STATISTIQUE
SCIENCE DES DONNÉES
UNIVERSITÉ DE MONTPELLIER

Master 2 Statistiques et Sciences des Données

Filtrage Collaboratif

DIAS Pierre
FESTOR Quentin
LAHJIOUJ Aïcha

Année 2024 - 2025

Table des matières

1 Introduction

L'objectif principal des entreprises et des sites marchands est d'augmenter le nombre de ventes. Pour cela, les sites de commerce en ligne mettent en place des systèmes de recommandation afin d'identifier les produits les plus susceptibles d'intéresser les visiteurs ou clients. Lorsque ces systèmes se basent uniquement sur les interactions entre clients et produits, il est question de filtrage collaboratif. Parmi ces approches, les plus performantes reposent sur la recherche de modèles exploitant un nombre réduit de facteurs latents (c'est-à-dire des variables qui ne sont pas directement observées, mais plutôt déduites), capables d'expliquer les interactions en faible dimension. D'autres systèmes s'appuient sur des données spécifiques aux clients (*user-based*), aux produits (*item-based*) ou encore sur des approches hybrides, qui ne seront pas abordées ici. Le principal problème du filtrage collaboratif réside dans la gestion des bases de données incomplètes, où des informations manquent. L'enjeu est de reconstituer les données absentes en s'appuyant sur celles qui sont disponibles.

Cela pose ainsi la question de la reconstruction des matrices creuses, indispensable pour un filtrage collaboratif.

Dans le cadre général, les données sont organisées sous forme d'une matrice X très creuse. Chaque ligne correspond à un individu i, et chaque colonne représente soit le nombre d'achats du produit j, soit une note d'évaluation allant de 1 à 5, avec des données manquantes (**NA**).

Ce concept sera illustré par un exemple référence basé sur les notes de séries allant de 1 à 5 si un utilisateur les a vues, et rien sinon.

Utilisateur	**The Walking Dead**	**The Witcher**	**Dark**	**Gossip Girl**	**The Crown**
Utilisateur 1	3	3	**NA**	2	3
Utilisateur 2	3	**NA**	**NA**	3	1
Utilisateur 3	1	2	**NA**	5	2
Utilisateur 4	**NA**	**NA**	5	3	2
Utilisateur 5	**NA**	3	4	**NA**	**NA**
Utilisateur 6	3	1	**NA**	4	**NA**
Utilisateur 7	2	3	4	**NA**	4

TABLE 1 – Notes des séries par utilisateur

Trois méthodes seront évoquées : la factorisation Non-négative de Matrice (NMF) [5], la décomposition en valeurs singulières (SVD) et la complétion de matrice [6].

Dans la suite, la méthode de factorisation Non-Négative de Matrice sera principalement décrite, puis les deux autres méthodes seront abordées.

2 Factorisation Non-négative de Matrice (*NMF*)

2.1 Principes

Dans un premier temps, les valeurs manquantes de la base de données sont remplacées par des valeurs typiques (comme des "0").
Soit $X_{n\times p}$ contenant uniquement des valeurs non négatives, sans lignes ni colonnes entièrement nulles. Soit r, le nombre de facteurs latents, un entier relativement petit par rapport à n et p (dans notre exemple, cela pourrait correspondre aux genres des séries, à leur durée ou encore à leur date de sortie).

La *NMF* de X consiste à trouver deux matrices parcimonieuses $W_{n\times r}$ et $H_{r\times p}$ à entrées positives (ne contenant que des valeurs positives ou nulles), telles que

$$X \approx WH$$

Le choix de r (où $r \ll \min(n,p)$) permet une réduction significative de la dimensionnalité.
La factorisation est résolue par la recherche d'un optimum local du problème d'optimisation suivant :

$$\min_{W,H\geq 0}[L(X,WH) + P(W,H)].$$

P est une pénalisation optionnelle qui favorise certaines propriétés des matrices W et H, telles que la parcimonie ou la régularité des solutions pour les données spectrales.

La fonction de perte L mesurant la qualité de l'approximation est souvent choisie de la manière suivante :

- Moindres carrés (LS) : $L_{LS}(A,B) = \text{tr}((A-B)(A-B)^T) = \sum_{i,j}(a_{i,j}-b_{i,j})^2$,
- Divergence de Kullback-Leibler (KL) : $L_{KL}(A,B) = KL(A||B) = \sum_{i,j} a_{i,j} \log\left(\frac{a_{i,j}}{b_{i,j}}\right) - a_{i,j} + b_{i,j}$.

Dans la librairie `NMF`[3], les variables (*features*) sont en lignes et les échantillons (*samples*) en colonnes. Cette disposition n'affecte pas les résultats si le critère des moindres carrés est utilisé (LS), mais elle a son importance avec la divergence de Kullback-Leibler, qui introduit une dissymétrie.

