche à optimiser une seule variable de manière exacte ou inexacte en fonction de la difficulté du problème. Toutes les autres variables sont fixées.

L'erreur est la norme $\ell 1$ appliquée à chaque composant des matrices. Elle est calculée comme la somme des valeurs absolues des différences entre chaque entrée de X et son équivalent dans (WSW^T), à l'exception des éléments de la diagonale.
Dans le code et pour la suite de ce rapport, l'erreur relative sera préférée à l'erreur absolue :

$$\text{erreur relative} = \frac{\|X - WSW^T\|_{OD,1}}{\|X\|_{OD,1}} \tag{3.4}$$

Autrement dit, on divise l'erreur par la somme des éléments de X hors diagonale. Cela nous permet d'avoir une valeur entre 0 et 1 qui sera plus facilement interprétable et facilitera la comparaison sur des données différentes.

L'algorithme s'arrête
- lorsque le nombre maximum d'itérations est atteint ;
- lorsque l'erreur est nulle. Une solution optimale est trouvée ;
- lorsque la progression entre deux itérations est trop faible. L'algorithme ne trouve plus de meilleure solution.

Avant de passer à l'itéré suivant, W est mis à l'échelle pour aider l'algorithme à converger vers une solution optimale [1].

1. *cfr.* section 3.3.4

L'algorithme 1 présente la structure globale de l'algorithme. Pour sa seconde version, les lignes 4 et 5 sont inversées.

Algorithm 1 ODtri-symNMF

Input: $X \in \mathbb{R}^{n\times n}$, $W_0 \in \mathbb{R}_+^{n\times r}$, $S_0 \in \mathbb{R}_+^{r\times r}$
Output: $W \in \mathbb{R}_+^{n\times r}$, $S \in \mathbb{R}_+^{r\times r}$

1: $W \leftarrow W_0$
2: $S \leftarrow S_0$
3: **for** $iter = 1 : maxiter$ **do**
4: $W \leftarrow UpdateW$ see Algortihm 2
5: $S \leftarrow UpdateS$ see Algorithm 3
6: $error \leftarrow \frac{\|X-WSW^T\|_{OD,1}}{\|X\|_{OD,1}}$
7: **if** $error = 0$ **then**
break
8: **else if** $Progression <<$ **then**
break
9: **end if**
10: $\alpha \leftarrow ComputeAlpha$
11: $W \leftarrow W * \sqrt{\alpha}$
12: **end for**

3.3.1 Mise à jour de W

L'algorithme 2 illustre comment la mise à jour de W s'effectue.
Reprenons le problème (3.2) et considérons la mise à jour de l'entrée $W(l,q)$. Nous avons donc $i = l$ et $k = q$. L'expression (3.3) devient :

$$\sum_{\substack{j=1\\ j\neq l}}^{n} \left| X(l,j) - \sum_{k=1}^{r}\sum_{p=1}^{r} W(l,k)S(k,p)W(j,p) \right|. \tag{3.5}$$

En posant :

$$a_j = X(l,j) - \sum_{\substack{k=1\\ k\neq p}}^{r}\sum_{p=1}^{r} W(l,k)S(k,p)W(j,p) \tag{3.6}$$

et

$$b_j = \sum_{p=1}^{r} S(q,p)W(j,p), \tag{3.7}$$

on peut réécrire l'expression (3.5) en

$$\sum_{\substack{j=1\\ j\neq l}}^{n} |a_j - W(l,q)b_j| \,. \tag{3.8}$$

Minimiser cette somme est un problème de médiane pondérée contrainte détaillé au point 3.3.3.

Concrètement, c'est ici que la technique de la descente de coordonnée intervient. L'algorithme parcourt la matrice W. À chaque $W(i,k)$, toutes les autres entrées sont fixées et on calcule les vecteurs a_j et b_j qui serviront à mettre la valeur de $W(i,k)$ à jour.

Algorithm 2 Update W

for $i = 1 : n$ **do**
  for $k = 1 : r$ **do**
    for $j = 1 : n$, $j \neq i$ **do**
      for $k_{bis} = 1 : r$, $k_{bis} \neq k$ **do**
        $temp \leftarrow W(i, k_{bis})S(k_{bis}, :)W(j, :)$
      end for
      $a_j \leftarrow X(i,j) - temp$
      $b_j \leftarrow S(k, :)W(j, :)$
    end for
    $W(i,k) = weigthedMedian(a, b)$ see Algorithm 4
  end for
end for

3.3.2 Mise à jour de S

À l'image de la mise à jour de W, nous reprenons le problème (3.2) en considérant l'entrée $S(y,z)$.
On pose

$$a_{ij} = X(i,j) - \sum_{\substack{k=1 \\ k\neq y}}^{r} W(i,k)S(k,z)W(j,z) - \sum_{\substack{p=1 \\ p\neq z}}^{r} W(i,y)S(y,p)W(j,p) - \sum_{\substack{k=1 \\ k\neq y}}^{r}\sum_{\substack{p=1 \\ p\neq z}}^{r} W(i,k)S(k,p)W(j,p) \tag{3.9}$$

et

$$b_{ij} = W(i,y)W(j,z). \tag{3.10}$$

On peut reformuler le problème comme

$$\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} \left| a_{ij} - S(y,z)b_{ij} \right|. \tag{3.11}$$

Encore une fois, pour minimiser cette valeur, nous utilisons la méthode des médianes pondérées contraintes décrite au point 3.3.3.

Le principe est le même que lors de la mise à jour de W. À la seule différence que nous savons que S est symétrique. En ligne 17 de l'algorithme 3, nous mettons à jour $S(k,p)$ et $S(p,k)$ en même temps. Cependant, l'algorithme parcourt toute la matrice S. Ce qui signifie qu'en dehors des éléments de la diagonale, à chaque itération, toutes les entrées de S seront mises à jour à deux reprises.

Algorithm 3 Update S

1: **for** $k = 1 : r$ **do**
2: **for** $p = 1 : r$ **do**
3: **for** $i = 1 : n$ **do**
4: **for** $j = 1 : n$, $j \neq i$ **do**
5: **for** $k_{bis} = 1 : r$ **do**
6: **for** $p_{bis} = 1 : r$ **do**
7: **if** $k_{bis} \neq k$ *or* $p_{bis} \neq p$ **then**
8: $temp \leftarrow W(i, k_{bis})S(k_{bis}, p_{bis})W(j, p_{bis})$
9: **end if**
10: **end for**
11: **end for**
12: $a_j \leftarrow X(i,j) - temp$
13: $b_j \leftarrow W(i,k)W(j,p)$
14: **end for**
15: **end for**
16: $S(k,p) = weigthedMedian(a,b)$ see Algorithm 4
17: $S(p,k) = S(k,p)$
18: **end for**
19: **end for**

3.3.3 Problème de médiane pondérée contrainte

Il existe un algorithme qui résout le problème de médiane pondérée en $\mathcal{O}(n)$ [5]. Dans notre cas, les expressions (3.8) et (3.11) sont convexes. Si la solution optimale est négative, on la remplace par 0.
L'algorithme 4 est identique à celui proposé dans [7] en inversant a et b. Ce dernier est initialement tiré de [4]. Il tourne en $\mathcal{O}(nlog(n))$.

Algorithm 4 Weighted median

Input: $a \in \mathbb{R}^d$, $b \in \mathbb{R}_+^d$
Output: $x \in \mathbb{R}$

$S \leftarrow \emptyset$
for i = 1 :d **do**
  if $b_i \neq 0$ **then**
    $S \leftarrow S \cup \left\{ \frac{a_i}{b_i} \right\}$
  end if
end for
$[S, Inds] \leftarrow sort(S)$
$b \leftarrow \frac{b(Inds)}{sum(a)}$
$i \leftarrow 1$
$CumulatedSum \leftarrow 0$
while $CumulatedSum < 0.5$ **do**
  $CumulatedSum \leftarrow CumulatedSum + b_i$
  $x \leftarrow S_i$
  $i \leftarrow i + 1$
end while

3.3.4 Mise à l'échelle

Pour éviter à l'algorithme de chercher des solutions trop lointaines d'un minimum, on met W à l'échelle à la fin de chaque itération et lors de l'initialisation.
Cette mise à l'échelle consiste à trouver le α optimal pour le problème suivant

$$\min_{\alpha} \quad |a - \alpha b| \tag{3.12}$$

où a et b sont respectivement les matrices X et (WSW^T) mises sous forme de vecteurs de dimension $(1, n^2)$. Il s'agit là encore d'un problème de médiane pondérée.
Finalement, on multiplie W par la racine carrée de *alpha* pour adapter ses valeurs aux valeurs de X.

3.3.5 Initialisation

S est initialisée comme la matrice identité de taille $(r \times r)$. $W(n \times r)$ est soit générée de manière aléatoire avec des réels positifs compris entre 0 et 1. Soit générée avec la méthode de détection de communauté *kmeans*. Une recherche de communautés est lancée sur la matrice X. Chaque centre obtenu grâce à *kmeans* est associé à une colonne de W. Cela permet de se rapprocher de la solution du problème.
Notons que ces deux méthodes reposent sur de l'aléatoire. Les matrices initiales ne seront donc pas identiques pour deux essais avec tous les autres paramètres inchangés.
Avant de lancer l'algorithme, la matrice W générée par l'une des deux méthodes est mise à l'échelle une première fois. Elle le sera ensuite à la fin de chaque itération.

Chapitre 4

Expériences numériques

4.1 Données synthétiques

Pour confirmer le bon fonctionnement de l'algorithme et afin d'étudier son comportement, il a été testé sur des données synthétiques. Des matrices d'adjacence ont été générées aléatoirement et quatre paramètres ont varié : la taille des matrices d'adjacence, le rang de factorisation, le type des données initiales et la méthode d'initialisation des matrices W.

4.1.1 Convergence

Afin de s'assurer du bon fonctionnement de l'algorithme, dans ses deux versions, l'évolution de l'erreur relative est observée. Ces analyses ont été faites

- sur des matrices de tailles (5×5), (20×20) et (35×35) ;
- pour des rangs de factorisation de 2, 3, 4 et 5 ;
- sur des données binaires et réelles, comprises entre 0 et 1 ;
- avec une initialisation de la matrice W aléatoire et via *kmeans*,
- avec un maximum de cinquante itérations.

Un échantillon de ces tests est présenté en figure 4.1, le reste des résultats se trouve en annexe A.

On constate d'abord que l'erreur décroit rapidement avant de se stabiliser. Précisons que l'algorithme s'arrête lorsque la différence d'erreur relative entre deux itérations est trop faible. Dès lors, le vecteur qui stocke cette information est rempli avec la dernière erreur calculée. C'est pourquoi les courbes s'arrêtent toutes à cinquante itérations, bien que dans la majorité des cas, l'algorithme n'opère pas plus de la moitié de ces itérations.
D'ailleurs, l'évolution la plus importante de l'erreur se fait dans les 5 à 10 premières itérations. On a donc une diminution rapide qui se stabilise vite, mais dans beaucoup de cas, pas à un niveau souhaitable.

