Factorisation matricielle non-négative et classification d'images

MOHAMED BEN HAMDOUNE[1] et YANNIS TANNIER[1]

Université Paris-Descartes, 12 Rue de l'École de Médecine, 75006 Paris, France,
mohamed.ben_hamdoune@etu.parisdescartes.fr
yannis.tannier@etu.parisdescartes.fr
https://www.univ-paris5.fr/

Résumé La classification automatique ou *clustering* consiste à partionner un ensemble d'objets (instances) décrit par un ensemble de variables en groupes (classes) homogènes. Avec l'avènement du *BigData* et de la science des données, le clustering est devenu une tâche très importante dans divers domaines dont l'imagerie. Les images sont des données très répandues notamment sur le web et les réseaux sociaux (Instagram, Pinterest, Flickr, Google,etc...).Le but sera de proposer un systéme de classification pour des images provenant de divers bases de données (photos, peintures, bandes dessinés, etc). La factorisation matricielle non-négative permet d'approximer une matrice de données positive par le produit de deux matrices de dimensions inférieures et positives. Par sa simplicité, cette méthode est devenue populaire et est utilisée à la fois dans la réduction de la dimension et également dans la classification automatique (clustering) en un nombre de classes k fixé par l'utilisateur.

Keywords : Apprentissage Non Supervisé, Spherical K-means, K-means, R, NMF, Python, Scikit-Learn, Nimfa, People-Art Dataset

Table des matières

Factorisation matricielle non-négative et classification d'images 1
MOHAMED BEN HAMDOUNE et YANNIS TANNIER
1 Introduction 4
1.1 Contexte et motivations 4
1.2 Contribution et organisation du rapport 4
2 Construction de la matrice numérique 4
2.1 Construction de la matrice 4
2.2 Les filtres numériques 5
2.3 Normalisation 6
3 Clustering 6
3.1 K-means et Spherical K-means 6
3.2 Déterminer le nombre de cluster k 7
3.3 Analyse et résultats 7
4 La factorisation matricelle non-négative 13
4.1 Présentation de la NMF 13
4.2 Différentes méthodes de NMF 14
5 Méthodes d'initialisation de la NMF 15
5.1 Aléatoire 15
5.2 K-means 15
5.3 Spherical K-Means 16
5.4 Double décomposition en valeurs singulières non négative 17
5.5 Analyse en Composantes Principales 18
5.6 Analyse en composantes Indépendantes 20
6 Étude comparative 20
6.1 Estimation du rang de la NMF 20
6.2 Évaluation des données 21
6.3 Résultats 24
7 Conclusion 25

Remerciements

Nous souhaitons adresser nos remerciements aux personnes qui nous ont aidé dans la réalisation de ce rapport.
En premier lieu, nous remercions M. Nadif et M. Labiad qui nous ont permis d'accéder à leur publications.
Nous remercions aussi M. Salem, notre responsable du master, pour son modèle de rédaction ainsi que ces informations durant cette année scolaire.
Enfin, nous souhaitions particulièrement remercier M.Febressy pour sa disponibilité tout au long de ce semestre ainsi que sa précieuse aide à la relecture et à la correction de notre rapport.

1 Introduction

1.1 Contexte et motivations

Dans le cadre de notre cursus scolaire, nous sommes amenés à réaliser un TER **(Travail d'étude et de recherches)**. Notre sujet est intitulé : **"Factorisation matricielle non-négative et classification d'images"**. Au cours de ce semestre nous avons étudié la NMF (**Non-Negative Matrix Factorization**) et aussi sa particularité à faire de la classification automatique (*clustering*) dans certains cas comme cela sera décrit dans ce rapport. La NMF est un sujet très intéressant car elle couvre de nombreux domaines : imagerie, reconnaissance de formes, fouille de textes. La recommandation est également un domaine où la NMF excelle et a permis de nombreuses avancées. Enfin, la NMF peut être une technique très puissante de réduction de dimension adaptée aux matrices creuses contenant des données positives, par exemple des occurrences ou dénombrements de mots.

1.2 Contribution et organisation du rapport

Le déroulement de ce **TER** suit finalement une logique qui est propre aux systèmes lors d'une modélisation de classification automatique. Tout d'abord comme écrit dans la première consigne, nous avons réalisé la matrice (Section 2.1), c'est-à-dire converti les images de départ disponibles sur *Github* [ici] dans un format qui permet d'être applicable dans une NMF. Les filtres (Section 2.2) disponibles les plus courants sont binaires, niveaux de gris, RGB. Pour ce TER, nous avons décidé d'appliquer un filtre greyscale. De plus nous discuterons sur la normalisation (Section 2.3) de cette matrice pour ensuite passer à la section suivante sur les méthodes d'analyse. Nous introduirons la notion de NMF (Section 4.1) comme méthode de factorisation matricielle non-négative et les deux méthodes de partitionnement utilisées que sont *K-means* et *Spherical K-means* permettant de faire du clustering avec la NMF. À partir de tout cela, nous réaliserons une étude comparative (Section 6) avec plusieurs combinaisons d'initialisation pour réaliser la NMF suivant un intervalle donné par les résultats de *K-means* et *Spherical K-means*. Finalement, nous conclurons (Section 7) sur ce projet avec les idées que nous avions et que nous n'avons pas pu réaliser, ainsi que notre avis sur ce sujet de TER.

2 Construction de la matrice numérique

2.1 Construction de la matrice

Le jeu de données qui nous a été donné à pour thématique *l'art* et plus précisément une collection d'images couvrant une grande variété de styles artistiques. Cette collection [1] regroupe en tout 4778 images regroupées dans 43 dossiers (catégories). Ces images ont été rassemblé dans le but de faire de la *cross-depiction* (La représentation croisée est la reconnaissance et la synthèse des objets qu'ils

soient photographiés, peints, dessinés, etc. C'est un problème important mais sous-étudié. Le but étant de reproduire artificiellement la capacité humaine de reconnaître et de représenter des objets). Les images sont diverses et variées et ils sont sous *"copyright"*. Elles sont essentiellement mises à disposition pour l'exploration de données dans un cadre de recherche non-commercial. Afin d'avoir une uniformisation entre les images pour effectuer l'analyse, nous avons d'abord redimensionné la taille de chaque image en 25x25. Pour cela nous avons utilisé **Pillow**, une librairie Python [2] qui permet de faire de la manipulation d'image. Toutes les images redimensionnées ont été enregistrées dans un dossier "resized" afin de pouvoir être réutilisées plus tard. Enfin le label (la catégorie de l'image) a également été récupéré.

```
images = glob.glob("PeopleArt/JPEGImages/*/*")
i = 1
for image in images:
    label = image.split("/")[2].lower()
    with open(image, 'r+b') as f:
        img = Image.open(f)
        img = resizeimage.resize_cover(img, [25, 25])
        img.save("PeopleArt/resized/"+label+"_"+str(i)+'.jpg',
    img.format)
    i += 1
```

2.2 Les filtres numériques

Une fois les images redimensionnées, nous avons appliqué diffèrents filtres. Pour cela, nous avons utilisé la librairie **Pillow** en *Python* qui propose plusieurs filtres :
- Le filtre niveau gris (*"greyscale"*) qui retourne des valeurs $x \in [0, 255]$.
- Le filtre binaire ("*binarisation*") des images retournant des valeurs qui sont soit 0 ou 1 suivant un seuil *"threshold"*.
- Le filtre *RGB* (Red-Green-Blue), qui retourne une matrice en 3 dimensions représentant chaque couleur. Dans le cadre de ce rapport, nous avons choisi le filtre greyscale par soucis de temps de calcul assez conséquent.