La solution à ce problème d'optimisation est généralement locale, car la fonction objectif n'est pas convexe par rapport à W et H. De plus, la solution n'est pas unique (voir 5.1) ; toute matrice $D_{r\times r}$ non négative et inversible fournit des solutions équivalentes :

$$X \approx WDD^{-1}H$$

Une fois la factorisation obtenue, les matrices W et H peuvent être utilisées pour des classifications (CAH, k-means), des représentations (ACP, MDS) et des prévisions à l'aide de méthodes d'apprentissage.

2.2 Algorithmes

De nombreuses variantes algorithmiques et approches de pénalisation ont été proposées pour la factorisation non-négative de matrices (NMF), avec une majorité d'implémentations en `Matlab` et en `R`. Les algorithmes les plus fréquemment cités incluent :

- L'algorithme standard NMF avec mise à jour multiplicative (décrite en annexe 5.2) ;
- L'algorithme des moindres carrés alternés (ALS) ;
- La descente de gradient.

Chacun de ces algorithmes peut être initialisé de différentes manières, par des initialisations aléatoires ou via des techniques comme la décomposition en valeurs singulières non négatives ($NNSVD$).

Les choix de la fonction objectif, de l'algorithme à utiliser, des méthodes d'initialisation et d'autres paramètres impliquent une série de comparaisons. Bien que certains algorithmes puissent parfois converger vers des points stationnaires qui ne sont pas nécessairement minimaux, l'ALS se distingue par sa flexibilité, permettant d'éviter des solutions locales sous-optimales. La librairie `NMF` de `R` offre plusieurs méthodes permettant de lancer plusieurs initialisations aléatoires pour choisir les meilleures configurations. A noter que la complexité algorithmique du NMF est un problème NP (*nondeterministic polynomial*).

2.3 Critères de choix

Différents critères guident le choix des méthodes, algorithmes et paramètres, notamment le rang r de factorisation, influençant les résultats. Parmi ces critères figurent les résidus, la part de variance expliquée, l'indice de parcimonie (pour évaluer l'ajustement) et des mesures de stabilité (corrélation, pureté, entropie, silhouette).

La stabilité des résultats de la NMF s'évalue par des critères de classification non supervisée, où chaque facteur (éléments des matrices W ou H) est identifié selon sa contribution. Le choix important du rang r repose sur des heuristiques, souvent déterminé par l'interprétation contextuelle et la stabilité des classifications. Parfois, il est préférable de choisir un rang r sous optimal afin de mieux faire ressortir les couples.

2.4 Graphiques

La librairie `NMF` offre divers graphiques inspirés de la bioinformatique, notamment des *heatmaps* qui montrent les valeurs positives ou nulles des matrices W et H issues de la décomposition. Une classification ascendante hiérarchique est appliquée par défaut pour organiser les lignes et les colonnes, mais ces paramètres peuvent être personnalisés.

Les cartes de consensus aident à choisir la méthode et la dimension de la décomposition en indiquant la stabilité des résultats à travers différentes exécutions de l'algorithme. De plus, des graphiques illustrent les indicateurs de performance en fonction du rang r.

3 Autres Méthodes

Il existe d'autres méthodes alternatives à la *NMF* pour approcher la matrice X. Ci-dessous sont présentées deux des méthodes les plus couramment utilisées.

3.1 Par Décomposition en Valeurs Singulières (*SVD*)

Une nouvelle fois, les valeurs manquantes de la base de données sont remplacées par des "0".
Le principe de la SVD est le suivant : L'objectif est de se rapprocher de la diagonalisation de matrice $X = U\Lambda U^T$ avec U orthogonale obtenue avec le théorême spectrale. Cependant la matrice X n'est pas diagonalisable. Une matrice V orthogonale est donc introduite pour décomposer X telle que $X = U\Lambda V^T$. Cela implique dans le cas où $n \leq p$ que :

$$X = \begin{bmatrix} | & & | \\ U_1 & \dots & U_n \\ | & & | \end{bmatrix} \times \begin{bmatrix} \sigma_1 & & 0 & 0 & \dots & 0 \\ & \ddots & & 0 & \dots & 0 \\ 0 & & \sigma_n & 0 & \dots & 0 \end{bmatrix} \times \begin{bmatrix} - & V_1^{'} & - \\ & \vdots & \\ - & V_p^{'} & - \end{bmatrix}$$

$$= \underbrace{\sigma_1 u_1 v_1^T + ... + \sigma_r u_r v_r^T}_{=\tilde{U}\tilde{\Lambda}\tilde{V}^T} + ... + \sigma_p u_p v_p^T + 0$$

Une troncature sur nos données au rang r permet de réduire la dimension des données. Une approximation est donc obtenue sous la forme $X \approx \tilde{U}\tilde{\Lambda}\tilde{V}^T$.