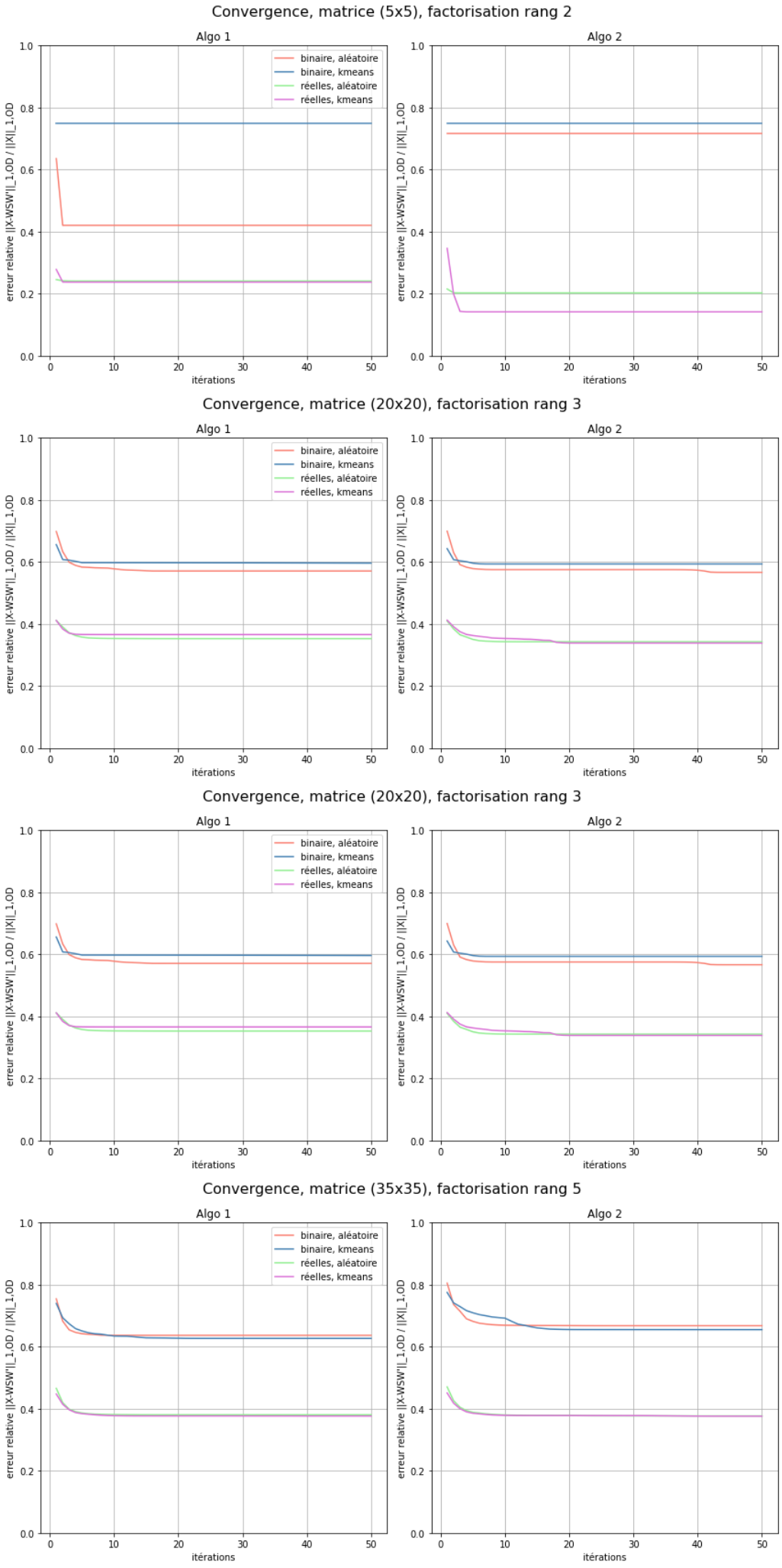


FIGURE 4.1 – Évolution de l'erreur relative.

On remarque que la diminution, dans les premières itérations, est plus lente pour les problèmes de plus grande dimension. C'est logique, car il y a de plus de paramètres à mettre à jour et cela demande donc plus d'essais avant d'arriver à une solution globale satisfaisante. Si l'on observe le cas extrême de la factorisation de rang 2, on voit une descente très brusque et même des droites horizontales.

Ce phénomène peut être dû au fait que l'algorithme trouve une solution très proche de la solution finale dès la première itération.
Rappelons qu'avant le calcul de la première erreur, la matrice W a subi, dans certains cas, une initialisation via *kmeans*, une mise à l'échelle et une mise à jour de toutes ses entrées. La matrice S a subi une double mise à jour de ses entrées non diagonales [1] et une mise à jour simple de sa diagonale. Sur un problème de petite taille, on peut imaginer que cela suffise à rapprocher très fortement la solution d'un minimum.

Précisons que l'algorithme ne s'arrête jamais avant 3 itérations. Cela signifie que même quand la courbe semble complètement plate, il y a en réalité une très légère amélioration entre la première et la deuxième itération.

Nous avons expliqué pourquoi ces courbes plates pouvaient être observées sur des problèmes de petite taille. On observe pourtant le problème pour la matrice (35×35) et le rang 4. De plus, l'erreur est particulièrement élevée. Une explication à cette anomalie sera avancée dans la section qui suit.

On constate finalement que l'erreur sur les données binaires est significativement plus importante que celle sur les données réelles positives. Alors que pour les mêmes données, la différence entre les méthodes d'initialisation est moins marquée. Ces constats seront développés dans la suite de ce travail.

1. Expliqué au point 3.3.2

4.1.2 Comparaison des versions de l'algorithme

Comme exposé au chapitre précédent, il existe deux versions de l'algorithme de trifactorisation hors diagonale qui ne diffèrent que par l'ordre de traitement de W et S. Le premier met à jour W puis S, le second fait l'inverse.

Ces deux versions ont été testées

— sur 12 matrices de tailles allant de (6×6) à (39×39) ;
— pour des rangs de factorisation de 2, 3, 4 et 5 ;
— sur des données binaires et réelles, comprises entre 0 et 1 ;
— avec une initialisation de la matrice W aléatoire,
— avec un maximum de cinquante itérations.

La figure 4.2 montre l'erreur relative finale en fonction de la taille de la matrice à factoriser pour le rang 2 et 5. Le rang 3 et 4 se trouvent en annexe B. Les observations et analyses qui suivent sont valables pour les quatre rangs.

L'erreur relative résiduelle a tendance à augmenter avec n. Effectivement, l'algorithme doit gérer plus de paramètres.
On peut observer que pour les données réelles, les deux versions de l'algorithme sont très proches. Tandis que sur les données binaires, la première version semble meilleure.

Néanmoins, la seconde version avait pour but de prioriser la mise à jour de S afin de le rendre plus dense et l'éloigner de sa forme initiale, la matrice identité. Bien qu'elle soit légèrement moins performante, cette deuxième version remplit son rôle sur les deux types de données. Pour les données binaires, on compte 6 matrices diagonales avec l'algorithme 1 et aucune avec l'algorithme 2. Pour les données réelles, c'est 1 matrice purement diagonale contre 0. Cependant, les valeurs hors diagonale sont souvent très faibles avec les données réelles.

S'il est évident qu'une matrice S diagonale n'est pas souhaitable, il reste délicat de juger de la capacité des matrices W et S à identifier les communautés et leurs relations sur des données synthétiques aléatoires. Cette section a pour but de comparer les algorithmes d'un point de vue quantitatif via l'erreur relative. Une analyse plus qualitative des matrices obtenues sera faite plus tard, lorsque des jeux données réelles seront utilisés.

Comme supposé au point précédant, l'erreur sur les données binaires est significativement plus importante que sur les données réelles, indépendamment de l'algorithme utilisé. On remarque également des irrégularités très marquées sur les courbes des données binaires. Là où les courbes des données réelles sont relativement plus lisses.

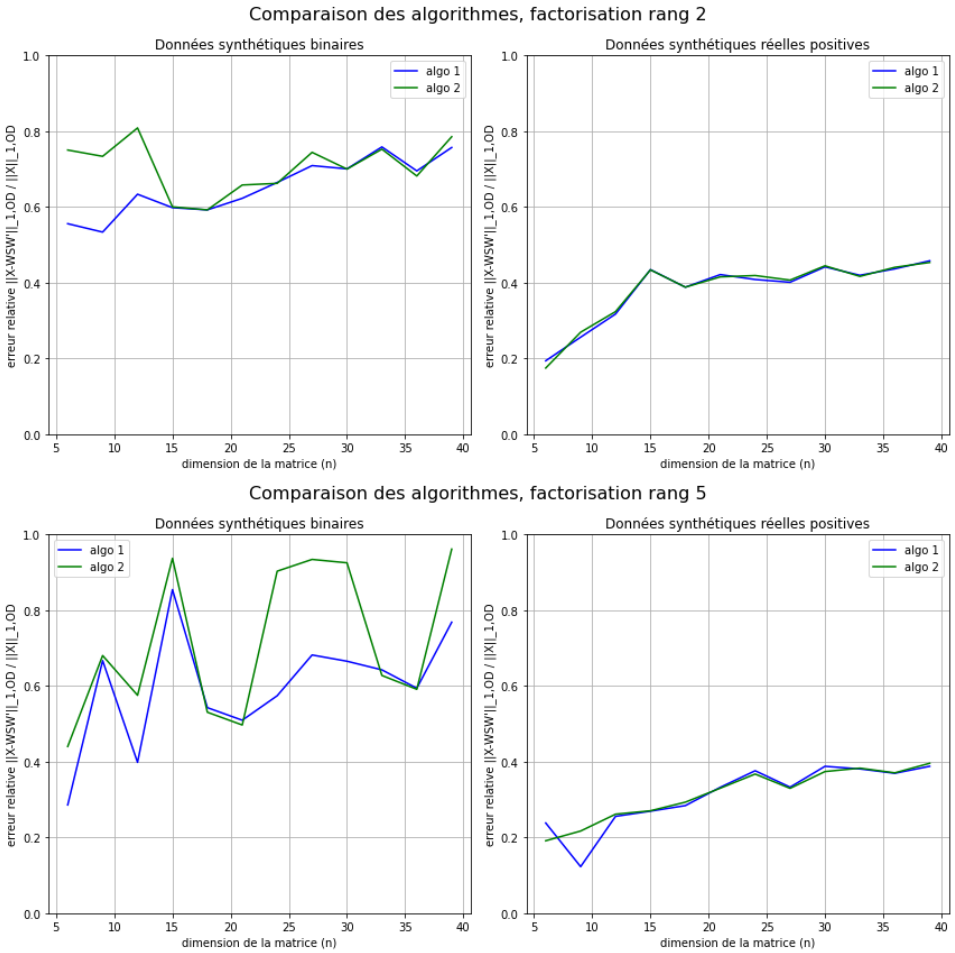


FIGURE 4.2 – Comparaisons des algorithmes sur des données synthétiques.

Nous formulons l'hypothèse que cela est dû à la nature même des données.

Dans la méthode des médianes pondérées [2], le x optimal est une valeur du vecteur $\frac{a}{b}$. Ce qui signifie qu'au plus a contient d'entrées nulles, au plus il y a de chances que la valeur retournée soit égale à 0.

Comme cette méthode est utilisée durant le processus de mise à l'échelle avec a égal à la matrice X sous forme de vecteur, si cette dernière est creuse, il y a de grandes chances d'obtenir un $\alpha = 0$.

2. Pour rappel : méthode qui trouve le x optimal pour $min|a - bx|$

En multipliant la matrice W initiale par un α nul, on lance l'algorithme avec un W nul et un $S = I$. La première version de l'algorithme peut fonctionner dans certains cas, car elle commence par mettre à jour W mais la seconde version doit commencer par optimiser S avec un W nul.

La fonction des médianes pondérées ne fonctionne pas dans les cas ou le vecteur b qui lui est donné. Effectivement, cela revient à vouloir minimiser une constante.
Dès lors, il est prévu de renvoyer un 1 pour ne pas accentuer le phénomène en renvoyant un 0. Mais cela a pour conséquence de remplir la matrice S de 1 et de complètement perturber la recherche.

L'algorithme est donc sensible aux matrices creuses, ce qui est plus souvent le cas avec des matrices binaires. Comme expliqué, sa deuxième version est plus sensible que la première. C'est pourquoi on observe des plus grandes irrégularités sur les données binaires et des plus gros pics avec l'algorithme 2 par comparaison au premier. Pour vérifier cette hypothèse, la figure 4.3 [3] met en évidence les cas où le α était nul. On voit que cela correspond souvent aux pics des courbes.

Remarquons que nous avons évoqué le problème lors de la mise à l'échelle à l'initialisation, mais celui-ci peut se produire à n'importe quelle itération.

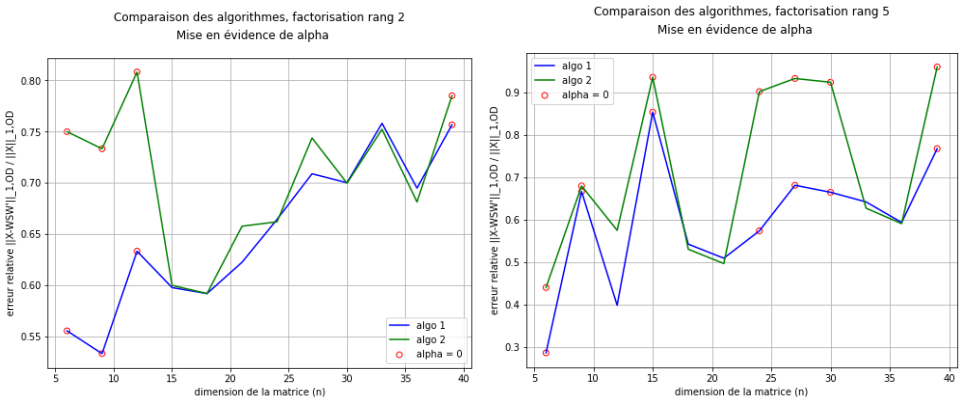


FIGURE 4.3 – Mise en évidence des $\alpha = 0$.