Pour créer notre matrice, nous avons donc converti chaque image en greyscale. Pillow nous retourne ainsi une matrice 25x25 comprenant des valeurs entre 0 et 255. Nous avons ensuite transformé cette matrice en vecteur de taille 625 et nous avons concaténé tout les vecteurs d'images afin de former notre matrice. La matrice finale comprend ainsi 4778 lignes et 625 colonnes.

```
images = shuffle(glob.glob(folder+"*"))
labels = np.array([])
X_data = np.array([])

for image in images:
    labels = np.append(labels,image.split("/")[3].split("_")
    [0].lower())
    image = Image.open(image).convert('L')
```

```
    arr = np.array(image)
    arr = arr.reshape(1,size*size)
    X_data = np.concatenate((X_data, arr), axis = 0)
np.savetxt('gs.csv', X_data, fmt = '%3d', delimiter = ',')
```

2.3 Normalisation

Une fois notre matrice crée, nous avons appliqué une normalisation en ligne. Cette normalisation consiste à appliquer une normalisation L_2 afin de mettre le vecteur à l'échelle et ainsi obtenir un vecteur unitaire. Un vecteur unitaire correspond à un vecteur dont la norme est égale à 1.
Ainsi, étant donnée notre matrice X, ou les lignes correspondent aux observations et les colonnes aux caractéristiques, nous avons divisé chaque vecteur $x \in X$ par la norme L_2 du vecteur :

$$L_2 = ||x||_2 = \sqrt{\sum_i x_i^2}.$$

Cela peut être fait très facilement en Python grâce à la librairie **Scikit-Learn** [3] :

```
from sklearn import preprocessing
X_normalized = preprocessing.normalize(datas, norm = 'l2')
```

3 Clustering

Le partitionnement de données (ou clustering) est une des méthodes d'analyse des données qui vise à diviser un ensemble de données en différents "groupes" homogènes, c'est-à-dire que les données de chaque sous-ensemble partagent des caractéristiques communes, qui correspondent le plus souvent à des critères de proximité. Dans ce rapport nous verrons deux méthodes de clustering que nous avons utilisé : K-means et Spherical K-means.

3.1 K-means et Spherical K-means

K-means est une méthode de partitionnement de données qui, à partir d'un ensemble de points, va pouvoir déterminer pour un nombre de classes fixé une répartition des points qui minimise un critère appelé *inertie* ou variance *intra-classe*.
Plus formellement étant donnée k, nous allons donc chercher à répartir les points $x_1, x_2, .., x_n$ en k groupes $C_1, C_2, , C_K$ de telle sorte que :

$$\sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{x \in C_k} ||x - (\frac{1}{|C_k|} \sum_{x \in C_k} x)||^2$$

soit le plus minimale.

Spherical K-means contrairement à K-means qui cherche à minimiser une distance euclidienne, on va plutôt définir le centre de chaque groupe de sorte à uniformiser et minimiser l'angle entre les composants.

3.2 Déterminer le nombre de cluster *k*

Comme nous l'avons vu précédemment, il est essentiel d'indiquer le bon nombre de cluster à K-means et Spherical K-means. Cependant déterminer le bon nombre de cluster est une tâche qui peut s'avérer très compliquée. Pour cela nous avons choisi 2 critères afin d'évaluer la qualité du partitionnement : la méthode **du Coude** et le coefficient de **Silhouette** (Des packages comme *"facto extra"* et *"Nbclust"* facilitent la possibilité d'effectuer un graphique sur les méthodes permettant d'avoir une idée sur le nombre de classe avec R [4]).

- La méthode du Coude : L'idée derrière cette méthode est d'exécuter K-Means sur l'ensemble des données pour un nombre de cluster de valeurs k (par exemple $k = 1$ à 10) et de calculer pour chaque rang k la somme des erreurs au carré (SSE). Ainsi on peut choisir un nombre de cluster k de telle sorte que l'ajout d'un autre cluster ne donne pas une meilleure modélisation des données.
Cette méthode s'appelle aussi la méthode du *"coude"* car si on affiche les informations sur un graphe, nous pouvons choisir le rang k au niveau où la courbe forme un *"coude"*.

- Le coefficient de Silhouette : Cette méthode est une mesure de la similitude entre un objet et son propre cluster (cohésion) par rapport aux autres clusters (séparations). Autrement dit, le coefficient de silhouette $s(x)$ permet d'évaluer, pour un point x donné, si ce point appartient au "bon" cluster : est-il proche des points du cluster auquel il appartient ? Est-il loin des autres points ?
La valeur de la silhouette est compris entre -1 et 1 où plus la valeur est proche de 1 et plus l'assignation de x à son cluster est satisfaisante.

3.3 Analyse et résultats

Afin de déterminer un intervalle de rang, nous avons appliqué la formule :

$$R = \frac{m * n}{m + n}$$

où m correspond au nombre de lignes de notre matrice et n le nombre de colonne.

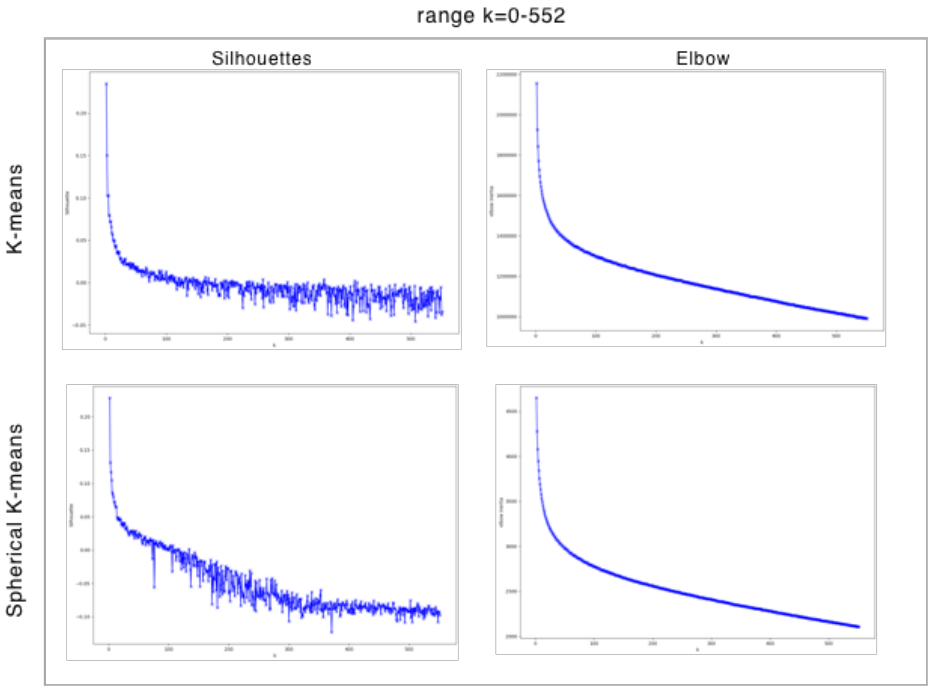


Figure 1. Mesures de K-means et Spherical K-means sur les rangs de 0 à 552.