En effet, d'après le théorème de Eckard-Young de 1936, cette approximation est la meilleure approximation au rang r de X pour la norme de Frobenius :

$$\underset{\tilde{X},\, \mathrm{rg}(\tilde{X})=r}{\arg\min} \; \|X - \tilde{X}\|_F = \tilde{U}\tilde{\Lambda}\tilde{V}^T$$

Les composantes étant orthogonales, le choix du rang r peut être défini suite à une ACP. Il est également possible d'imposer des contraintes de régularité sur la perte quadratique.

3.2 Par complétion de matrice

Le choix de la complétion de matrice est justifié lorsque la base de données comporte des zéros ainsi que des valeurs manquantes. Il ne faut plus remplacer les données manquantes par des zéros. L'utilisation de la *SVD* ou de la *NMF* devient alors abusive. Une solution est alors d'approximer une matrice très creuse, comportant de nombreuses valeurs manquantes, par une matrice de faible rang et ensuite d'utiliser la *NMF* ou la *SVD*.

4 Comparaison des méthodes

Méthode	**Principe**	**Avantages**	**Inconvénients**
NMF	Approximation de X par deux matrices non-négatives de faible rang Valeurs manquantes = 0	Optimum local donc convergence rapide W et H à entrée positive	X doit être semi-définie positive Compléxité algorithmique NP
SVD	Approximation de X par deux matrices orthogonales de faible rang Valeurs manquantes = 0	Optimum global Gère les valeurs négatives	Obtention de valeurs négatives Compléxité $n \times p$, obligation de tronquer pour grandes dimensions
Complétion de matrice	Approximation de X par une matrice de faible rang Valeurs manquantes $\neq 0$	Distingue les "0" des données manquantes	Complexité computationnelle Risque de surajustement

TABLE 2 – Comparaison des méthodes NMF, SVD et Complétion de matrice

5 Annexe

5.1 Problème de convexité

La méthode `NMF` ne conduit pas à une solution globale, mais se limite à fournir une solution locale. Nous allons le démontrer dans le cadre unidimensionnel [4]. Soient $X_{n\times p}, W_{n\times r}$ et $H_{r\times p}$ avec W, H matrices de coefficients positifs ou nuls tels que $X \approx WH$.
Nous voulons minimiser $\|X - WH\|^2$.
Cas $n = p = 1$, nous noterons donc les matrices en minuscule car ce sont des scalaires.
Posons

$$\Phi_x(w,h) = (x - wh)^2 = x^2 - 2xwh + w^2h^2$$

Calculons le gradient :

$$\nabla\Phi_x(w,h) = \begin{bmatrix} \dfrac{\partial\Phi_x(w,h)}{\partial w}(w,h) \\ \dfrac{\partial\Phi_x(w,h)}{\partial h}(w,h) \end{bmatrix} = \begin{bmatrix} 2wh^2 - 2xh \\ 2hw^2 - 2xw \end{bmatrix}$$

Calculons la matrice hessienne :

$$H(\Phi_x(w,h)) = \begin{bmatrix} \dfrac{\partial^2\Phi_x(w,h)}{\partial w^2}(w,h) & \dfrac{\partial^2\Phi_x(w,h)}{\partial w\partial h}(w,h) \\ \dfrac{\partial^2\Phi_x(w,h)}{\partial h\partial w}(w,h) & \dfrac{\partial^2\Phi_x(w,h)}{\partial h^2}(w,h) \end{bmatrix} = \begin{bmatrix} 2h^2 & 4wh - 2x \\ 4hw - 2x & 2w^2 \end{bmatrix}$$

Nous pouvons ainsi calculer la trace et le déterminant de la matrice hessienne :

$$Tr(H(\Phi_x(w,h))) = 2(h^2 + w^2) \geq 0 \quad \text{(Somme des valeurs propres)}$$

$$det(H(\Phi_x(w,h))) = \underbrace{h^2w^2 - (2hw - x)^2 > 0}_{\Leftrightarrow h^2w^2 > (2hw-x)^2} \quad \text{(Produit des valeurs propres)}$$

La condition de positivité du déterminant n'est pas vérifiée pour toutes les valeurs de h, w et x donc la matrice hessienne n'est pas semi-définie positive. Par conséquent, la fonction n'est pas convexe, et nous ne pouvons pas garantir l'existence d'un optimum global. Un optimum local sera suffisant.

5.2 Descente de gradient multiplicative pour la NMF

La méthode de la descente de gradient multiplicative est un algorithme particulièrement connu dans le domaine de la `NMF`, introduit par Lee et Seung (1999). L'objectif est de minimiser la fonction de perte suivante :

$$L(X, WH) = \|X - WH\|^2 = \sum_{i,j}\left(X_{ij} - \sum_k W_{ik}H_{kj}\right)^2$$

Néanmoins, la contrainte de non-négativité rend l'optimisation difficile à résoudre par une méthode de descente de gradient classique.