3. *cfr.* annexe B pour le rang 3 et 4.

4.1.3 Comparaison des initialisations

Pour juger de l'efficacité de l'utilisation de *kmeans* dans l'initialisation de la matrice W, les deux versions de l'algorithme ont encore été testées sur des données synthétiques.
D'une part, on procède à une initialisation aléatoire, comme au point précédent, et d'autre part, on initialise W avec les centres des groupes trouvés par *kmeans*.
Les tests ont été réalisés

- sur 12 matrices de tailles allant de (6×6) à (39×39) ;
- pour des rangs de factorisation de 2, 3, 4 et 5 ;
- sur des données binaires et réelles, comprises entre 0 et 1 ;
- avec une initialisation de la matrice W aléatoire et avec *kmeans*,
- avec un maximum de cinquante itérations.

Les résultats pour les rangs 2 et 5 sont représentés sur la figure 4.4 et les résultats pour les rangs 3 et 4 se trouvent en annexe C.

De manière générale, l'initialisation avec *kmeans* est plus efficace. La différence est plus marquée pour les données binaires que pour les données réelles. On observe toujours des pics sur les courbes des données binaires. Aussi bien avec *kmeans* qu'avec la génération aléatoire. Mais les courbes de *kmeans* semblent moins affectées. En analysant les résultats, on observe 15 α nuls pour l'initialisation aléatoire contre 3 α nuls pour l'initialisation *kmeans* sur les 48 matrices traitées. On peut imaginer que le fait d'être plus proche de la solution avec une initialisation mieux adaptée nous empêche de nous diriger vers une solution insignifiante et de nous retrouver bloqués comme c'est souvent le cas avec l'initialisation aléatoire.

Conclusion sur les données synthétiques

De ces trois premières expériences, on déduit que le premier algorithme, qui met à jour W avant S, surpasse le deuxième, qui fait l'inverse. De plus, on constate que l'initialisation *kmeans* est plus performante que l'initialisation aléatoire. On remarque aussi que les données binaires sont plus difficilement appréhendées par les algorithmes.
Mais au-delà de la minimisation de l'erreur, il est aussi important d'obtenir des matrices W et S pertinentes, qui identifient les communautés et leurs relations. Pour vérifier cela, nous devons travailler avec des données réelles pour lesquelles nous connaissons, au moins partiellement, la réponse attendue.

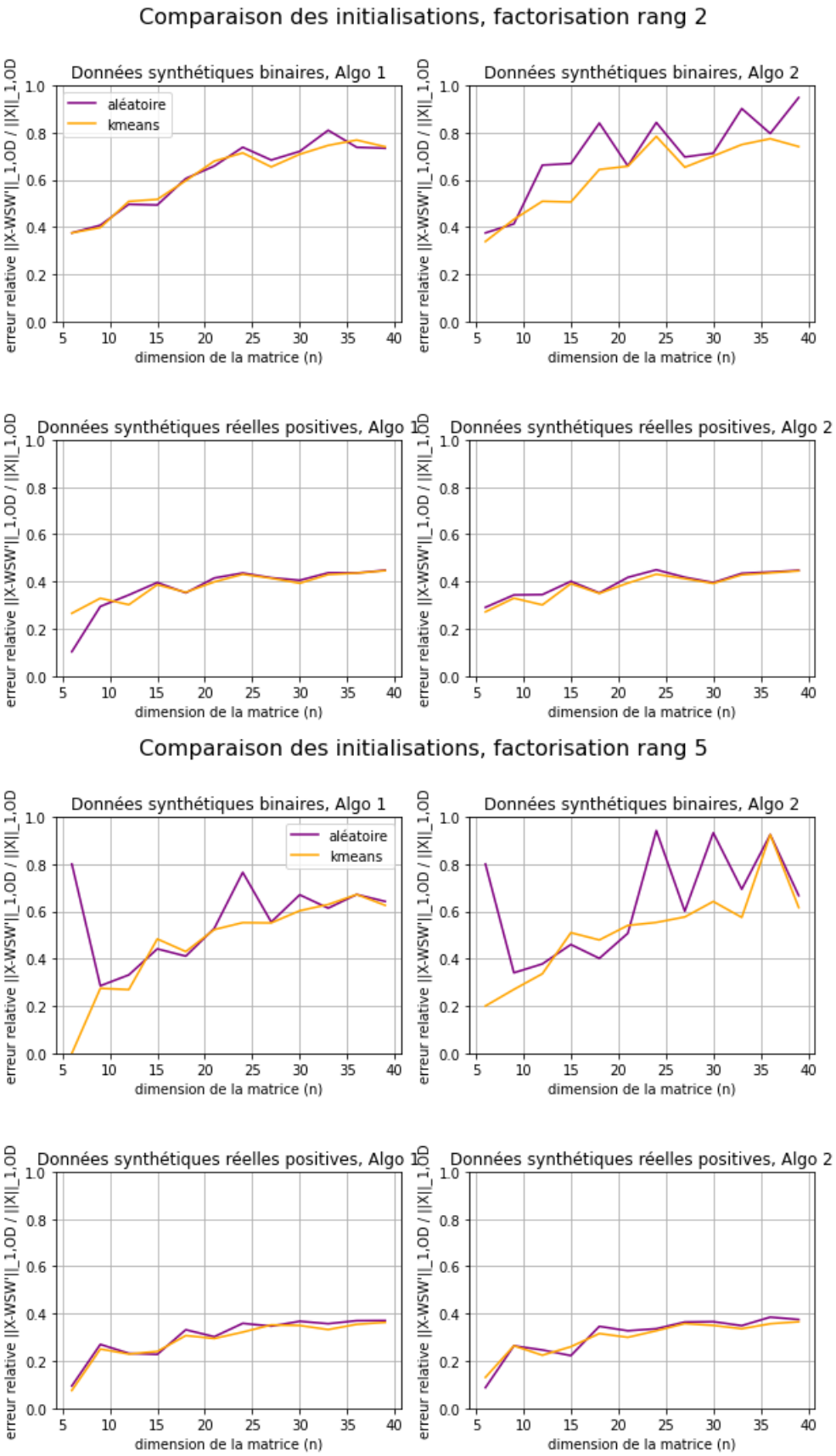


FIGURE 4.4 – Comparaison de méthodes d'initialisation sur des données synthétiques.

4.2 Données psychiatriques

Dans cette section, il est question de données psychiatriques. Une matrice $Y \in \mathbb{R}^{m \times n}$ contient les évaluations de m sujets sur n symptômes sur une échelle ordinale. La matrice $X \in \mathbb{R}^{n \times n}$, construite à partir de Y est composée des corrélations partielles entre les symptômes. Consulter [1] et [2] pour plus d'informations sur le calcul de X.
En considérant X comme la matrice d'adjacence d'un graphe dont chaque nœud est un symptôme, il devient pertinent d'identifier les groupes de nœuds ayant des liens étroits. Cela pourrait nous aider, par exemple, à évaluer quels symptômes seront affectés par une intervention ciblée sur un symptôme spécifique. Ce graphe est représenté sur la figure 4.5

Nous travaillons ici avec les données de 359 femmes souffrant de trouble de stress post-traumatique (*post-traumatical stress disorder - PTSD*, en anglais). Ces symptômes sont au nombre de 17 et sont tirés d'une classification standard des troubles mentaux utilisée par les professionnels de la santé mentale appelée DSM. Dans sa quatrième version [4], cette classification regroupe les symptômes en trois groupes :

— symptôme d'excitation, être nerveux par exemple,
— symptôme d'évitement, éviter les rappels du traumatisme par exemple,
— revivre le traumatisme, faire de mauvais rêves à ce propos par exemple.

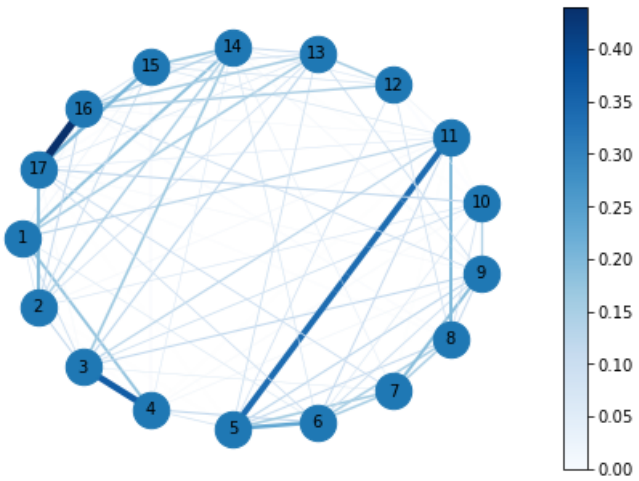


FIGURE 4.5 – Graphe de la matrice d'adjacence X.

4. la dernière version est la cinquième, mais nous nous référons à [1]

4.2.1 Analyse de l'erreur

Nos deux méthodes d'initialisation reposent sur un processus aléatoire. Ce qui signifie qu'elles ne donneront pas systématiquement le même résultat. Pour trouver la meilleure association, nous les avons utilisées cent fois sur base de la même matrice d'adjacence X décrite plus tôt, avec les deux algorithmes et un nombre maximum de 100 itérations. Voici l'erreur relative résiduelle moyenne relevée dans les quatre cas :

	Algorithme 1	**Algorithme 2**
Kmeans	0.591	0.618
Aléatoire	0.597	0.62

TABLE 4.1 – Erreur relative résiduelle moyenne sur les données psychiatriques.

Selon l'erreur relative moyenne et pour cette matrice, le classement de la meilleure combinaison est le suivant :

1. algorithme 1 avec initialisation *kmeans* ;
2. algorithme 1 avec initialisation aléatoire ;
3. algorithme 2 avec initialisation *kmeans* ;
4. algorithme 2 avec initialisation aléatoire.

Les valeurs sont proches. Pour la meilleure combinaison, on a une erreur de l'ordre de presque 60%. Ce qui est important.

Si l'on se réfère aux expériences menées sur les données synthétiques, dans le cas de données réelles, avec un rang 3 et une matrice 17×17, on devrait s'attendre à une erreur relative se situant entre 0.3 et 0.4 pour les quatre cas. Cependant, les données synthétiques étaient moins creuses que la matrice X considérée ici. On sait que les matrices creuses affectent les performances de l'algorithme (*cfr.* point 4.1.2).

On compte 115 zéros sur 289 entrées. Sans la diagonale qui est nulle, on a 98 zéros sur 272 entrées, ce qui représente 36%. Cela pourrait expliquer les mauvaises performances de l'algorithme qui se rapprochent plus des performances sur des données binaires pour les mêmes paramètres.

4.2.2 Analyse des communautés

Il est aussi important d'évaluer la qualité de W et S. Pour ce faire, nous allons lancer une fois chaque algorithme avec chacune des deux initialisations et étudier la meilleure combinaison pour identifier les groupes de symptômes.

Voici une classification proposée dans [1] qui nous servira de référence :
- symptôme d'excitation : $(5, 6, 7, 8, 9, 10, 11)$;
- symptôme d'évitement : $(1, 3, 4, 14)$;
- revivre le traumatisme : $(2, 10, 12, 13, 14, 15, 16, 17)$.

Les symptômes 10 et 14 ont la spécificité d'être présents dans deux groupes.

Après une première analyse des résultats, il apparait que S est diagonale dans la majorité des cas. Cela ne donne donc aucune information sur les relations entre les communautés. On perd l'intérêt de la tri-factorisation.
Après plusieurs essais, on peut néanmoins déjà constater que l'algorithme 2 est le plus enclin à construire une matrice S non diagonale, particulièrement avec l'initialisation *kmeans*. Rappelons que c'est précisement pour régler ce problème que cette version a été conçue. Malheureusement, les valeurs en dehors de la diagonale restent trop faibles et trop rares pour être interprétées.

Par conséquent, nous adaptons alors l'initialisation des matrices.
Elles ont été traitées pour ne garder que la valeur maximale sur chaque ligne. Cela a pour effet de proposer à l'algorithme des communautés disjointes, un symptôme n'appartient qu'à une communauté, celle pour laquelle la valeur de W était maximale, les autres valeurs ont été supprimées. Et l'algorithme est contraint à compléter S. C'est déjà cette technique qui avait été utilisée pour l'illustration du point 2.2.