Les résultats en Figure 1 nous montrent qu'une analyse précise de la silhouette et de la méthode du coude est assez compliqué avec un intervalle de rang aussi grand. Cependant nous pouvons déjà voir qu'entre 20 et 70, les graphiques semblent intéressant pour K-means et Spherical K-means. De plus on sait également que nos images sont classées en 43 catégories (dossier), cela nous donne une première estimation du rang que l'on pourrait trouver. Nous nous y sommes donc intéressées et avons refait les analyses avec cet nouvel intervalle.

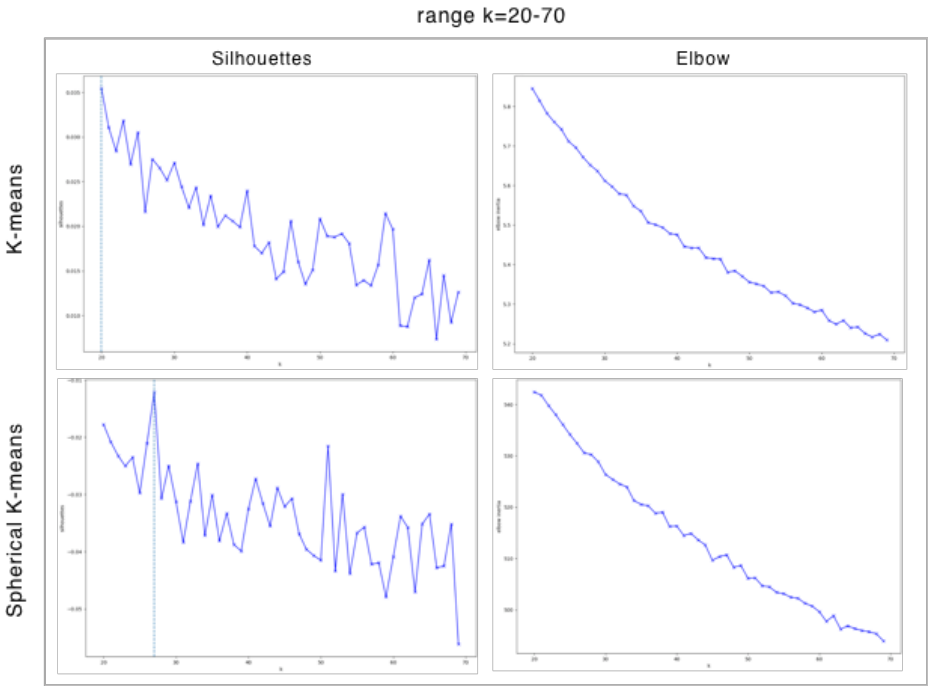


Figure 2. Mesures de K-Means et Spherical K-means sur les rangs de 0 à 552.

Ainsi, nous pouvons voir que d'après la Figure 2, la silhouette pour K-means semble meilleur pour les rangs 20, 23, 40, 59 et 27, 33, 41, 51 pour Spherical K-Means. Toutefois ces résultats sont à nuancer car la valeur la plus haute obtenue pour la silhouette est de 0.036, ce qui est très faible et nous donne une première indication quant à la qualité de partition. Après avoir déterminé les rangs pour K-means et Spherical K-means, nous avons voulu comparer les résultats en calculant L'ARI et la NMI pour chacune des partitions obtenues avec la partition original, correspondant à la partition des catégories GitHub. En effet, ces catégories nous donnent une information sur un potentiel regroupement des images, ainsi nous avons voulu voir si ce regroupement correspond aux données.

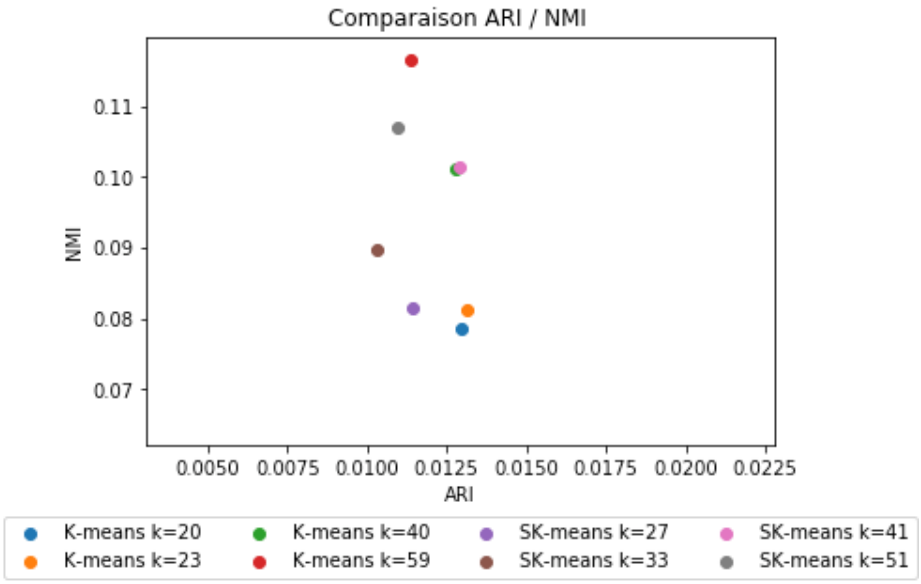


Figure 3. Comparaison ARI/NMI sur les différents rangs.

Les résultats en Figure 3 nous montrent que les méthodes et les rangs se valent plus ou moins tous. Les résultats tournent autour de 0.08 et 0.012 pour la NMI et 0.01 et 0.15 pour l'ARI. Ces résultats sont faibles et vont dans le sens des résultats obtenus avec la silhouette.

Nous avons voulu aller plus loin en gardant cette information à priori sur le nombre de classe. Ainsi nous avons voulu tester les résultats obtenus avec les différentes méthodes et initialisation de la NMF avec $k = 43$. Nous avons calculé l'ARI/NMI et comparer avec la partition original (GitHub).

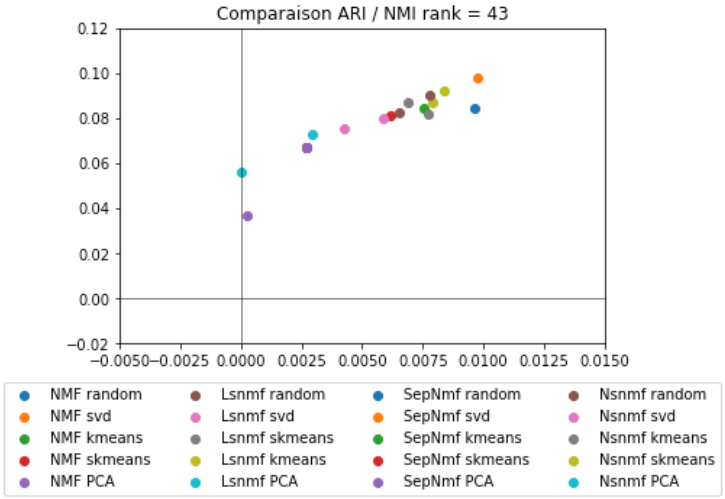


Figure 4. Récapitulatif de l'ARI/NMI pour le rang 43.