Une méthode populaire pour résoudre ce problème est donc la descente de gradient multiplicative. Cette méthode propose des règles de mise à jour pour les matrices W et H qui garantissent la non-négativité des matrices.

Les règles de mise à jour multiplicatives s'expriment comme suit : w_{ik} quantifie l'appétence du i-ème utilisateur pour le k-ème facteur latent, tandis que h_{kj} reflète la contribution du k-ème facteur latent à l'item j. Ces règles sont données par les expressions suivantes :

$$W_{ik} \leftarrow W_{ik} \underbrace{\left(\frac{(XH^T)_{ik}}{(WHH^T)_{ik}}\right)}_{\text{Rapports positifs}},$$

$$H_{kj} \leftarrow H_{kj} \overbrace{\left(\frac{(W^TX)_{kj}}{(W^TWH)_{kj}}\right)}.$$

Ces mises à jour sont répétées jusqu'à convergence vers un minimum local, c'est-à-dire jusqu'à ce que la variation de $L(X, WH)$ devienne négligeable. Les dénominateurs de ces expressions représentent des produits matriciels qui permettent de corriger les approximations et d'atteindre un meilleur ajustement à chaque itération. Ce schéma garantit que les matrices restent non négatives à chaque étape.

5.3 Résolution du problème

Nous allons illustrer la résolution de notre exemple par l'utilisation des trois méthodes mentionnées dans le corps du texte. Les codes sont fournis dans notre dépôt `git` [2].

Pour les recommandations, nous avons fait en sorte que ce ne soit pas le meilleur score qui soit choisi mais le meilleur score parmi les valeurs manquantes. En effet, l'intérêt de recommander une série à un utilisateur l'ayant déjà vu semble moindre.

5.3.1 En utilisant la *NMF*

En utilisant la fonction `NMF` du package scikitlearn en Python, avec les attributs suivant :

- $n\_components = r$: qui détermine le nombre de facteurs latents à extraire, le rang choisi par défaut 2.
- $init =' random'$: qui indique que les matrices W et H seront initialisé de manière aléatoire.
- $random\_state = 0$: permet de garantir la reproductibilité des résultats

Pour le choix de r (n_componant), il est possible d'afficher les résultats suivants :

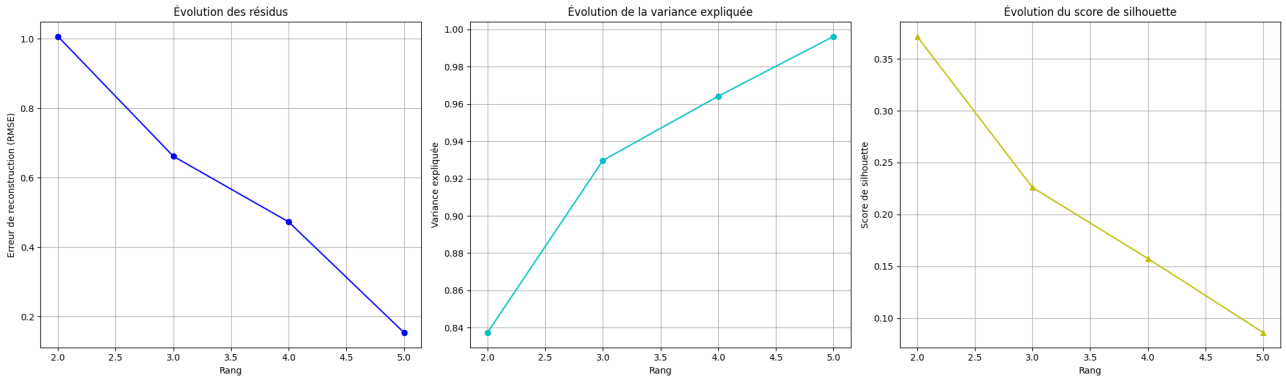

FIGURE 1 – Évolution des différents critères en fonction du rang des matrices de la factorisation par NMF

Dans la figure 1 nous retrouvons trois critères :

- les residus qui sont l'erreur entre la matrice d'origine et son approximation
- La variance expliquée qui indique la qualité de la décomposition et la capacité de NMF à représenter fidèlement les données originales
- Le score silhouette qui permet de quantifier la qualité du regroupement ou du partitionnement des données après factorisation.

Nous observons que, à mesure que le nombre de composantes augmente, les résidus diminuent, ce qui est prévisible.

En effet, l'augmentation du nombre de composantes facilite la recherche d'une combinaison linéaire de ces dernières qui se rapproche de plus en plus de X, entraînant ainsi une réduction des résidus.

La variance expliquée augmente donc aussi pour les mêmes raisons.