On obtient, dans certains cas, une matrice S plus intéressante. Le calcul de l'erreur moyenne sur 100 essais montre que cette dernière ne semble pas trop affectée :

	Algorithme 1	**Algorithme 2**
Kmeans	0.590 -0.001	0.610 -0.008
Aléatoire	0.63 +0.033	0.677 +0.057

TABLE 4.2 – Erreur relative résiduelle moyenne sur les données psychiatriques, nouvelle initialisation.

On observe une très légère amélioration pour les initialisations *kmeans* et une détérioration de l'ordre de 3.3% et 5.7% pour les initialisations aléatoires. Effectivement, dans le premier cas, on renforce une classification normalement correcte et dans le second, on renforce une classification aléatoire potentiellement très éloignée de la bonne classification. Ce qui peut expliquer l'évolution de l'erreur.

Avec cette évolution de l'erreur relative résiduelle moyenne, le milieu du classement s'inverse et l'algorithme 2 avec *kmeans* se montre plus performant que l'algorithme 1 avec initialisation aléatoire.

Les graphes qui suivent ont été construits avec chacune des colonnes de W. Ils mettent en évidence chacune des trois communautés identifiées par l'algorithme. Pour établir la classification, on se réfère au plus grand des trois éléments de chaque ligne de W. Par exemple, si pour la ligne 5, c'est la valeur de la deuxième colonne qui est la plus élevée, on range l'élément 5 dans le groupe 2.

Cette méthode n'est pas optimale, notamment pour les symptômes présents dans plusieurs groupes. Car notre méthode empêche cette double appartenance.
Par exemple, si pour la même ligne 5, la valeur de la première colonne était très proche de la valeur maximale de la deuxième colonne, on aurait pu raisonnablement ranger l'élément 5 dans les groupes 1 et non le 2.

Pour chaque cas, nous tentons de rattacher les groupes proposés par l'algorithme à la classification mentionnée plus tôt. Si l'élément 10 ou 14 manque dans un groupe, cela ne sera pas considéré comme une erreur étant donné l'impossibilité de l'intégrer dans les deux groupes auxquels ils appartiennent. Seuls les éléments mal placés sont pointés en rouge.

Algorithme 1 avec initialisation *kmeans*
erreur relative : 0.64

$$W = \begin{bmatrix} 0.01 & 0.74 & 0.04 \\ 0.0 & 0.06 & 0.24 \\ 0.42 & 0.09 & 0.02 \\ 0.86 & 0.21 & 0.0 \\ 0.07 & 0.0 & 0.0 \\ 0.1 & 0.06 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.01 \\ 0.03 & 0.0 & 0.02 \\ 0.02 & 0.0 & 0.0 \\ 0.0 & 0.14 & 0.06 \\ 0.0 & 0.0 & 0.23 \\ 0.0 & 0.17 & 0.24 \\ 0.0 & 0.22 & 0.2 \\ 0.02 & 0.02 & 0.26 \\ 0.0 & 0.0 & 0.55 \\ 0.0 & 0.0 & 0.74 \end{bmatrix} \quad S = \begin{bmatrix} 0.95 & 0.0 & 0.0 \\ 0.0 & 0.95 & 0.0 \\ 0.0 & 0.0 & 0.96 \end{bmatrix}$$

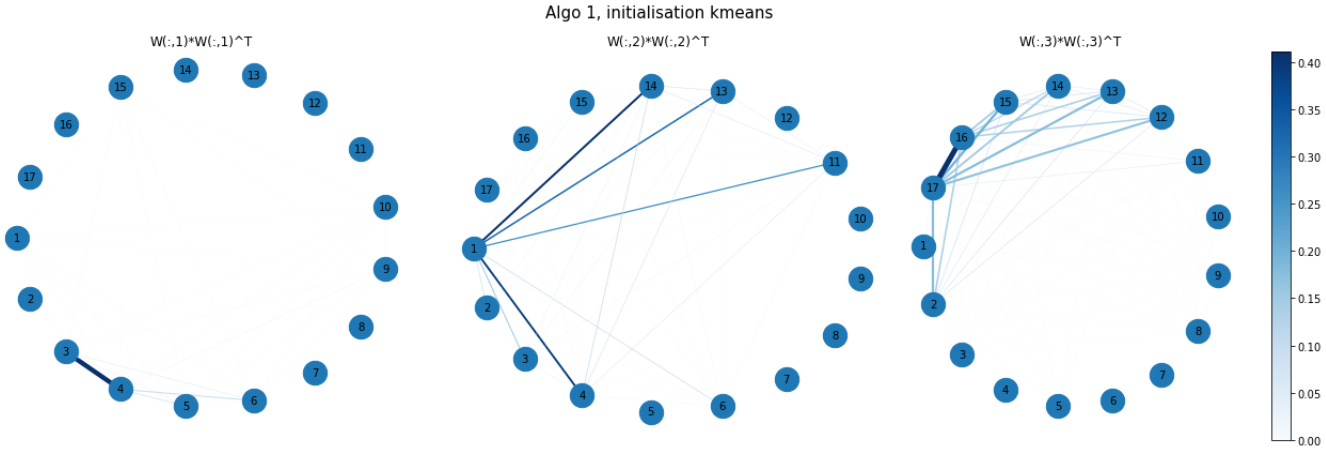


FIGURE 4.6 – Graphes reconstruits à partir des colonnes de W.

(3, 4, 5, 6, 7, 9, 10)	(1, 11, 14)	(2, 8, 12, 13, 15, 16, 17)
Excitation	**Évitement**	**Revivre le traumatisme**
4 éléments mal placés		

TABLE 4.3 – Classification des données psychiatriques algorithme 1 - *kmeans*.

Algorithme 1 avec initialisation aléatoire
erreur : 0.68

$$W = \begin{bmatrix} 0.22 & 0.34 & 0.0 \\ 0.0 & 0.14 & 0.53 \\ 0.5 & 0.0 & 0.16 \\ 0.72 & 0.0 & 0.01 \\ 0.08 & 0.0 & 0.0 \\ 0.13 & 0.0 & 0.0 \\ 0.02 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.03 & 0.04 & 0.0 \\ 0.02 & 0.0 & 0.09 \\ 0.19 & 0.06 & 0.02 \\ 0.0 & 0.11 & 0.0 \\ 0.0 & 0.37 & 0.02 \\ 0.06 & 0.43 & 0.11 \\ 0.0 & 0.24 & 0.07 \\ 0.0 & 0.23 & 0.1 \\ 0.0 & 0.11 & 0.3 \end{bmatrix} \quad S = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

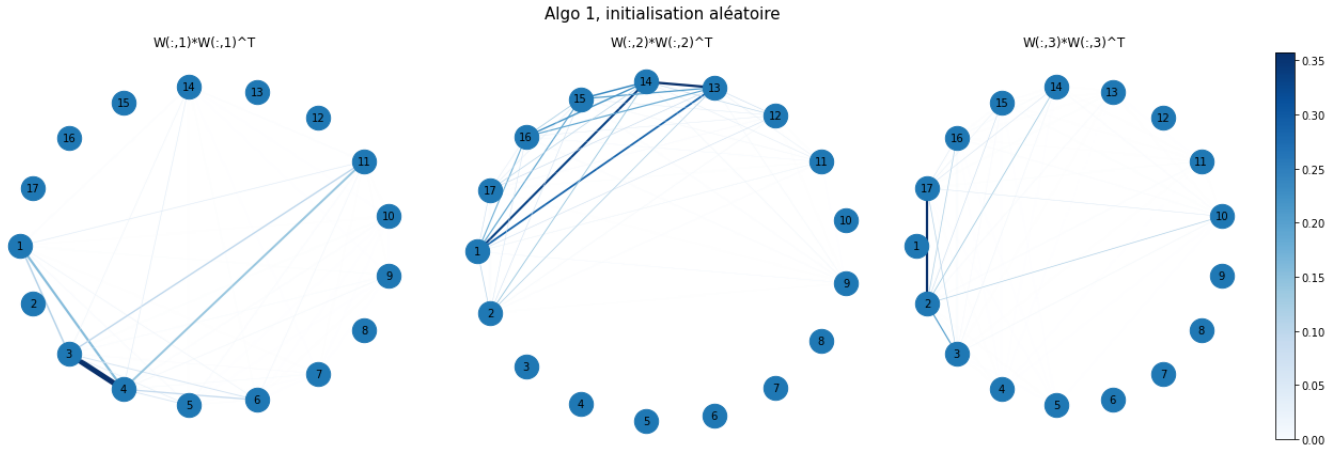


FIGURE 4.7 – Graphes reconstruits à partir des colonnes de W.

(3, 4, 5, 6, 7, 8, 11)	(1, 9, 12, 13, 14, 15, 16)	(2, 10, 17)
Excitation	**Revivre le traumatisme**	**Évitement**
7 éléments mal placés		

TABLE 4.4 – Classification des données psychiatriques algorithme 1 - aléatoire.

Algorithme 2 avec initialisation *kmeans*
erreur : 0.58

$$W = \begin{bmatrix} 0.0 & 0.0 & 0.33 \\ 0.0 & 0.04 & 0.2 \\ 0.0 & 0.0 & 0.29 \\ 0.04 & 0.02 & 0.0 \\ 0.14 & 0.0 & 0.0 \\ 0.13 & 0.0 & 0.08 \\ 0.09 & 0.0 & 0.07 \\ 0.09 & 0.0 & 0.0 \\ 0.06 & 0.0 & 0.01 \\ 0.03 & 0.0 & 0.01 \\ 0.05 & 0.04 & 0.0 \\ 0.0 & 0.0 & 0.09 \\ 0.0 & 0.07 & 0.0 \\ 0.0 & 0.1 & 0.0 \\ 0.0 & 0.01 & 0.27 \\ 0.0 & 0.0 & 0.39 \\ 0.01 & 0.14 & 0.0 \end{bmatrix} \quad S = \begin{bmatrix} 10.81 & 0.08 & 0.0 \\ 0.08 & 4.76 & 4.7 \\ 0.0 & 4.7 & 0.12 \end{bmatrix}$$

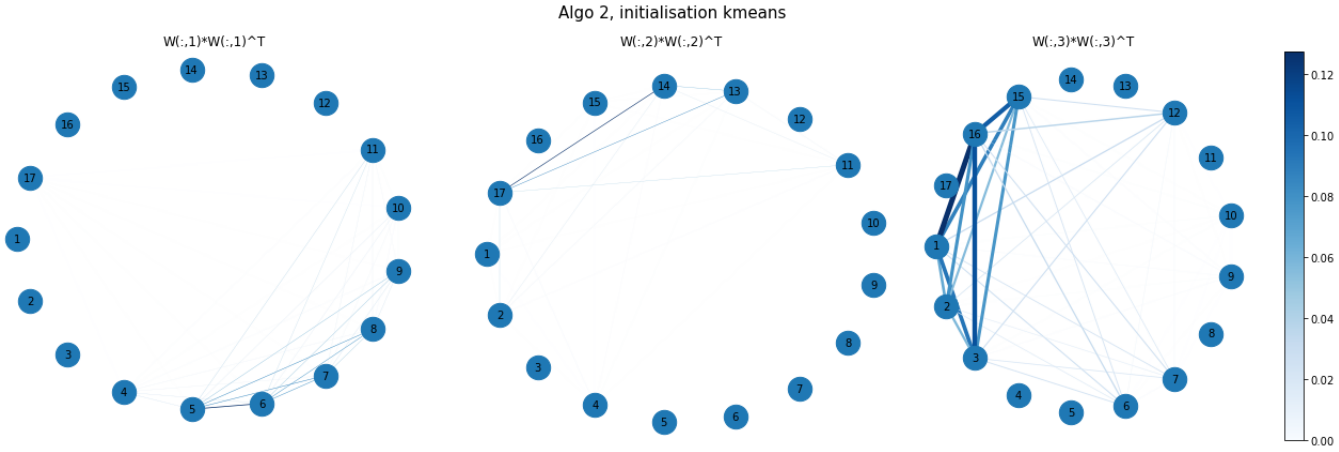


FIGURE 4.8 – Graphes reconstruits à partir des colonnes de W.

(4, 5, 6, 7, 8, 9, 10, 11)	(13, 14, 17)	(1, 2, 3, 12, 15, 16)
Excitation	**Évitement**	**Revivre le traumatisme**
5 éléments mal placés		

TABLE 4.5 – Classification des données psychiatriques algorithme 2 - *kmeans*.