Sans surprise les résultats sont très faible, confirmant ce que l'on a vu plus haut. Cependant nous pouvons voir que SepNmf avec une initialisation SVD et SepNmf avec une initialisation aléatoire s'en sortent le mieux, SepNmf étant une méthode de NMF particulièrement bien adapté pour les pixels.

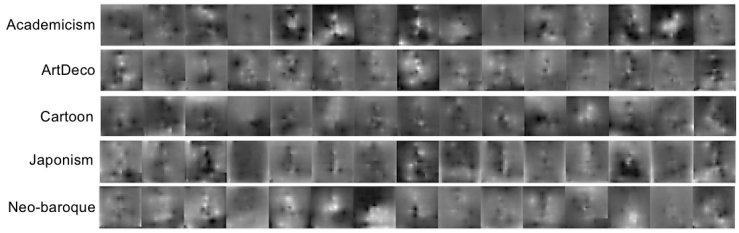


Figure 5. Récapitulatif sur la recomposition des images.

Enfin, la visualisation d'un échantillon d'image par catégorie nous montre bien que la NMF a du mal à extraire des caractéristiques spécifiques pour chaque classe.

Afin de renforcer nos résultats obtenus, nous avons par la suite utilisé une méthode de classification supervisé, la SVM, avec les matrices des images en features et la catégorie des images en labels.

```python
from sklearn import svm
from sklearn.model_selection import cross_val_score

clf = svm.SVC()
scores = cross_val_score(clf, datas, labels, cv=10)

print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.
    std() * 2))
# Accuracy: 0.05 (+/- 0.02)
```

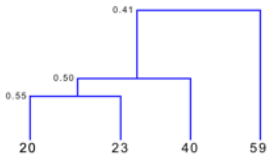
Nous obtenons une accuracy de 5% via une validation croisée.

Ce dernier résultat nous pousse à conclure que la catégorisation originale est trop dispersée et qu'il n'existe sans doute pas de pattern permettant de différencier ces catégories. Nous laissons donc tomber cette catégorisation et nous allons essayer de trouver la meilleure partition sur ce jeu de donnée sans considérer la catégorisation originale répartie sur les 43 dossiers.

C'est pour cette raison que nous avons calculé l'ARI et la NMI en croisant toutes les partitions obtenues, c'est-à-dire pour les rang 20, 23, 40, 59 pour K-means et 27, 33, 41, 51 pour Spherical K-means. Pour cela, nous avons créé un dendrogramme pour savoir qu'elles étaient les partitions les plus proches.

Enfin, on peut voir que pour K-means, les partitions 20, 23 et 40 sont relativement proche. Nous choisissons donc de garder uniquement la partition 20 et 59. Pour Spherical K-Means, toutes les partitions se valent, nous gardons donc que la partition 27.

K-means	**20**	**23**	**40**	**59**
20	1	0.55	0.42	0.38
23		1	0.50	0.38
40			1	0.41
59				1



Sk-means	**27**	**33**	**41**	**51**
27	1	0.50	0.40	0.46
33		1	0.43	0.42
41			1	0.41
51				1

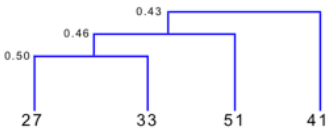


Figure 6. Récapitulatif pour K-Means et Spherical K-Means avec leur arbre hiérarchique.

4 La factorisation matricelle non-négative

4.1 Présentation de la NMF

Cette section donne une définition formelle des problèmes de factorisation matricielle non-négative [5] et définit les notations utilisées dans le cadre de ce TER.

Soit X une matrice non-négative $n \times p$, (i.e avec $x_{ij} \geq 0$, telle que $X \geq 0$), et $r > 0$ un entier positif. La factorisation matricielle non-négative consiste à trouver une approximation

$$X \approx WH \ , \tag{1}$$

où W, H sont $n \times r$ et $r \times p$ matrices non-négativs respectivement.

En pratique, le rang de factorisation r est souvent choisi de telle sorte que $r \ll \min(n, p)$. L'objectif derrière ce choix est de résumer et diviser l'information contenue dans X en facteurs r : les colonnes de W.

Selon le domaine d'application, ces facteurs ont des noms différents : images de base, métagènes, signaux source. Dans ce rapport, nous utilisons de manière équivalente et alternative les termes *matrice des bases* pour faire référence à la matrice W, et *matrice de coefficients* pour se référer à la matrice H. La principale approche du **NMF** [6] consiste à estimer les matrices W et H comme un minimum local :

$$\min_{W,H \geq 0} \underbrace{[D(X, WH) + R(W, H)]}_{=F(W,H)} \tag{2}$$

où

- D est une fonction de perte qui mesure la qualité de l'approximation. Les fonctions de perte communes sont basées soit sur la distance de Frobenius

$$D : A, B \mapsto \frac{Tr(AB^t)}{2} = \frac{1}{2} \sum_{ij} (a_{ij} - b_{ij})^2,$$

  ou la divergence Kullback-Leibler.

$$D : A, B \mapsto KL(A||B) = \sum_{i,j} a_{ij} \log \frac{a_{ij}}{b_{ij}} - a_{ij} + b_{ij}.$$

- R est une fonction de régularisation facultative, définie pour appliquer les propriétés sur les matrices W et H, telles que la *smoothness* ou la *sparsité* [7].

Un paramétre critique dans la NMF est le rang de la factorisation r. Il définit le nombre de variables utilisés pour approcher la matrice cible. Étant donné une méthode NMF et la matrice cible, une façon courante de décider de r est d'essayer différentes valeurs, de calculer une mesure de qualité des résultats et de choisir la meilleure valeur en fonction de ces critères de qualité.

4.2 Différentes méthodes de NMF

Il existe plusieurs algorithmes permettant de réaliser une NMF (une librairie en Python du nom de Nimfa [8] implémente plusieurs modèles de la NMF avec plusieurs algortihmes et plusieurs possibilités d'initialisation), généralement ces algorithmes fonctionnent de manière itérative, en construisant une séquence de matrice (W, H) qui va tenter de réduire à chaque itération une *fonction coût*. Il existe également des variantes de la fonction et d'autres techniques d'optimisation tel que la descente de gradient permettant de calculer et mettre à jour plus efficacement H et W.
Dans la suite de ce rapport, nous expliquerons les diverses méthodes que nous avons utilisé :

- **Brunet** : La première méthode utilisée est l'algorithme de brunet [9], correspondant à la méthode Standard de NMF, basée sur la divergence de Kullback-Leibler. Cette méthode multiplicative est une amélioration de l'algorithme initialement proposé par Lee [10] et permettant de réduire le *numerical underflow*.

$$
\begin{aligned}
H_{kj} &\leftarrow H_{kj} \frac{\left(\sum_l \frac{W_{lk} V_{lj}}{(WH)_{lj}}\right)}{\sum_l W_{lk}} \\
W_{ik} &\leftarrow W_{ik} \frac{\sum_l [H_{kl} A_{il} / (WH)_{il}]}{\sum_l H_{kl}}
\end{aligned}
$$

- **Lsnmf** : Méthode de factorisation de la matrice des moindres carrés non-négatif utilisant une méthode de gradient (optimisation contrainte) pour chaque sous-problème. Cette méthode proposée par Lin en 2007 [11] a l'avantage de converger beaucoup plus rapidement que les approches courantes de mise à jour multiplicatives.