La balance entre le nombre de composantes et la généralisation est cruciale. Avec trop peu de composantes, la reconstruction est insuffisante. En revanche, un nombre excessif de composantes entraîne un risque de sur-ajustement, où le modèle capture non seulement les structures importantes, mais aussi le bruit, compromettant ainsi sa capacité à généraliser.

Le score silhouette diminue à mesure que le nombre de composantes augmente. Cela indique que la qualité du regroupement des points dans l'espace latent se dégrade.

Nous choisissons donc un $n\_componant = 2$

```
model=NMF(n_components=2, init='random', random_state=0)
W=model.fit_transform(X)
H=model.components_
X_nmf = np.dot (W , H )
```

Nous utilisons la méthode $fit\_transform$ pour obtenir la matrice W et H, avant de multiplier les deux matrices pour obtenir la matrice X reconstituée ci dessous :

$$\begin{bmatrix} 3 & 3 & \mathbf{0} & 2 & 3 \\ 3 & \mathbf{0} & \mathbf{0} & 3 & 1 \\ 1 & 2 & \mathbf{0} & 5 & 2 \\ \mathbf{0} & \mathbf{0} & 5 & 3 & 2 \\ \mathbf{0} & 3 & 4 & \mathbf{0} & \mathbf{0} \\ 3 & 1 & \mathbf{0} & 4 & \mathbf{0} \\ 2 & 3 & 4 & \mathbf{0} & 4 \end{bmatrix} \overset{\text{NMF}}{\Rightarrow} \begin{bmatrix} 2.25 & 1.64 & \mathbf{1.19} & 3.37 & 1.85 \\ 2.07 & \mathbf{1.03} & \mathbf{0.00} & 3.18 & 1.23 \\ 2.71 & 1.47 & \mathbf{0.28} & 4.15 & 1.73 \\ \mathbf{1.13} & \mathbf{2.21} & 3.79 & 1.44 & 2.3 \\ \mathbf{0.19} & 1.74 & 3.77 & \mathbf{0.00} & \mathbf{1.74} \\ 2.44 & 1.21 & \mathbf{0.00} & 3.74 & \mathbf{1.44} \\ 1.05 & 2.62 & 4.82 & \mathbf{1.24} & 2.70 \end{bmatrix}$$

Utilisateur	Nouvelle série recommandée	Score d'appréciation
Utilisateur 1	Dark	1.19
Utilisateur 2	The Witcher	1.03
Utilisateur 3	Dark	0.28
Utilisateur 4	The Witcher	2.21
Utilisateur 5	The Crown	1.74
Utilisateur 6	The Crown	1.45
Utilisateur 7	Gossip Girl	1.24

TABLE 3 – Nouvelle série recommandée et score d'appréciation par utilisateur avec NMF

Pour observer les différents groupes d'individus qui émergent en fonction des deux facteurs latents, nous allons analyser la matrice de consensus.

Celle-ci est obtenue à partir de la matrice des utilisateurs W à l'aide de la fonction `pairwise_distances` :

```
consensus_matrix = 1 - pairwise_distances(W, metric='cosine')
```

— `pairwise_distances` : fonction qui calcule les distances entre toutes les paires d'entrées dans une matrice ;
— `metric='cosine'` : les distances sont calculées avec la distance cosinus.

Chaque cellule dans la matrice représente à quel point deux utilisateurs sont similaires dans leurs évaluations, avec des valeurs comprises entre 0 (aucune similarité) et 1 (similarité maximale).

Nous obtenons ainsi la matrice de consensus suivante :

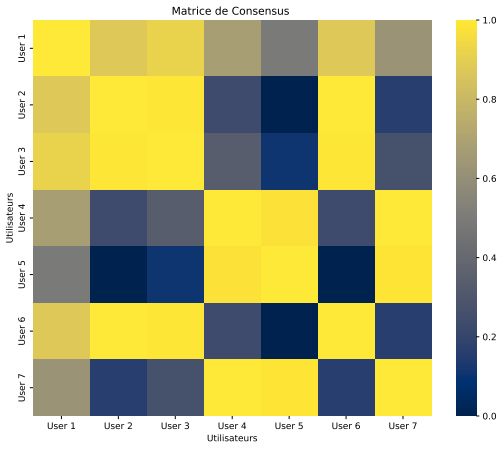

FIGURE 2 – Matrice de Consensus des utilisateurs par rapport aux deux facteurs latents

La matrice (Figure 2) révèle deux groupes distincts : les utilisateurs 1, 2, 3 et 6 partagent des préférences similaires, tandis que les utilisateurs 4, 5 et 7 se démarquent avec des évaluations nettement différentes. Ces groupes corroborent aussi avec les recommandations faites pour les utilisateurs.