Algorithme 2 avec initialisation aléatoire
erreur : 0.61

$$W = \begin{bmatrix} 0.0 & 0.04 & 0.19 \\ 0.0 & 0.13 & 0.01 \\ 0.0 & 0.04 & 0.14 \\ 0.03 & 0.02 & 0.03 \\ 0.19 & 0.0 & 0.0 \\ 0.2 & 0.0 & 0.0 \\ 0.16 & 0.0 & 0.0 \\ 0.14 & 0.0 & 0.0 \\ 0.11 & 0.03 & 0.0 \\ 0.05 & 0.0 & 0.01 \\ 0.09 & 0.06 & 0.0 \\ 0.0 & 0.0 & 0.06 \\ 0.0 & 0.16 & 0.0 \\ 0.0 & 0.2 & 0.0 \\ 0.0 & 0.09 & 0.08 \\ 0.0 & 0.0 & 0.18 \\ 0.06 & 0.0 & 0.08 \end{bmatrix} \quad S = \begin{bmatrix} 4.39 & 0.0 & 0.0 \\ 0.0 & 2.95 & 3.65 \\ 0.0 & 3.65 & 0.0 \end{bmatrix}$$

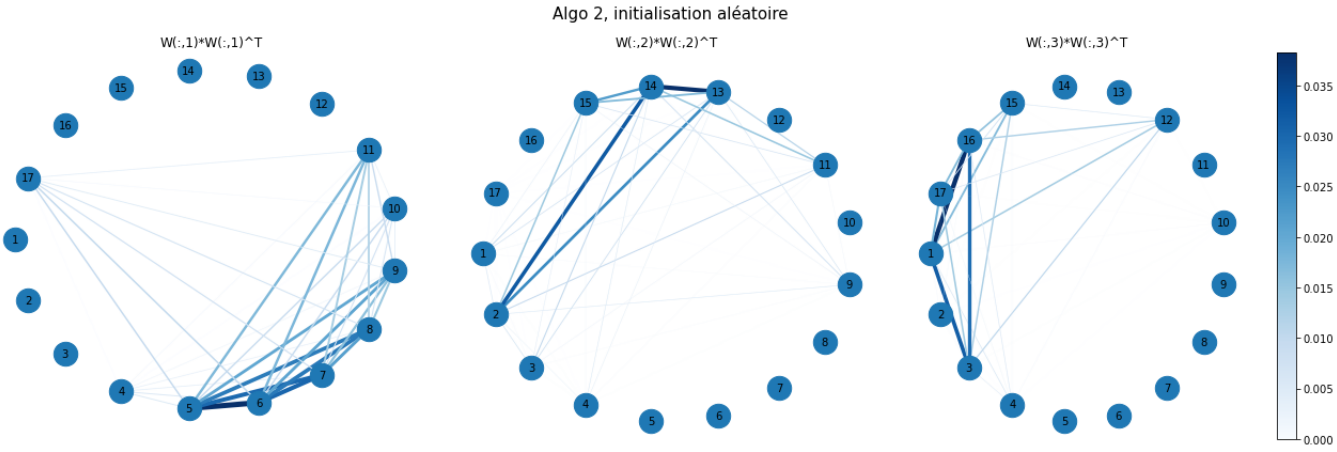


FIGURE 4.9 – Graphes reconstruits à partir des colonnes de W.

(4, 5, 6, 7, 8, 9, 10, 11)	(2, 13, 14, 15)	(1, 3, 12, 16, 17)
Excitation	**Revivre le traumatisme**	**Évitement**
4 éléments mal placés		

TABLE 4.6 – Classification des données psychiatriques algorithme 2 - aléatoire.

Selon le nombre d'éléments mal placés, le classement est le suivant :

1. algorithme 1 avec initialisation *kmeans* et algorithme 2 avec initialisation aléatoire ;
2. algorithme 2 avec initialisation *kmeans* ;
3. algorithme 1 avec initialisation aléatoire.

Les éléments mal placés sont mis en évidence en rouge. En analysant les lignes de W correspondant à ces éléments, on remarque que la valeur d'appartenance de ces symptômes au groupe auquel ils auraient dû être associés est rarement nulle. Elle est même souvent proche de la valeur maximale de la ligne (*i.e.* celle sur laquelle se base la classification). Cela signifie que certaines erreurs de classifications se jouent sur des différences de valeurs faibles.

Au contraire, certains symptômes sont systématiquement bien classés. Le cinquième symptôme, mis en évidence en vert, en est un bon exemple. On constate dans la matrice W que la différence entre les trois valeurs de la ligne est nette.

Pour l'évaluation de la qualité de la classification, nous n'avons pas les informations et les connaissances nécessaires pour juger de la pertinence des groupes proposés par l'algorithme de ODtri-symNMF. L'évaluation de ces communautés est donc faite par rapport à celles proposées dans [1]. Mais cette classification pourrait ne pas être la seule. D'autres liens valables pourraient exister, donnant un autre sens aux groupes. Il est d'ailleurs expliqué dans [1] qu'après analyse des résultats, les groupes formés représentaient des ensembles de comportement observés ensemble chez les sujets plutôt que les groupes de symptômes attendus (*i.e.* excitation, évitement et revivre le traumatisme).

L'examen de nos résultats met en évidence que la matrice S est restée diagonale pour les deux essais avec l'algorithme 1, ce que nous voulions éviter. Pour les 2 essais avec l'algorithme 2, une forte connexion est observée entre les communautés 2 et 3, c'est-à-dire l'évitement et le fait de revivre le traumatisme. Les valeurs 4.7 et 3.65 sont relevées pour l'initialisation *kmeans* et l'initialisation aléatoire. Il est raisonnable de penser que cette connexion est juste, les deux communautés partagent le symptôme 14. Dans une première classification de [1], les deux groupes partagent également le symptôme 13. De plus, c'est souvent entre ces deux groupes que les erreurs de classification se produisent. Ce qui pourrait refleter leur proximité.

L'algorithme 2 avec l'initialisation *kmeans* fait également apparaître une faible connexion (0.08) entre les communautés 1 et 2 (*i.e.* excitation et évitement). Ces communautés n'ont pourtant pas de symptôme en commun, que ce soit dans la première ou la deuxième version de la classification. Il est d'ailleurs mentionné que le groupe "excitation", qui est en réalité un groupe de symptômes représentant une humeur négative, est le groupe le plus homogène d'un point de vue clinique. Cela pourrait expliquer pourquoi il y a peu d'erreur sur ce groupe : il est bien défini et identifiable par rapport aux autres. On observe cependant que les symptômes 3 et 4, qui appartiennent d'ailleurs au groupe "évitement", se retrouvent parfois dans le groupe "excitation". Cela pourrait justifier que la connexion entre ces groupes n'est pas nulle, mais très faible.

Finalement, d'après nos résultats, les communautés excitation et revivre le traumatisme (*i.e.* 1 et 3 pour l'initialisation *kmeans* et 1 et 2 pour l'initialisation aléatoire) ne sont pas liées. Elles partagent pourtant le symptôme 10. Mais on remarque que les erreurs de classification de la communauté 1 sont liées à "évitement" et non à "revivre le traumatisme". Autrement dit, on peut imaginer qu'en dehors du symptôme 10, les symptômes de "excitation" et de "revivre le traumatisme" sont éloignés, étant donné qu'on n'observe aucune erreur de classification entre ces deux groupes.

Pour conclure cette expérience, si l'on combine les résultats des différents classements, l'algorithme 1 avec initialisation *kmeans* est le plus performant à chaque fois. Cependant, l'algorithme 2 est le seul à réussir à donner du sens à la matrice S, moyennant une révision de la matrice W initiale. C'est pourquoi le meilleur compromis semble être l'algorithme 2 avec initialisation *kmeans*.

4.3 Club de karaté de Zachary

Reprenons une dernière fois le jeu de données du club de karaté.
Jusqu'à présent, il était question d'un rang de factorisation de 2, pour identifier les 2 communautés : celle restée avec l'administrateur et celle partie avec l'instructeur.
Dans cette section, nous allons tenter de factoriser la matrice d'adjacence X avec un rang de 4 pour découvrir si des sous-groupes se cachent dans les 2 groupes clairement identifiés. Le cas échéant, la matrice S nous renseignera les connexions entre ces 4 groupes.
X étant binaire, c'est également l'occasion d'analyser le comportement de l'algorithme sur des données binaires non synthétiques.

4.3.1 Analyse de l'erreur

Nous calculons les erreurs relatives résiduelles moyennes de la même manière que pour les données psychiatriques. Pour un problème comme celui-ci (*i.e.* binaire, rang 4, matrice (34×34)), on devrait s'attendre à une erreur entre 0.6 et 0.8 (*cfr.* annexe C).

	Algorithme 1	**Algorithme 2**
Kmeans	0.756	0.794
Aléatoire	0.756	0.794

TABLE 4.7 – Erreur relative résiduelle moyenne sur les données du club de karaté.

Nous sommes dans le cas où le α de la mise à l'échelle vaut 0. On a donc une erreur très importante et il n'y a pas de différence entre les initialisations étant donné qu'après multiplication par la racine carrée de α, la matrice W initiale est nulle dans tous les cas.

Nous nous passons de la mise à l'échelle à l'initialisation. Voici les erreurs moyennes recalculées.

	Algorithme 1	**Algorithme 2**
Kmeans	0.669 -0.09	0.786 -0.008
Aléatoire	0.746 -0.01	1 +0.206

TABLE 4.8 – Erreur relative résiduelle moyenne sur les données du club de karaté, sans mise à l'échelle initiale.

On voit que l'erreur a légèrement diminué, de 0.8% à 9%. Sauf pour l'algorithme 2 avec l'initialisation aléatoire qui renvoie systématiquement des W remplies de 1 et des S nulles.

Selon l'erreur relative résiduelle moyenne et pour cette matrice, le classement de la meilleure combinaison algorithme - initialisation est le suivant :

1. algorithme 1 avec initialisation *kmeans* ;
2. algorithme 2 avec initialisation *kmeans* ;
3. algorithme 1 avec initialisation aléatoire ;
4. algorithme 2 avec initialisation aléatoire.

Le même qu'à ce stade de l'analyse des données psychiatriques.

4.3.2 Analyse des communautés

Encore une fois, au-delà de l'analyse de l'erreur, il est important d'avoir des matrices W et S qui ont du sens. Comme pour les données psychiatriques, le problème des S diagonales apparait. Nous utilisons encore la technique de la valeur maximale pour chaque ligne afin de pousser les algorithmes à faire évoluer S vers une forme non diagonale.

	Algorithme 1	**Algorithme 2**
Kmeans	0.745 +0.076	0.537 -0.249
Aléatoire	0.692 -0.054	0.997 -0.003

TABLE 4.9 – Erreur relative résiduelle moyenne sur les données du club de karaté, sans mise à l'échelle initiale et avec une nouvelle initialisation.

On observe des différences significatives par rapport aux derniers résultats. L'algorithme 1 avec l'initialisation *kmeans* qui était jusqu'à présent constamment le meilleur et désormais troisième. L'algorithme 2 avec *kmeans* s'améliore beaucoup et devient le plus performant. C'est d'ailleurs le seul que nous analyserons. Les autres combinaisons ne réussissant pas à donner des matrices satisfaisantes.

Algorithme 2 avec initialisation *kmeans*
erreur : 0.54

$$W = \begin{bmatrix}
0.0 & 0.93 & 0.0 & 0.0 \\
0.0 & 0.93 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 1.0 \\
0.0 & 0.0 & 0.0 & 1.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 1.0 \\
0.0 & 0.0 & 1.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 1.0 \\
0.0 & 0.0 & 1.0 & 0.0 \\
0.0 & 0.0 & 1.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 1.0 \\
0.0 & 0.0 & 1.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 1.0 \\
0.0 & 0.0 & 1.0 & 0.0 \\
0.0 & 0.0 & 1.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 1.0 & 0.0 \\
0.0 & 0.0 & 1.0 & 0.0 \\
0.0 & 0.0 & 1.0 & 0.0 \\
0.88 & 0.0 & 0.0 & 0.0 \\
0.88 & 0.0 & 0.0 & 0.0
\end{bmatrix} \quad S = \begin{bmatrix}
1.31 & 0.0 & 1.14 & 0.0 \\
0.0 & 1.16 & 0.0 & 1.08 \\
1.14 & 0.0 & 0.0 & 0.0 \\
0.0 & 1.08 & 0.0 & 0.0
\end{bmatrix}$$

Pour identifier les groupes, la valeur maximale par ligne a été utilisée. Cette fois-ci, il n'y a pas eu de problème d'ambiguïté, car il n'y a qu'une valeur non nulle par ligne. On observe également 13 lignes comptant 4 zéros. Elles sont mises en évidence en bleu. Ce qui en fait un groupe plus important que les 4 communautés identifiées.