- **Nsmf** : Nonsmooth Non negative Matrix Factorisation. Cette méthode proposée par Montano en 2006 est une version modifiée de la méthode de Lee permettant d'avoir de meilleur résultat dans le cas de matrice dite "*sparse*" [7].

- **SepNmf** : Separable Nonnegative Matrix Factorization. Cette méthode a été proposée initialement par Donoho et Stodden en 2003 puis améliorée par Arora en 2012. La Sepnmf est particulièrement adaptée pour l'imagerie hyperspectrale et l'analyse de document.

5 Méthodes d'initialisation de la NMF

5.1 Aléatoire

La plupart des fonctions coûts de la NMF ne sont pas convexes et sont donc sensibles à l'initialisation des matrices de facteurs W et H comme expliqué par N. Gillis [6]. Une bonne stratégie d'initialisation peut considérablement atténuer le problème de convergence des algorithmes NMF. Ici, le terme *"bon"* se réfère à la stratégie d'initialisation qui conduit à une réduction rapide de l'erreur pour la fonction coût avec une vitesse de convergence rapide. Dans la littérature, l'initialisation aléatoire a été couramment utilisée, par exemple dans le travail de Lee et Seung [10], où l'on doit exécuter plusieurs fois un l'algorithme avec différentes matrices initiales et choisir la meilleure solution. Cependant, choisir les matrices initiales au hasard donne souvent une mauvaise solution, nous parlerons aussi [12] d'une approche d'initialisation structurée pour la NMF où ils ont initialisé les matrices factorisées en utilisant les centroïdes de classes provenant d'un Spherical K-Means en exemple. Pour de nombreuses méthodes de clustering, leurs fonctions coût sont non convexes et leur optimisation implique généralement des algorithmes itératifs qui partent d'une estimation initiale. Une initialisation correcte joue un rôle clé dans l'obtention de bons résultats de regroupement. L'initialisation aléatoire a été largement utilisée par les chercheurs en raison de sa simplicité. Cependant, les suppositions aléatoires produisent souvent des résultats médiocres et l'algorithme de clustering itératif doit être exécuté plusieurs fois avec des points de départ différents afin d'obtenir de meilleures solutions. De nombreuses techniques d'initialisation avancées ont été proposées pour améliorer l'efficacité, par exemple des choix spécifiques des centres de cluster initiaux de la méthode K-Means classique ou une décomposition de valeur singulière basée sur factorisation matricielle non négative. Cependant, il manque toujours un principe d'initialisation qui est couramment applicable pour un large éventail de méthodes de regroupement itératif. En particulier, il y a peu de recherches sur le fait de savoir si une méthode de regroupement pourrait bénéficier des initialisations par les résultats d'autres méthodes de regroupement.

5.2 K-means

Comme nous l'avons vu précédemment, la NMF est strictement liée au problème de clustering, il est donc logique d'utiliser un algorithme de clustering pour augmenter les performances de la factorisation. En particulier, compte tenu de la matrice non-négative, on peut voir ses colonnes comme des points sur un axe pour chaque colonne dans dans les réels positifs, et appliquer un simple clustering K-Means, ou un Spherical K-means pour obtenir la matrice de centroïdes W.

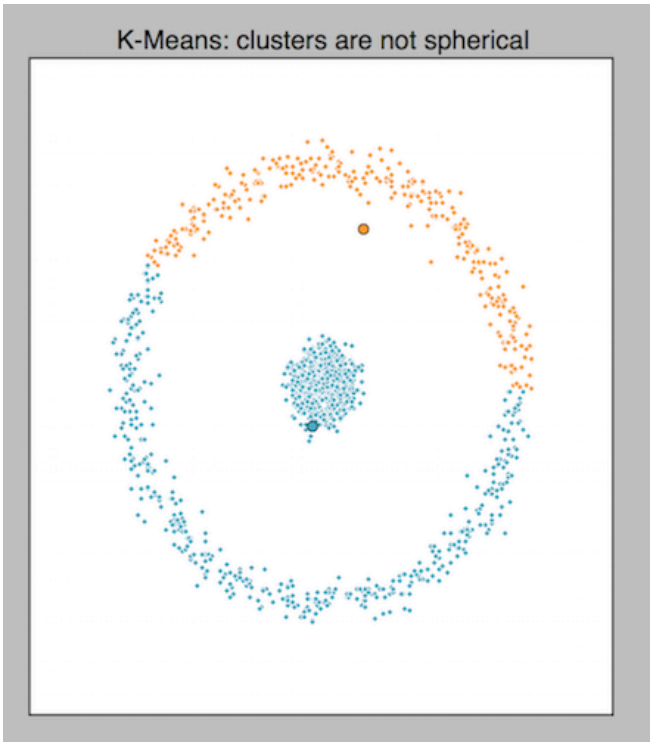


Figure 7. Illustration démontrant la non sphéricalité des partitions par K-Means.

K-Means garantit l'indépendance linéaire des centroïdes et améliore l'initialisation obtenue pour la factorisation par la suite. De plus en choisissant différents centroïdes de départ pour les algorithmes k signifie, nous pouvons obtenir différents W initial, de sorte que ce processus n'est pas déterministe et la sortie finale de NMF peut varier.

5.3 Spherical K-Means

En 2001, Dhillon et Modha ont introduit la méthode [13] Spherical K-Means pour regrouper de grands ensembles de données textuelles éparses. Cette méthode a été établie en tirant parti de certains pré-traitements souvent effectués lorsque l'on travaille avec du texte dans un modèle d'espace vectoriel. Afin d'attribuer un poids égal à chacun des n points d'un ensemble de données, les vecteurs de colonnes $x_1, x_2, .., x_n$ de la matrice de données $M \times N$ originale X sont normalisées pour être de longueur unitaire dans la norme euclidienne. L'effet de cette normalisation est de ne prendre en compte que la direction de chaque vecteur et non la longueur. Nous limitons l'ensemble de données de la matrice initiale pour qu'il soit entièrement constitué d'éléments non négatifs ce qui est une propriété naturelle de la plupart des modèles d'espace vectoriel de texte et d'image. Une

partie significative et non triviale de l'algorithme de Spherical K-Means (et respectivement K-Means) est de choisir k.
Le nombre de clusters disjoints pour partitionner les données est semblable au choix de r dans une factorisation matricielle non négative, la sélection de k pour Spherical K-Means reste un sujet ouvert et largement inexploré. C'est en grande partie à cause des objectifs variables que l'on peut avoir en se regroupant et de la forte dépendance d'un *ńrang latentż* sur le type et la structure des données. Dans des dimensions très élevées, il peut être impossible de visualiser l'ensemble de données de manière à révéler une valeur optimale pour k. En pratique, la borne $k << min(m, n)$ est habituellement choisie comme point de départ. Les approches pratiques pour sélectionner k sont analogues à celles utilisées dans l'analyse en composantes principales où une valeur k est choisie en fonction de la pente changeante de la fonction coût. Une méthode d'initialisation de ces centroïdes consiste à choisir (de façon aléatoire) chaque centroïde comme étant un vecteur de colonne différent de l'ensemble de données. Cette initialisation tente d'éviter une difficulté commune de nombreuses techniques K-Means qui se produisent lorsqu'un cluster initialisé ne contient aucun point après une itération. Pour les besoins du TER, nous avons choisi un intervalle de k pour chaque ensemble de test. Une fois le nombre de clusters déterminé, l'algorithme nécessite une initialisation centroïde pour chaque nombre de classe. Il convient de noter que l'initialisation Spherical K-Means a un impact non trivial sur le schéma de cluster résultant car Spherical K-Means (comme d'autres méthodes de descente du gradient [11]) sont susceptibles d'être piégés dans les maxima locaux lors de la recherche du vrai maximum global de la fonction coût donnée.