5.3.2 En utilisant la *SVD*

En utilisant la librairie `SVD` en python, et notamment les commandes ci-dessous, on peut obtenir la matrice reconstruite X_{SVD} :

```
U, Sigma, Vt = np.linalg.svd(X, full_matrices=False)
X_svd = U @ np.diag(Sigma) @ Vt
```

Notre X projeté sur les deux premières composantes SVD :

$$\begin{bmatrix} 3 & 3 & \mathbf{0} & 2 & 3 \\ 3 & \mathbf{0} & \mathbf{0} & 3 & 1 \\ 1 & 2 & \mathbf{0} & 5 & 2 \\ \mathbf{0} & \mathbf{0} & 5 & 3 & 2 \\ \mathbf{0} & 3 & 4 & \mathbf{0} & \mathbf{0} \\ 3 & 1 & \mathbf{0} & 4 & \mathbf{0} \\ 2 & 3 & 4 & \mathbf{0} & 4 \end{bmatrix} \overset{\text{SVD}}{\Rightarrow} \begin{bmatrix} 3.00 & 3.00 & 6.85 \times 10^{-17} & 2.00 & 3.00 \\ 3.00 & -1.11 \times 10^{-15} & 1.22 \times 10^{-15} & 3.00 & 1.00 \\ 1.00 & 2.00 & 1.16 \times 10^{-15} & 5.00 & 2.00 \\ 1.38 \times 10^{-15} & 3.64 \times 10^{-15} & 5.00 & 3.00 & 2.00 \\ 1.65 \times 10^{-15} & 3.00 & 4.00 & -5.04 \times 10^{-15} & 3.09 \times 10^{-15} \\ 3.00 & 1.00 & 6.30 \times 10^{-16} & 4.00 & -1.72 \times 10^{-15} \\ 2.00 & 3.00 & 4.00 & -4.09 \times 10^{-15} & 4.00 \end{bmatrix}$$

Utilisateur	Nouvelle série recommandée	Score d'appréciation
Utilisateur 1	Dark	6.85e-17
Utilisateur 2	Dark	1.22e-15
Utilisateur 3	Dark	1.16e-15
Utilisateur 4	The Witcher	3.64e-15
Utilisateur 5	The Crown	3.09e-15
Utilisateur 6	Dark	6.30e-16
Utilisateur 7	Gossip Girl	-4.09e-15

TABLE 4 – Nouvelle série recommandée et score d'appréciation par utilisateur avec SVD

De plus, nous constatons que la différence entre la matrice X et la matrice X_{SVD} apparaît au niveau des coefficients initialement nuls. Ces recommandations ne semblent pas satisfaisante ce qui est sûrement dû à nos données.

5.3.3 En utilisant la complétion de matrice

Supposons maintenant qu'au lieu de remplacer les valeurs manquantes dans la matrices, nous voulions d'abord l'approcher par une matrice similaire :

Il nous faudrait donc utiliser la complétion de matrice. Nous pouvons utiliser la fonction `SoftImpute` de la librairie `fancyimpute` (méthode itérative qui utilise la *nuclear-norm regularization*) :

```
X_chap = SoftImpute(max_iters=100, verbose=0).fit_transform(X)
```

Ainsi nous obtenons les transformations de matrices suivantes :

$$\begin{bmatrix} 3.0 & 3.0 & \mathbf{NA} & 2.0 & 3.0 \\ 3.0 & \mathbf{NA} & \mathbf{NA} & 3.0 & 1.0 \\ 1.0 & 2.0 & \mathbf{NA} & 5.0 & 2.0 \\ \mathbf{NA} & \mathbf{NA} & 5.0 & 3.0 & 2.0 \\ \mathbf{NA} & 3.0 & 4.0 & \mathbf{NA} & \mathbf{NA} \\ 3.0 & 1.0 & \mathbf{NA} & 4.0 & \mathbf{NA} \\ 2.0 & 3.0 & 4.0 & \mathbf{NA} & 4.0 \end{bmatrix} \overset{\text{Complétion}}{\Rightarrow} \begin{bmatrix} 3.00 & 3.00 & \mathbf{2.92} & 2.00 & 3.00 \\ 3.00 & \mathbf{1.02} & \mathbf{0.60} & 3.00 & 1.00 \\ 1.00 & 2.00 & \mathbf{3.47} & 5.00 & 2.00 \\ \mathbf{0.38} & \mathbf{2.90} & 5.00 & 3.00 & 2.00 \\ \mathbf{1.14} & 3.00 & 4.00 & \mathbf{1.95} & \mathbf{2.47} \\ 3.00 & 1.00 & \mathbf{0.90} & 4.00 & \mathbf{1.29} \\ 2.00 & 3.00 & 4.00 & \mathbf{2.55} & 4.00 \end{bmatrix} \overset{\text{NMF}}{\Rightarrow} \begin{bmatrix} 1.90 & 2.44 & 3.26 & 3.17 & 2.43 \\ 2.67 & 0.85 & 0.56 & 3.28 & 1.11 \\ 2.32 & 2.50 & 3.23 & 3.65 & 2.54 \\ 0.86 & 3.09 & 4.57 & 2.33 & 2.88 \\ 0.87 & 2.75 & 4.03 & 2.19 & 2.57 \\ 3.11 & 1.09 & 0.82 & 3.87 & 1.39 \\ 1.60 & 3.09 & 4.36 & 3.14 & 2.98 \end{bmatrix}$$