(33, 34)	L'administrateur et un membre très proche, disons l'extrême gauche.
(1, 2)	L'instructueur et un membre très proche, disons l'extrême droite
(9, 15, 16, 19, 21, 23, 24, 30, 31, 32)	Une partie de la communauté "administrateur", la gauche
(3, 4, 8, 14, 18, 20, 22)	Une partie de la communauté "instructeur", la droite
(5, 6, 7, 10, 11, 12, 13, 17, 25, 26, 27, 28, 29)	Les centraux. En violet, les membres rattachés à l'instructeur. En orange, ceux rattachés à l'administrateur

TABLE 4.10 – Classification des données du club de karaté.

Nous voulions 4 groupes, mais avec les non classés (les centraux), nous avons finalement 5 groupes. On remarque qu'aucun membre n'est mal placé bien que 13 ne soient pas placés du tout.

Si on se penche sur S on constate une connexion entre le groupe 1 et 3 (*i.e.* l'extrême gauche et la gauche) et entre le groupe 2 et 4 (*i.e.* l'extrême droite et la droite). Ce qui semble tout à fait correcte.
Pour expliquer le groupe des centraux, on peut reprendre l'analyse de la symNMF, via matrice W_{symNMF} qui divise les membres en 2 groupes. Certaines lignes attestent de l'appartenance totale à un groupe : on trouve un zéro dans l'autre colonne. D'autres sont plus floues : il n'y a de zéro dans aucune des deux colonnes. Cependant, on ne peut identifier de corrélation entre les membres centraux selon symNMF et notre groupe de non classés.

La figure 4.10 nous montre les résultats d'un point de vue graphique. Les 4 groupes sont nettement visibles sur les matrices. Étant donné que les entrées de W sont soit nulles, soit égales à 1 ou très proches, la différence est nette. Pour les graphes, les deux premières petites communautés sont facilement identifiables. Il y a un lien bleu entre les deux duos. Les deux autres groupes sont moins perceptibles, car les membres au sein de ces communautés ne sont pas tous directement liés entre eux. Par exemple, 9 et 15 sont dans le même groupe, mais ne sont pas directement liés dans X. Il n'y a donc aucune arrête les reliant.

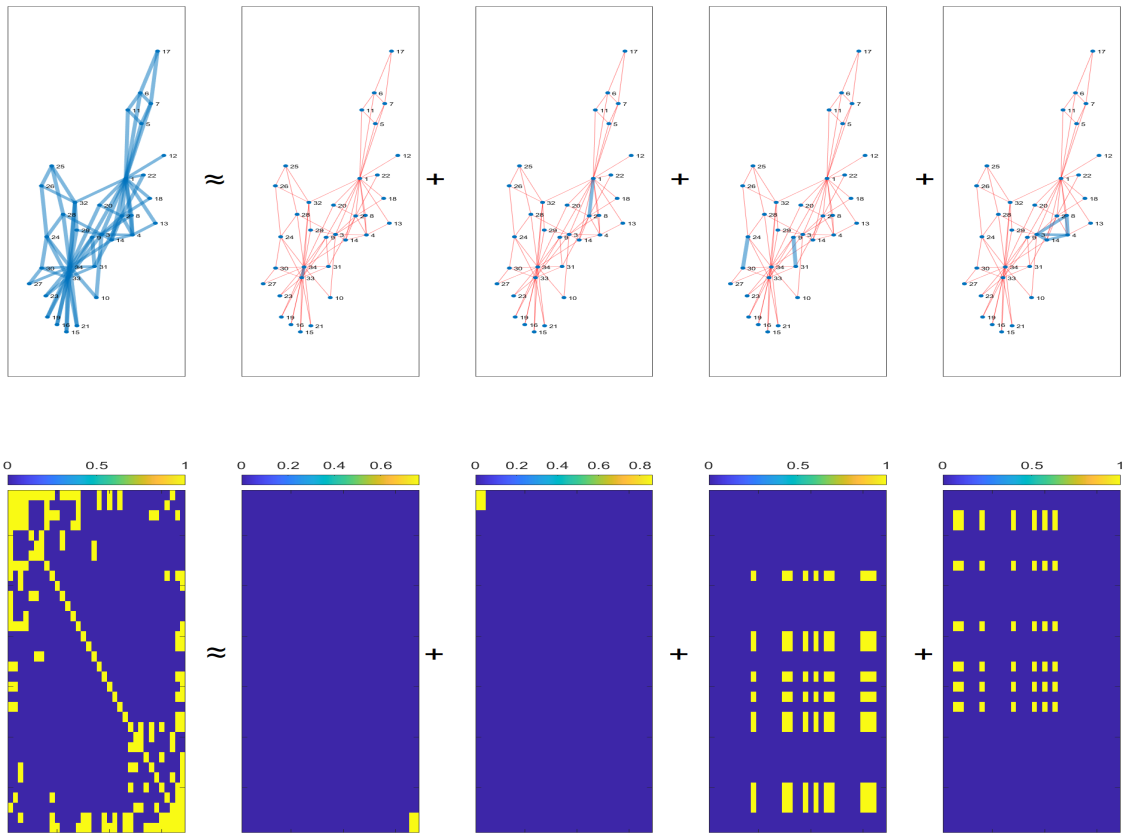


FIGURE 4.10 – Illustration de la ODtri-symNMF sur le jeu de données *Zachary's karate club*. En haut : les graphes dont les poids des arêtes sont attribués par les matrices correspondantes. L'épaisseur de l'arête est proportionnelle à son poids, si ce dernier est inférieur à 0.1, l'arête est rouge. En bas et de gauche à droite : X et $W(:,i)W(:,i)^T$ pour $i = (1,2,3,4)$. Illustration réalisée avec le code de [3].

Pour conclure sur ce jeu de données, il se confirme que les données binaires sont plus difficiles à traiter que les données réelles. La combinaison de l'algorithme 2 avec *kmeans* est encore une fois le meilleur compromis. Il apporte des résultats sensés avec une erreur relativement faible en comparaison aux autres combinaisons d'algorithme et d'initialisation.
La pertinence de la classification peut être jugée jusqu'à un certain point. Pour aller plus loin, il faudrait plus de connaissances sur les membres du club.

Chapitre 5

Conclusion

Ce travail présente un nouveau modèle de factorisation symétrique positive de matrices : la tri-factorisation symétrique positive hors diagonale de matrices.
Ce modèle contribue au large éventail de variantes de la factorisation symétrique positive de matrices et de manière plus étendue, aux variantes de la factorisation positive de matrices.

Les bases mathématiques sur lesquelles reposent le modèle ont été exposées et la logique informatique qui a permis son exécution a été détaillée.

Une fois construit, l'algorithme a été testé sur des données synthétiques pour une évaluation quantitative. La version initiale de l'algorithme, complétée d'une initialisation basée sur la méthode de détection de communauté *kmeans*, s'est démarquée des alternatives proposées.

Des jeux de données réelles ont également servi à évaluer le modèle, cette fois-ci d'un point de vue plus qualitatif. L'enjeu était d'identifier correctement les communautés présentes dans les réseaux ainsi que les connexions qui les relient. Dans ce contexte, c'est l'algorithme dans sa seconde version, toujours complétée d'une initialisation basée sur *kmeans*, qui s'est révélé comme le meilleur choix, bien que des ajustements lors de l'initialisation furent nécessaires.

Globalement, le modèle parvient à approximer les matrices qu'on lui présente et à classer les éléments en groupes cohérents. Malheureusement, il peine parfois à extraire les interactions entre ces communautés, ce qui devrait être une de ses forces.
On notera une forte sensibilité à l'initialisation, particulièrement pour les données binaires qui sont moins bien appréhendées par le modèle. Et on regrettera également une erreur relative souvent élevée.

Finalement, si les avantages du modèle tri-symNMF, premier parent de notre modèle, ont largement été mis en avant (*i.e.* informations sur les connexions entre les communautés), les avantages liés au modèle ODsymNMF, deuxième parent, sont passés au second plan. Ces avantages étant principalement liés aux performances de convergence et de temps de calcul, une comparaison avec d'autres modèles sur ces points pourront être ajoutée à ce travail.

Chapitre 6

Perpectives

De nombreuses directions de travail peuvent être envisagées en partant de ce projet.

Avant d'aller plus loin avec ce modèle, une première perspective de travail serait de tenter de diminuer l'erreur. Il serait également bienvenu de trouver des solutions pour mieux traiter les données binaires qui représentent une grande portion du domaine d'application des méthodes de détection de communauté.

Pour ce faire, plusieurs pistes peuvent être explorées. De nombreux modèles introduisent un ou plusieurs paramètres dans leur objectif pour pénaliser la matrice S afin d'éviter sa forme diagonale. D'autres imposent une seule entrée non nulle par ligne de W [1].
Il serait aussi intéressant d'envisager une révision de la méthode des médianes pondérées contraintes ou éventuellement de tester des alternatives. Certains problèmes rencontrés durant les expériences numériques semblent liés à cette méthode ou du moins, à sa forme actuelle.

Une autre direction de travail serait également de faire migrer le modèle vers l'utilisation de la norme de Frobenius. Celle-ci étant beaucoup plus utilisée dans le domaine de la symNMF.

Une version de l'initialisation à partir de symNMF ou d'autres techniques de détection de communautés autres que *kmeans* serait également une idée.

Finalement, comme mentionné dans le chapitre précédent, une étude sur les performances de convergence et de temps de calcul pourrait être menée. Pour cela, il serait préférable de revoir le code qui n'est pas toujours optimisé.

1. tri-factorisation positive orthogonale - *tri-ONMF*

Aussi, il n'y a aucune information, à ce stade, sur la capacité de ce modèle à gérer des grandes et très grandes quantités de données.

Ensuite, ODtri-symNMF pourra être comparé aux autres modèles de détection de communauté, qu'ils soient issus de la famille de la NMF ou pas.

De nombreuses autres possibilités existent.

Bibliographie

[1] Pierre De Handschutter, Nicolas Gillis, and Wivine Blekic. Deep symmetric matrix factorization. *European Signal Processing Conference*, 2023.

[2] Sacha Epskamp and Eiko I Fried. A tutorial on regularized partial correlation networks. *Psychological methods*, 23(4) :617, 2018.

[3] Nicolas Gillis. *Nonnegative matrix factorization.* Society for Industrial and Applied Mathematics, 2021.

[4] Nicolas Gillis and Robert J Plemmons. Dimensionality reduction, classification, and spectral mixture analysis using non-negative underapproximation. *Optical Engineering*, 50(2) :027001–027001, 2011.

[5] Chaya Gurwitz. Weighted median algorithms for l1 approximation. *BIT*, 30(2) :301–310, 1990.

[6] Ngoc-Diep Ho. *Nonnegative matrix factorization algorithms and applications.* PhD thesis, Citeseer, 2008.

[7] François Moutier, Arnaud Vandaele, and Nicolas Gillis. Off-diagonal symmetric nonnegative matrix factorization. *Numerical Algorithms*, 88 :939–963, 2021.