5.4 Double décomposition en valeurs singulières non négative

Les avantages de la SVD (Singular Value Decomposition) comprennent une propriété d'optimalité avec le fait que la SVD tronquée produit la meilleure approximation k (en termes de distances au carré), puisque le calcul est rapide et robuste. Ensuite l'unicité de la factorisation car l'initialisation n'affecte pas les algorithmes SVD [7].

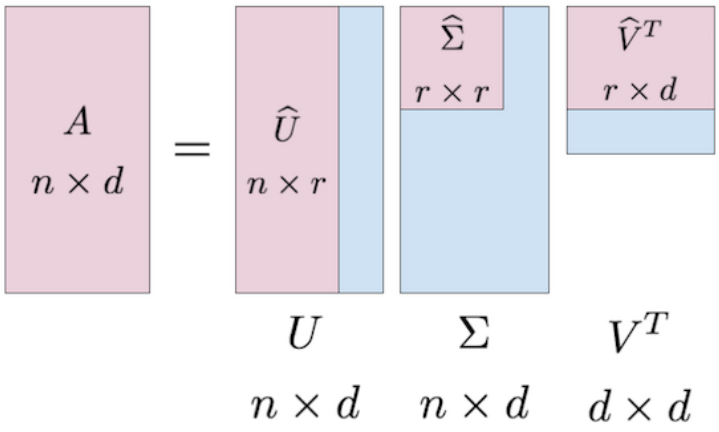


Figure 8. Illustration de la SVD.

Un des plus grand point faible de la NMF sont ses problèmes de convergence. Contrairement à la SVD, il n'y a pas de factorisation unique de la NMF. Les différents algorithmes de la NMF peuvent converger vers différents minima locaux (et même cette convergence vers des minima locaux n'est pas garantie), l'initialisation de l'algorithme devient critique. En pratique, la connaissance du domaine d'application peut aider à guider les choix d'initialisation. La solution au problème sans contraintes de positivité obtenues à travers la SVD, en général ne résout pas le problème NMF, car il peut y avoir des éléments négatifs dans les matrices W, H, mais si nous mettons les éléments négatifs à zéro, ou nous effectuons un autre processus pour en faire des matrices non négatives, nous pouvons utiliser la solution SVD comme point de départ pour les algorithmes itératifs, malgré l'initialisation aléatoire de W et H. La décomposition de valeur singulière double non négative ($NNDSVD$) tente d'ajuster la SVD d'une manière légèrement différente en produisant directement une décomposition non négative. Les variables W et H sont toutes deux positives, mais elles sont généralement éparses, elles sont donc modifiées en ajoutant ϵ plus petit que la moyenne des éléments aux entrées nulles. Dans le cas symétrique, cette méthode peut être ajustée pour obtenir $W = H$, et elle gagne en coût de calcul, puisque la SVD peut être remplacée par un processus de diagonalisation. Cette méthode d'initialisation n'a pas de partie aléatoire, donc un algorithme NMF ne produira qu'un seul minimum local.

5.5 Analyse en Composantes Principales

Dans cette section, nous parlerons de l'ACP (Analyse en Composantes Principales). L'analyse en composantes principales est une méthode statistiques multivariée initié par K. Pearson en 1901. L'un des problèmes les plus connus dans l'exploration de données est la reconnaissance de formes, à savoir le problème

de trouver des caractéristiques similaires dans différents objets. Les domaines de Text Mining et Image Processing nécessitent des méthodes pour résoudre ces problèmes afin de catégoriser les données, où pour trouver des fonctionnalités communes simples qui nous permettent de décrire les objets de manière compressée. L'un des principaux outils utilisés de nos jours pour la compression et l'identification des caractéristiques communes est l'ACP, obtenue à partir de la décomposition des valeurs singulières des données. En effet, étant donné un ensemble d'objets (images, documents, signaux, etc.), identifiés par des vecteurs réels empilés comme des colonnes de la matrice, l'analyse trouve la meilleure approximation d'un rang inférieur des données originales, et les stocke dans un espace d'ordre de grandeurs plus petites que l'entrée. Nous pouvons utiliser la SVD pour trouver les k caractéristiques les plus importantes qui sont communes à tous les objets d'entrées.

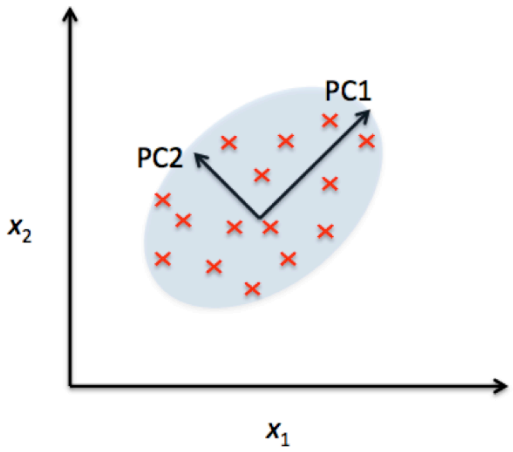


Figure 9. Illustration de l'ACP.

La SVD nous donne la meilleure approximation k de A en norme L_2 et Frobenius, (et en général dans toute norme unitairement équivalente) et elle est relativement facile à calculer, donc largement utilisé dans beaucoup d'applications. Une particularité de l'ACP est qu'elle produit des entrées négatives dans les entités et les coefficients, même lorsque A est non négatif.
Récemment, de nombreuses applications ont introduit la requête non négative dans leur algorithme [14] de reconnaissance de formes afin d'obtenir l'interprétabilité de la sortie. Dans ce contexte, on peut l'utiliser dans le cadre de la NMF car sa décomposition préserve la positivité des données rendant les résultats lisibles.