On obtient alors les recommandations suivantes :

Utilisateur	Nouvelle série recommandée	Score d'appréciation
Utilisateur 1	Dark	3.26
Utilisateur 2	The Witcher	0.85
Utilisateur 3	Dark	3.23
Utilisateur 4	The Witcher	3.09
Utilisateur 5	The Crown	2.57
Utilisateur 6	The Crown	1.39
Utilisateur 7	Gossip Girl	3.14

TABLE 5 – Nouvelle série recommandée et scores d'appréciation par utilisateur

On retrouve souvent les séries Dark et Gossip Girl qui sont les séries les mieux notées et pas nécessairement les séries les plus vues.

5.3.4 Comparaison des résultats

La méthode par complétion de matrice se distingue par son approche vis-à-vis des données manquantes, nous allons donc comparer seulement les résultats obtenus avec la NMF et la SVD.

Méthode	**Erreur de reconstitution**
NMF	1.01
SVD	2.44×10^{-15}

TABLE 6 – Erreur de reconstitution selon la méthode

L'erreur de reconstitution de la méthode SVD est bien meilleure seulement son utilisation semble peu intéressante pour nos données.

5.4 Application à la base de données *Netflix Movie Rating Dataset*[1]

Ce jeu de données contient trois colonnes et un total de 17 337 458 lignes, chacune représentant une évaluation d'un film par un utilisateur.

- **User_ID** : Un numéro unique qui identifie chaque utilisateur ayant laissé une évaluation.
- **Rating** : La note donnée par l'utilisateur au film, sur une échelle de 1 à 5, avec 5 étant la meilleure appréciation.
- **Movie_ID** : Un identifiant unique pour chaque film évalué.

Voici un extrait de notre base de donnée transformées pour notre problème de recommandation :

User_ID	**Movie_3**	**Movie_8**	**Movie_28**	**Movie_30**	**Movie_32**	...
6	NaN	NaN	NaN	3.0	NaN	...
7	NaN	5.0	4.0	5.0	NaN	...
79	NaN	NaN	NaN	3.0	NaN	...
97	NaN	NaN	NaN	NaN	NaN	...
134	NaN	NaN	5.0	NaN	NaN	...
⋮	⋮	⋮	⋮	⋮	⋮	⋱

TABLE 7 – Extrait de la base de donnée avec un échantillon d'évaluations de films

Il est important de noter que dans ce jeu de données, plus de 90% des valeurs sont manquantes ce qui rend la matrice très sparse.

5.4.1 Application de la NMF

Dans un premier temps, nous allons remplacer les données manquantes par "0", puis appliquer la NMF.

Nous obtenons la matrice reconstitué suivante (c'est un extrait une nouvelle fois), obtenue avec un nombre de composantes par défaut de 2.

$$X_{\text{NMF}} = \begin{bmatrix} 0.05 & 0.22 & 0.09 & 0.12 & 0.33 & 0.12 \\ 0.09 & 0.37 & 0.14 & 0.19 & 0.55 & 0.18 \\ 0.01 & 0.08 & 0.13 & 0.22 & 0.12 & 0.21 \\ 0.05 & 0.20 & 0.06 & 0.08 & 0.29 & 0.07 \\ 0.00 & 0.02 & 0.16 & 0.28 & 0.04 & 0.27 \\ 0.02 & 0.10 & 0.07 & 0.11 & 0.14 & 0.10 \end{bmatrix}$$

Nous obtenons grâce à la `NFM` les recommandations suivante :

Film recommandé	Nombre de recommandations	Proportion (%)
2452	15 907	11.09
1905	15 596	10.87
4306	14 139	9.86

TABLE 8 – Films les plus recommandés

On retrouve une erreur de reconstitution de 0.9096.

Le paramètre *n_components* dans la fonction `NMF` étant le choix du rang, nous avons opté pour la validation croisée K-fold pour le déterminer.