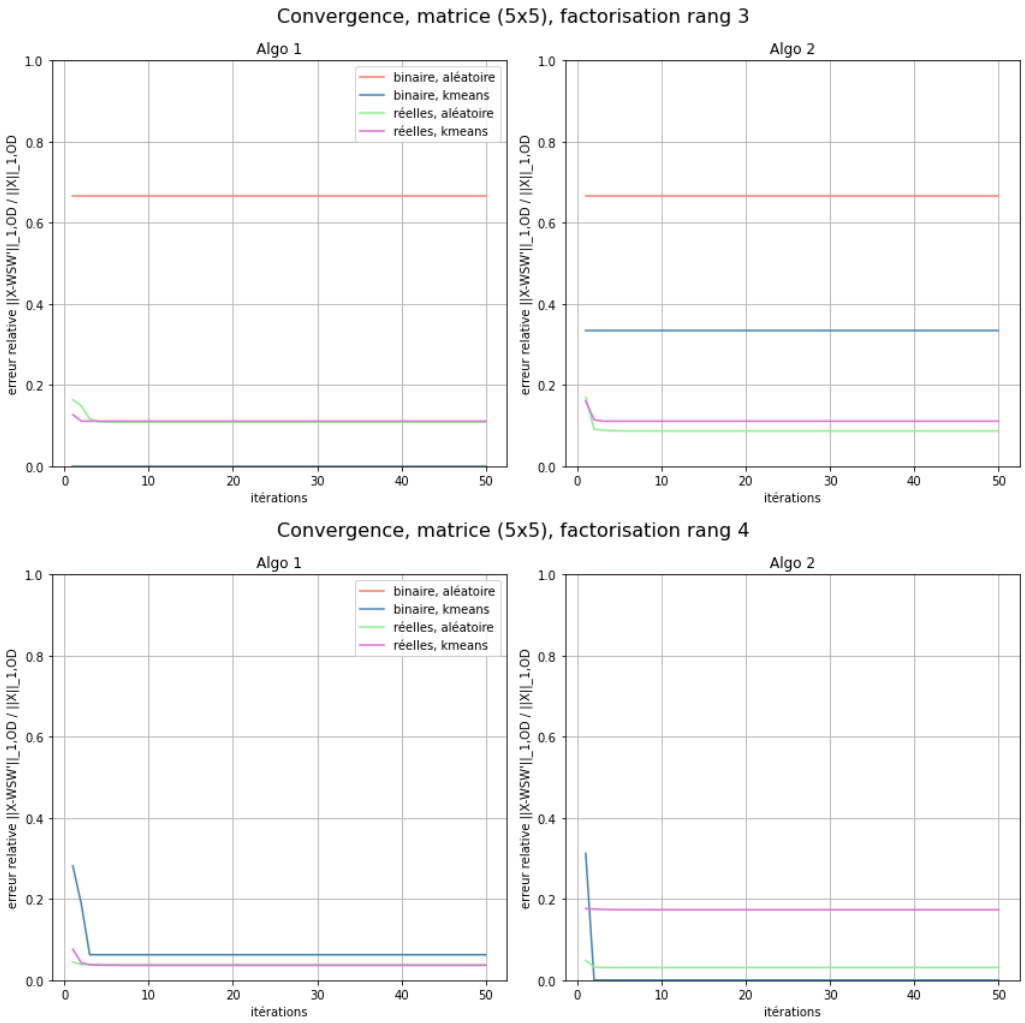
[8] Yulong Pei, Nilanjan Chakraborty, and Katia Sycara. Nonnegative matrix tri-factorization with graph regularization for community detection in social networks. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

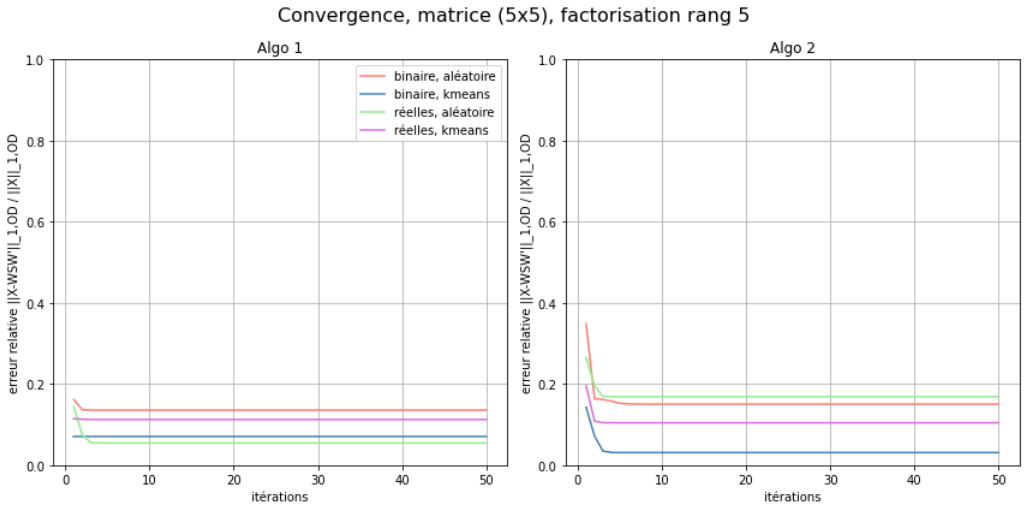
[9] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale : a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596, 2013.

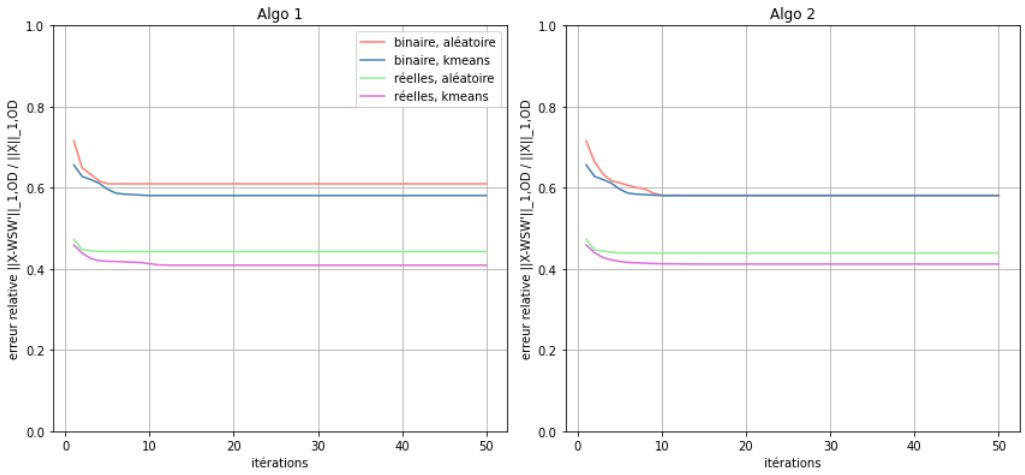
[10] Yu Zhang and Dit-Yan Yeung. Overlapping community detection via bounded nonnegative matrix tri-factorization. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 606–614, 2012.

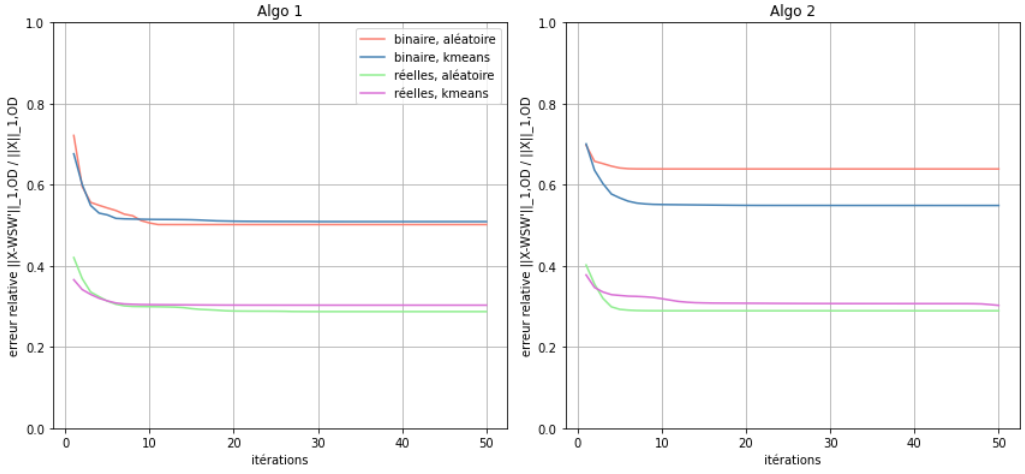
Annexe A

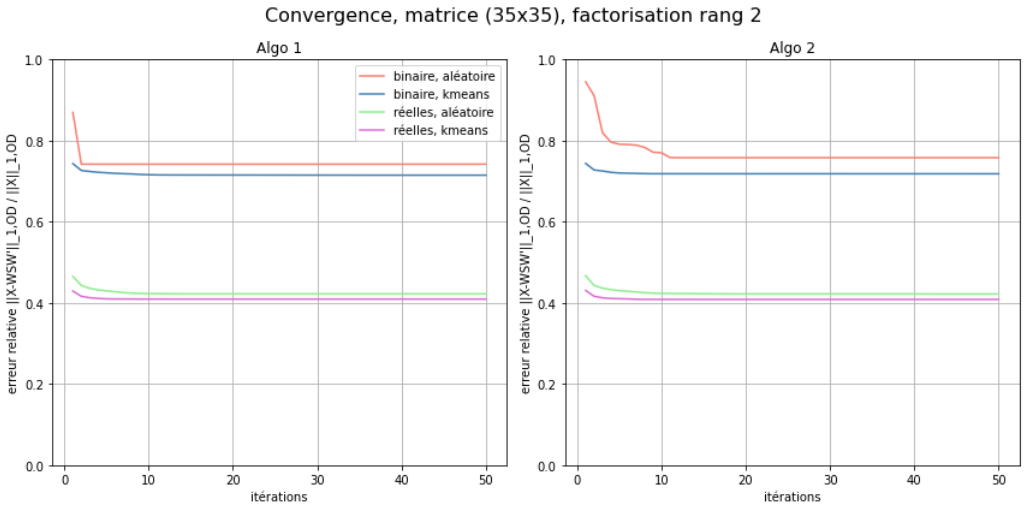
Convergence

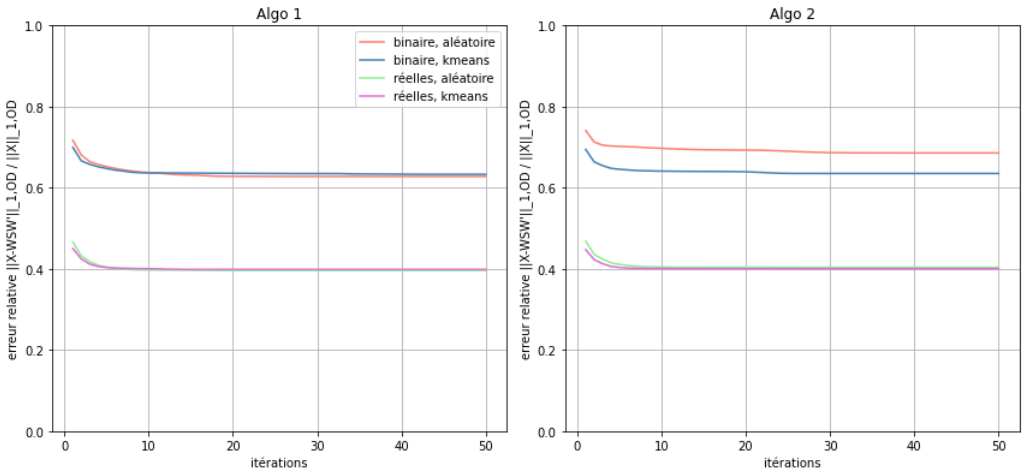


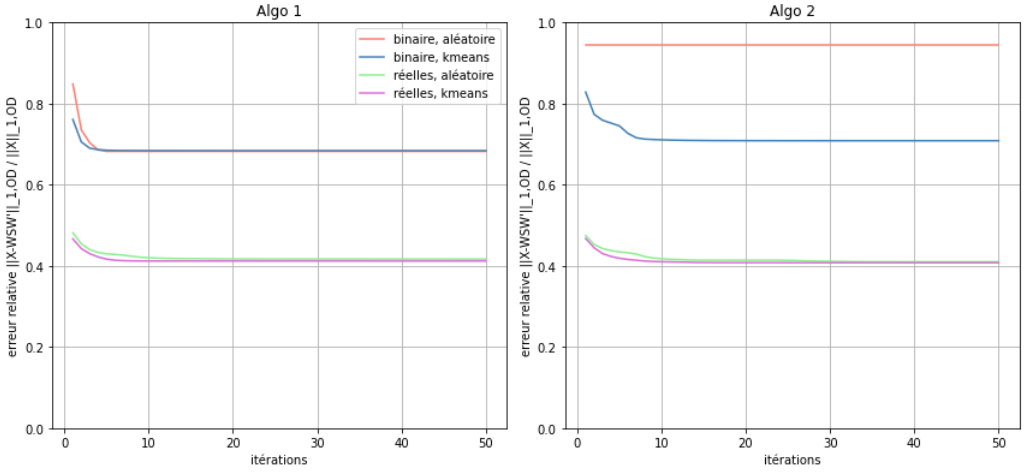

Convergence, matrice (5x5), factorisation rang 5
Algo 1
Algo 2
binaire, aléatoire
binaire, kmeans
réelles, aléatoire
réelles, kmeans
erreur relative ||X-WSW'||_1,OD / ||X||_1,OD
itérations


Convergence, matrice (20x20), factorisation rang 2
Algo 1
Algo 2
binaire, aléatoire
binaire, kmeans
réelles, aléatoire
réelles, kmeans
erreur relative ||X-WSW'||_1,OD / ||X||_1,OD
itérations