5.6 Analyse en composantes Indépendantes

L'analyse indépendante des composants est similaire à l'ACP puisqu'elle fait également tourner une matrice dans un nouvel ensemble de coordonnées. Cependant, contrairement à l'ACP, l'objectif de l'ICA est de réaliser ensemble d'échantillons de données de sorte qu'ils soient statistiquement indépendants les uns des autres. Étant donné la matrice X dont les colonnes sont des échantillons de données individuels, l'ICA tente de déterminer les matrices A et Y telles que les colonnes de Y soient statistiquement indépendantes :

$$X = AY \tag{3}$$

En outre, l'ICA calcule l'inverse de A, l'appelant W, de telle sorte que :

$$Y = WX \tag{4}$$

W et A sont respectivement appelés matrices non mélangées et matrice de mélange parce que les observations X sont considérées comme des mélanges de sources statistiquement indépendantes Y. L'indépendance statistique est ici définie par les densités de probabilité des échantillons. Considérons deux variables aléatoires y_1 et y_2 avec des densités de probabilité $p_1(y_1)$ et $p_2(y_2)$. Si la densité de probabilité conjointe $p(y_1, y_2)$ peut être écrite comme le produit des fonctions de densité de probabilité individuelles, alors les variables sont dites indépendantes. Dans le cas de l'analyse d'images, l'ICA ne peut pas être appliqué directement. Le but n'est pas de déterminer les composantes dont les formes sont indépendantes les unes des autres, mais plutôt de déterminer les composantes dont les contributions relatives, sont indépendantes les unes des autres. En d'autres termes, les changements dans la contribution d'une composante devraient fournir le moins d'informations possible sur les changements dans les contributions des autres composantes. Une bonne première étape consiste alors à trouver les composantes principales des caractéristiques via l'ACP.

$$X = UAY \tag{5}$$

Maintenant, les colonnes du produit UA contiennent les composantes indépendantes qui, lorsqu'elles sont mélangées par Y, produisent les caractéristiques observés contenus dans les colonnes de X. Les graphiques concernant l'ICA sont disponibles sur le dépôt Github [ici].

6 Étude comparative

6.1 Estimation du rang de la NMF

Le rang de factorisation est un paramètre très important de la NMF. Tout comme pour le nombre de cluster de K-means et Spherical K-Means. Il peut s'avérer difficile de déterminer le bon nombre de cluster. La façon la plus courante pour déterminer le nombre de cluster k est d'essayer plusieurs valeurs de k, calculer

plusieurs mesures de qualités et choisir la meilleure valeur de k suivant ces critères. Ainsi, plusieurs approches ont été proposées afin de choisir la valeur de k optimal :
- La première approche est de choisir la première valeur de r où le coefficient de *cophenetic* commence à décroître.
- Une deuxième approche consiste à choisir la première valeur de r ou la courbe de RSS présente un point d'inflexion.
- Enfin, une troisième approche est de choisir k quand le coefficient de silhouette est le plus élevé.

6.2 Évaluation des données

La première étape a consisté à déterminer le meilleur rang pour la NMF. Pour cela, on a calculé la NMF sur un rang d'intervalle 20 à 70..

Figure 10. Récapitulatif des différents résultats obtenus pour le rang 20 à 70.

Ainsi, ces résultats nous poussent à choisir comme rang $k = 33$. En effet, nous pouvons voir que le coefficient de cophénétique est autour de 1 avant une forte décroissance. De plus le consensus et la silhouette présentent un meilleur graphique.

Une fois le rang déterminé, nous avons voulu tester les différentes méthodes et initialisation, le tableau ci-dessous regroupe les critères pour toutes les méthodes testées.

Méthodes	Sparseness basis	Sparseness coef	Silhouette	Residuals	Evar
Brunet - random	0.2304	0.6505	-0.0834	256.65	0.946
Brunet - svd	0.4578	0.5784	0.01967	326.62	0.931
Brunet - kmeans	0.2627	0.6082	-0.0528	261.81	0.945
Brunet - skmeans	0.2692	0.6012	-0.0585	263.84	0.944
Brunet - pca	0.4806	0.8464	-0.1377	353.14	0.926
Lsnmf - random	0.1249	0.8313	-0.0693	231.07	0.951
Lsnmf - svd	0.1311	0.8041	-0.0658	232.15	0.951
Lsnmf - kmeans	0.1302	0.8228	-0.0625	232.19	0.951
Lsnmf - skmeans	0.1281	0.8253	-0.0793	231.89	0.951
Lsnmf - pca	0.1429	0.7938	-0.0671	234.50	0.950
Sepnmf - random	0.0855	0.7595	-0.0607	448.95	0.906
Sepnmf – svd	0.0855	0.7595	-0.0607	448.95	0.906
Sepnmf – kmeans	0.0855	0.7595	-0.0607	448.95	0.906
Sepnmf - skmeans	0.0855	0.7595	-0.0607	448.95	0.906
Sepnmf - pca	0.0855	0.7595	-0.0607	448.95	0.906
Nsnmf – random	0.3394	0.7493	-0.0809	312.45	0.934
Nsnmf – svd	0.4753	0.6826	-0.0673	370.97	0.922
Nsnmf – kmeans	0.3769	0.7399	-0.0871	315.50	0.933
Nsnmf - skmeans	0.3867	0.7275	-0.0822	319.57	0.933
Nsnmf - pca	0.9461	0.8439	-0.0864	1991.0	-4168.44

Figure 11. Récapitulatif sur les différentes modèles.

La colonne *Evar* donne les valeurs obtenues pour la variance expliquée :

$$evar = 1 - \frac{RSS}{SST} \tag{6}$$

ou RSS est la somme des carrés des résidus (residual sum of squares) et où SST, la somme totale des carrés (total sum of squares).

Les erreurs et les résidus sont deux mesures étroitement liées et facilement confondues de la déviation d'une valeur observée d'un élément d'un échantillon statistique à partir de sa "valeur théorique". L'erreur d'une valeur observée est l'écart de la valeur observée par rapport à la valeur vraie (inobservable) d'une quantité d'intérêt (par exemple, une moyenne de pixel ayant la même valeur dans une colonne), et le résidu d'une valeur observée est la différence entre valeur observée et la valeur estimée de la quantité d'intérêt (par exemple, une moyenne de colonne ayant les mêmes valeurs). La distinction est importante car les concepts sont parfois appelés les erreurs de régression et les résidus de régression et où ils conduisent à la notion de résiduels étudiés.

Afin de choisir la meilleure méthode, nous cherchons donc à minimiser le *residuals* et maximiser la silhouette et *l'Evar*. Nous pouvons voir que la méthode Lsnmf présente les meilleurs résultats pour le *residuals* et *l'Evar*. La méthode d'initialisation semble ne pas faire une différence, les résultats sont très similaires

concernant la Lnsmf. Il est plus intéressant de choisir une initialisation aléatoire afin d'économiser du temps de calcul et de complexité.

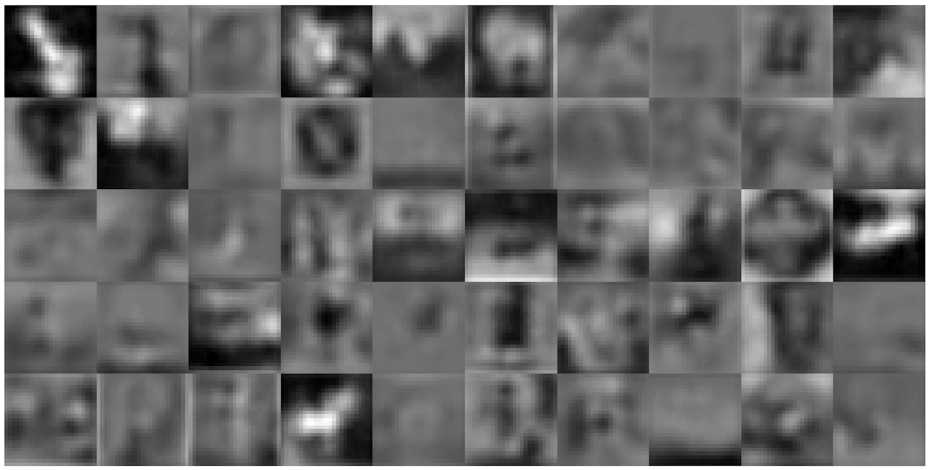


Figure 12. Recomposition avec le rang 33.