Pour cela, les données sont divisées en 5 plis. À chaque itération, 4 plis sont utilisés pour l'entraînement et 1 pli pour le test, et comparer l'erreur moyenne de reconstruction sur l'échantillon de test et d'entraînement. Nous obtenons les résultats suivants :

n_components	**Erreur moyenne (train)**	**Erreur moyenne (test)**
2	0.8280	0.8279
5	0.7650	0.7650
10	0.7271	0.7272
50	0.6126	0.6125
100	0.5281	0.5279
200	0.3955	0.3950
300	0.3035	0.3022
400	0.2425	0.2410

TABLE 9 – Erreur moyenne de reconstruction pour différents n_components

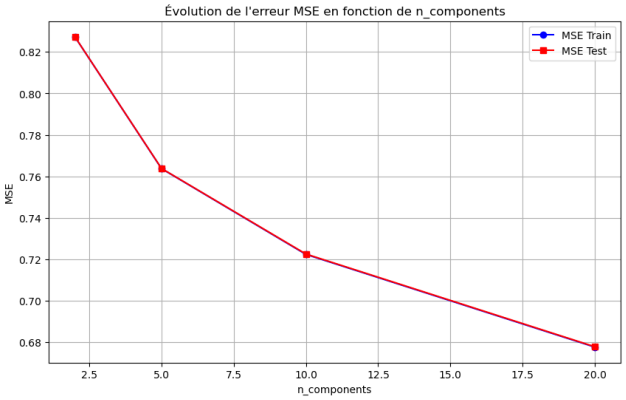

FIGURE 3 – Evolution des erreurs MSE en fonction du n_component

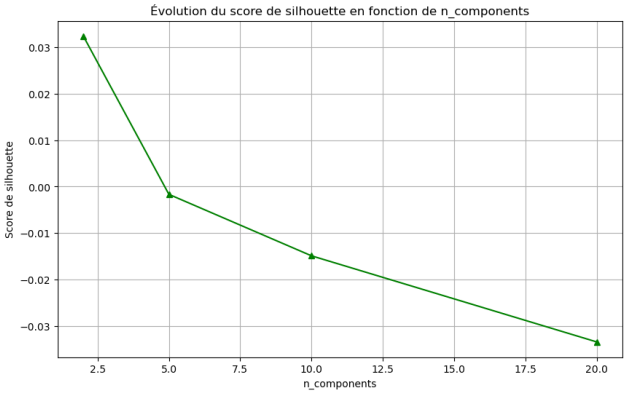

FIGURE 4 – Evolution de la silhouette en fonction de n_component

Nous constatons que les erreurs moyennes de reconstruction de la matrice X sur l'échantillon de test et d'entraînement sont sensiblement proche, et la variation de l'erreur reste négligeable . Néanmoins,à mesure que le nombre de composants augmente, le temps d'exécution s'allonge, comme le montre le tableau ci-dessous :

Nombre de composantes (r)	**Temps d'exécution (minutes)**
10	2.51
20	3.17
30	4.52
50	6,92
200	43.75

TABLE 10 – Temps d'exécution du NFM en fonction du nombre de composants

Nous observons que le temps d'exécution augmente rapidement. Dans cette étude, nous privilégierons un temps d'exécution plus court, même si cela implique une erreur plus élevée.

5.5 Application de la SVD

Après avoir essayé d'utiliser la SVD, nous avons été confronté à un problème de taille (de matrice). Par consequent la méthode usuelle de SVD est inutilisable, nous avons donc décidé d'utiliser la méthode `TruncatedSVD` de `scikit-learn` :

```
svd = TruncatedSVD(n_components=2, random_state=42)
svd.fit(X)
X_svd = svd.transform(X)
X_reconstructed = X_svd @ svd.components_
```

Film recommandé	Nombre de recommandations	Proportion (%)
2452	15 796	11.01
1905	15 399	10.73
4306	13 930	9.71

TABLE 11 – Films les plus recommandés avec la méthode SVD tronquée

On obtient ainsi des proportions similaires à celles de la NMF, bien qu'elles soient tout de même différentes. L'erreur de reconstitution de la matrice est de 0.9089 ce qui est un peu mieux que 0.9096 de notre NMF précédente.

Références

[1] Rishit Javia. *Netflix Movie Rating Dataset-Kaggle*. https://www.kaggle.com/datasets/rishitjavia/netflix-movie-rating-dataset.

[2] Q. Festor P. Dias, A. Lahjiouj. *Dépôt git des codes*. https://github.com/Qufst/Filtrage_collaboratif.

[3] Scikit-learn. *Documentation Module `NMF`*. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html.

[4] Ahmad Varasteh. *Non-Negative Matrix Factorization (NMF) | Multiplicative Update Rules By Lee And Seung*. https://www.youtube.com/watch?v=o4pPTwsd-5M&list=LL&index=5.

[5] Wikistat. *NMF Factorisation par matrices non négatives*. http://wikistat.fr/pdf/st-m-explo-nmf.pdf.

[6] Wikistat. *Scénario : Recommandation par Factorisation ou Complétion* . https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-scenar-explo7-nmf.pdf.