Convergence, matrice (20x20), factorisation rang 5
Algo 1
Algo 2
binaire, aléatoire
binaire, kmeans
réelles, aléatoire
réelles, kmeans
erreur relative ||X-WSW'||_1,OD / ||X||_1,OD
itérations


Convergence, matrice (35x35), factorisation rang 2
Algo 1
Algo 2
binaire, aléatoire
binaire, kmeans
réelles, aléatoire
réelles, kmeans
erreur relative ||X-WSW'||_1,OD / ||X||_1,OD
itérations


Convergence, matrice (35x35), factorisation rang 3
Algo 1
Algo 2
binaire, aléatoire
binaire, kmeans
réelles, aléatoire
réelles, kmeans
erreur relative ||X-WSW'||_1,OD / ||X||_1,OD
itérations


Convergence, matrice (35x35), factorisation rang 4
Algo 1
Algo 2
binaire, aléatoire
binaire, kmeans
réelles, aléatoire
réelles, kmeans
erreur relative ||X-WSW'||_1,OD / ||X||_1,OD
itérations

Annexe B

Comparaisons des versions

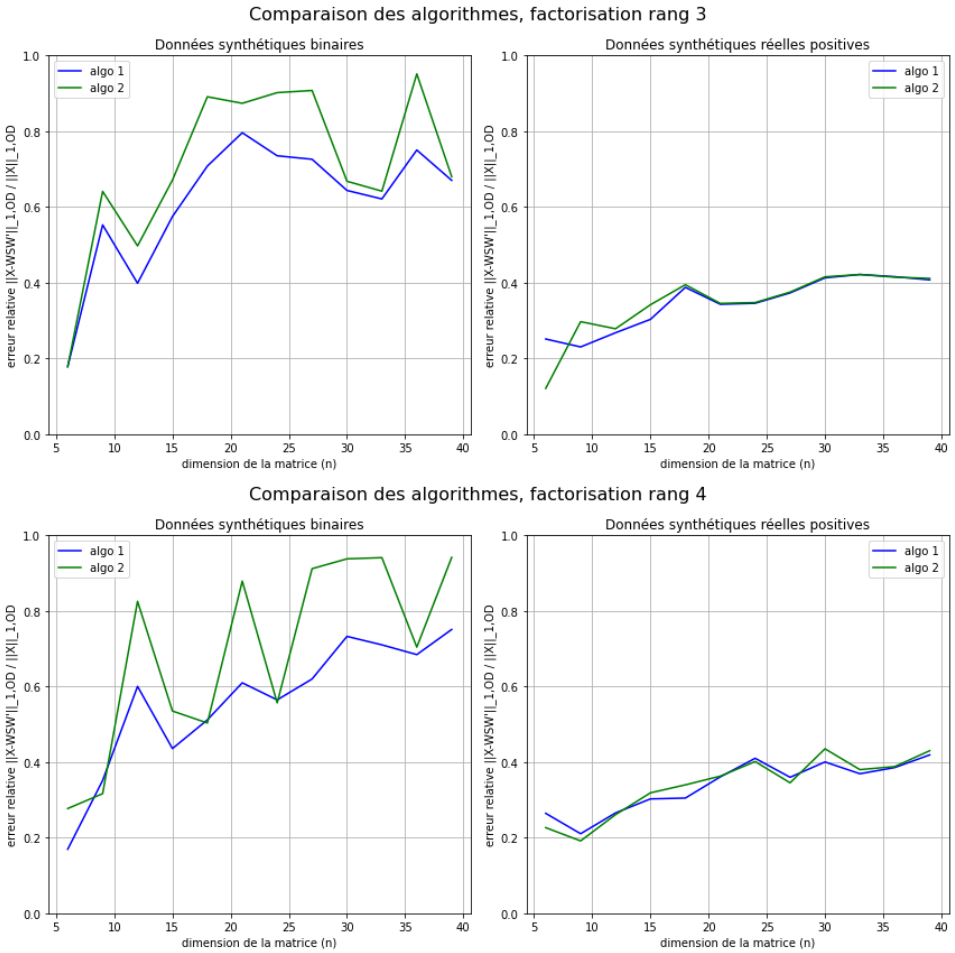


FIGURE B.1 – Comparaison des algorithmes sur des données synthétiques

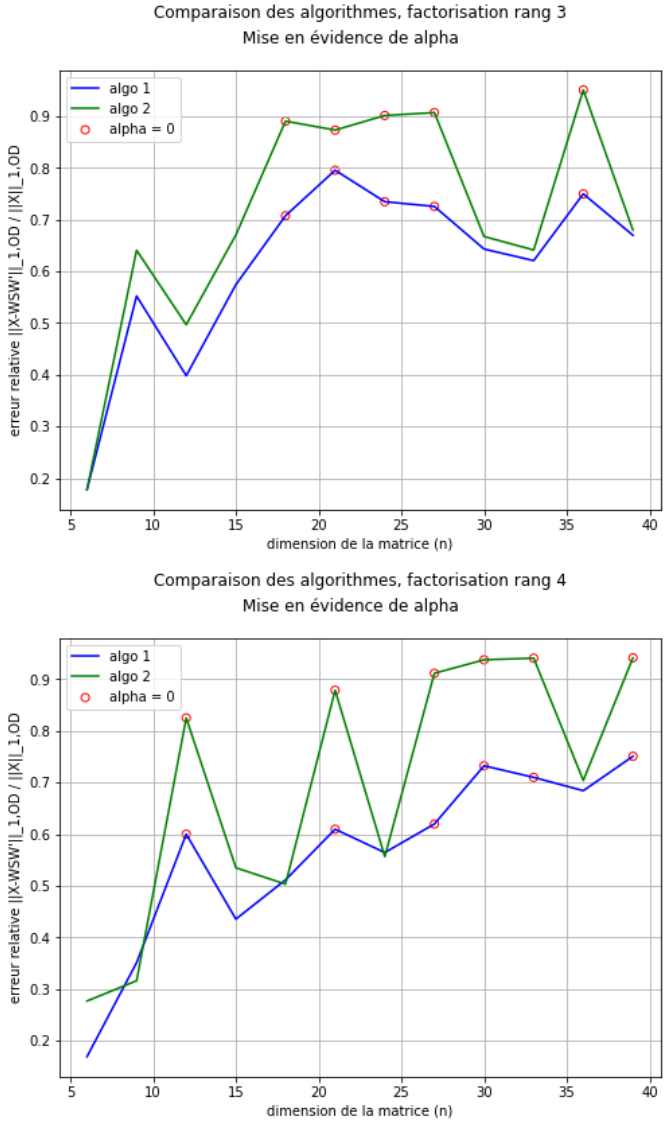
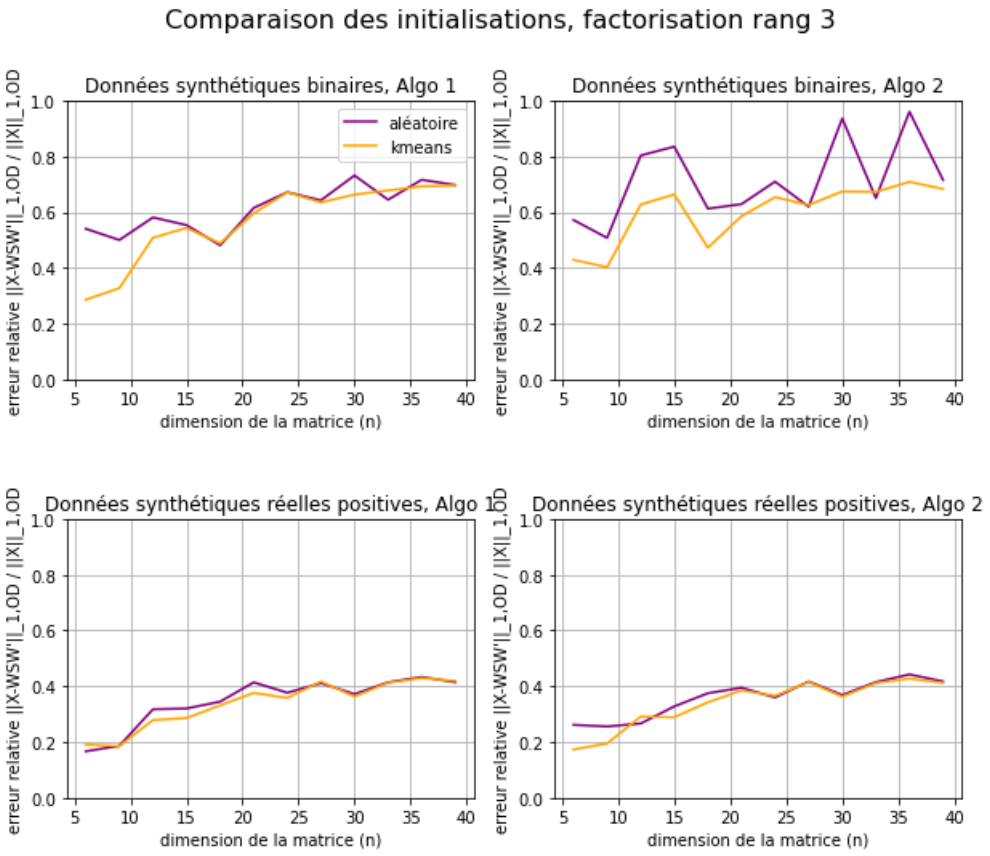
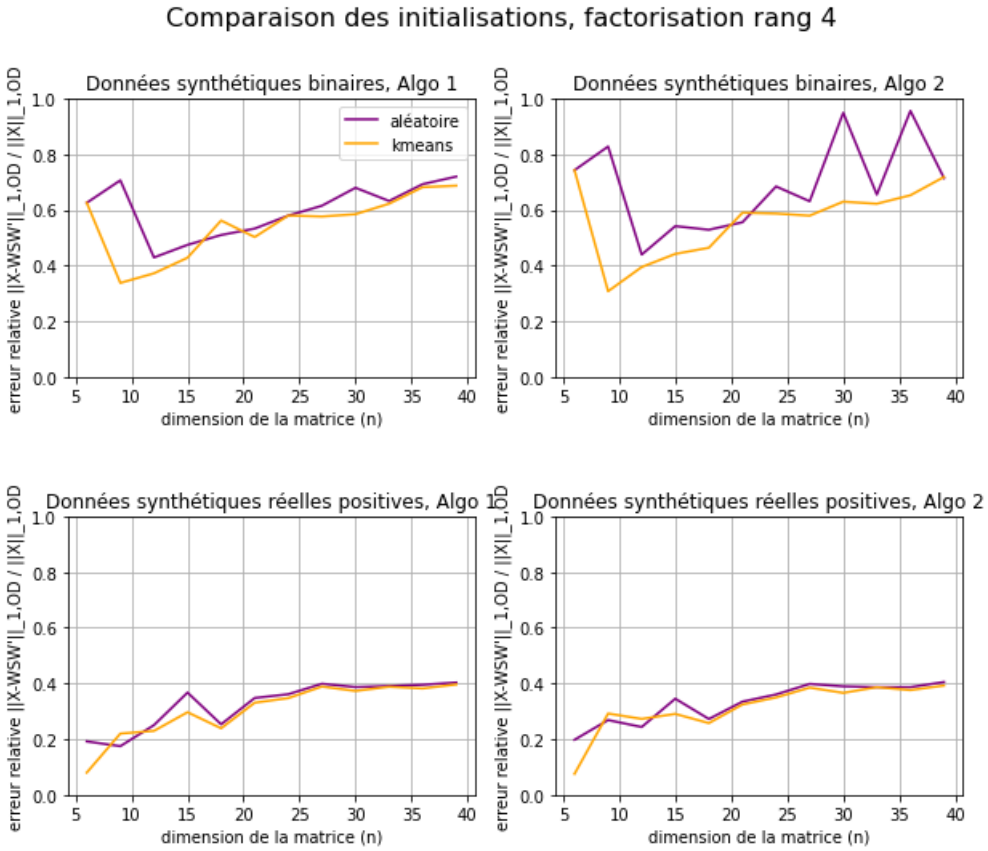


FIGURE B.2 – Mise en évidence des $\alpha = 0$

Annexe C

Comparaisons des initialisations




Comparaison des initialisations, factorisation rang 4
Données synthétiques binaires, Algo 1
Données synthétiques binaires, Algo 2
Données synthétiques réelles positives, Algo 1
Données synthétiques réelles positives, Algo 2
aléatoire
kmeans
erreur relative ||X-WSW'||_1,OD / ||X||_1,OD
dimension de la matrice (n)