Figure 13. Recomposition des images avec le jeu de données original.

6.3 Résultats

Après avoir déterminé le meilleur rang pour la NMF et le nombre de cluster optimal pour K-Means et Spherical K-Means, nous avons voulu comparer les partitions obtenues entre la NMF et le clustering. Pour cela nous avons calculé l'ARI et la NMI :

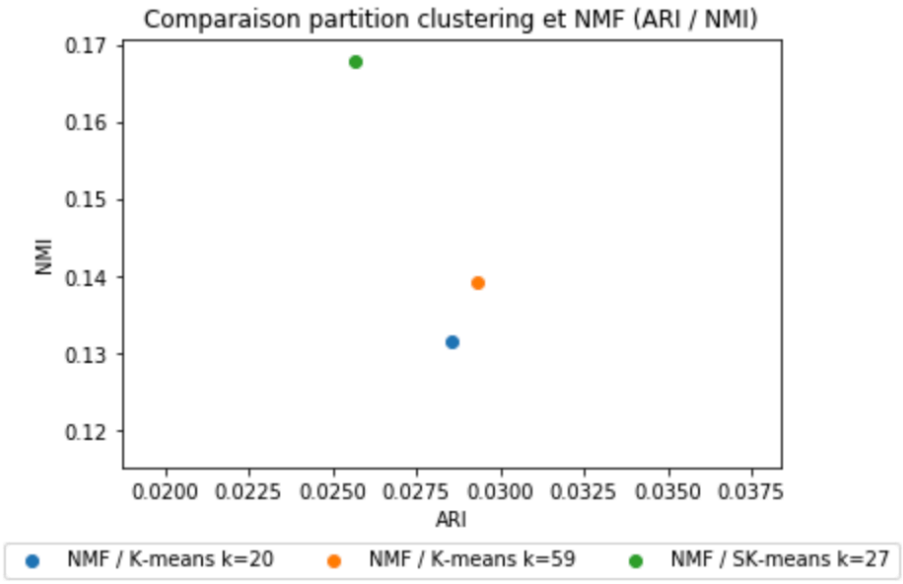


Figure 14. Comparaison partition NMF/Clustering

Nous pouvons voir que les résultats ARI/NMI sont très faible, ce qui signifie que les partitions obtenues avec la NMF et K-Means/Spherical K-Means sont très différentes et qu'il n'existe aucune similarité entre ces partitions.
Ces résultats sont cependant assez cohérent avec les résultats obtenues tout au long de ce rapport. En effet le score de silhouette des différentes initialisation de K-Means et Spherical K-Means étaient très faible aussi, ce qui peut signifier qu'il n'existe pas de bonne séparation entre les classes dans les données. Il est donc fort probable que plusieurs exécutions de K-Means nous donnent des partitions très différentes.

7 Conclusion

Pour conclure nous aborderons les cas que nous n'avons pas traités ainsi que du jeu de données en soi, le clustering est un domaine très compliqué car nous voyageons dans le flou comme on a pu le constater durant ce semestre mais nous avons appris beaucoup sur la classification automatique. Tout d'abord, nous avons donné les différents résultats et leur explication mais cela nous a appris plusieurs choses très utiles en tant qu'étudiants en sciences des données. Même avec un pré-traitement réfléchi ou sans, finalement les résultats était mauvais ainsi qu'une multitude de combinaisons différents de la NMF (algorithmes, méthodes, initialisations) ne changeront rien au faits. Le problème vient du jeu de données qui est en soi pas adapté et correspond aux points faibles de la NMF. Les coefficients sont des nombres positifs certes mais pour chaque vecteur dans la base, la quantité d'information est généralement une petite partie que nous utilisons pour reconstruire nos points. Les lignes ayant une trop grande variété en soi ne permettent pas à la NMF de trouver un *pattern*. En particulier, il faut se rendre compte que si la NMF est beaucoup utilisée dans les sciences, son rigoureux fondement est seulement découvert depuis moins de 30 ans. Au moment où nous écrivons et il est très probable que nous n'avons pas encore trouvé le meilleur algorithme pour cela. Ensuite, nous avons fait face à un manque de temps pour réaliser les analyses avec une matrice RGB et binaire. Il faut savoir que nous avons utilisé deux différents services de *Cloud Computing* (Amazon Web Service, Google Cloud Computing), les librairies *Nimfa* et *NMR* pour R utilise le CPU alors que pour des calculs matriciels intense, un minimum est d'avoir recours au GPU afin de réduire drastiquement le temps de calcul. Toutes les ressources (graphiques, scripts) du projet sont disponibles [ici]. Finalement, le jeu de données ayant était construit pour de la cross-depiction, nous pouvons affirmer que la NMF comme technique n'est pas adaptée pour ce problème là (d'autres solutions existent comme les réseaux de neurones convolutifs [15] ou Deep Semi-NMF [16]).

Références

1. N. Westlake, H. Cai, and P. Hall, "Detecting people in artwork with cnns," in *European Conference on Computer Vision*, pp. 825–841, Springer, 2016.
2. G. Rossum, "Python reference manual," tech. rep., Amsterdam, The Netherlands, The Netherlands, 1995.
3. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn : Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
4. R Core Team, *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.
5. R. Gaujoux and C. Seoighe, "A flexible r package for nonnegative matrix factorization," *BMC Bioinformatics*, vol. 11, p. 367, Jul 2010.

6. N. Gillis and S. A. Vavasis, “Fast and robust recursive algorithmsfor separable nonnegative matrix factorization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 698–714, April 2014.
7. C. Boutsidis and E. Gallopoulos, “Svd based initialization : A head start for nonnegative matrix factorization,” *Pattern Recognition*, vol. 41, no. 4, pp. 1350 – 1362, 2008.
8. M. Zitnik and B. Zupan, “Nimfa : A python library for nonnegative matrix factorization,” *Journal of Machine Learning Research*, vol. 13, pp. 849–853, 2012.
9. J.-P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov, “Metagenes and molecular pattern discovery using matrix factorization,” *Proceedings of the National Academy of Sciences*, vol. 101, no. 12, pp. 4164–4169, 2004.
10. D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” pp. 556–562, 2001.
11. C.-J. Lin, “Projected gradient methods for nonnegative matrix factorization,” *Neural Comput.*, vol. 19, pp. 2756–2779, Oct. 2007.
12. S. Wild, “Seeding nonnegative matrix factorizations with the spherical k-means clustering,” 01 2002.
13. K. Hornik, I. Feinerer, M. Kober, and C. Buchta, “Spherical k-means clustering,” vol. 50, pp. 1–22, 09 2012.
14. K. Allab, L. Labiod, and M. Nadif, “A Semi-NMF-PCA Unified Framework for Data Clustering,” *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 29, pp. 2–16, Jan. 2017.
15. H. Cai, Q. Wu, T. Corradi, and P. Hall, “The cross-depiction problem : Computer vision algorithms for recognising objects in artwork and in photographs,” *CoRR*, vol. abs/1505.00110, 2015.
16. J. Flenner and B. Hunter, “A deep non-negative matrix factorization neural network,” 2017.