UMONS
Université de Mons

Faculté Polytechnique

Factorisation Non-Négative de Matrices Creuses en Norme ℓ_1

Extraction de Thèmes et Classification Non-Supervisée de Textes

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

Kévin DUBRULLE
Étudiant Master en Ingénieur Civil Informatique et Gestion, à Finalité Spécialisée en Intelligence Artificielle et Aide à la Décision

POLYTECH MONS

Sous la direction de :
Nicolas GILLIS
Arnaud VANDAELE

Juin 2023

pôle hainuyer

Remerciements

J'aimerais commencer par remercier toutes les personnes m'ayant apporté du soutien dans l'élaboration de ce travail de fin d'étude, que ce soit un soutien psychologique ou une aide directe. Je remercie aussi tous ceux m'ayant aidé durant ces cinq années d'études.

Plus particulièrement, j'aimerais remercier monsieur Nicolas GILLIS et monsieur Arnaud VANDAELE, mes promoteurs de ce TFE. Sans eux, ce travail n'aurait certainement pas été possible. Ce sont les discussions avec eux qui m'ont permis d'écrire ce travail, de me guider dans les différentes étapes d'élaboration de chaque point, ... Je remercie également François Moutier, qui a travaillé sur certains algorithmes présentés.

Je remercie également Florian DUBOIS, à la fois pour son aide apporté pour le TFE qui m'a notamment permis d'élaborer certaines équations, mais aussi pour son aide et sa présence durant toutes ces années d'études à ses côtés.

Je remercie également la faculté, ses professeurs, technicien ainsi que tout le personnel académique de m'avoir permis d'étudier ici. Je remets un remerciement particulier à tout le département IG ainsi qu'à monsieur Philippe FORTEMPS en tant que président de cette option qui m'a tant apporté.

Je remercie également tous les étudiants ayant eu ce parcours d'étude avec moi. Sans eux, ces années n'auraient certainement pas été aussi belles.

Je remercie également tous mes profs de secondaire et primaire ayant eu un impact sur mes choix de vies, ainsi que ma famille pour le soutien qu'ils m'auront apporté tous le long de mon parcours scolaire.

Je m'excuse pour toutes les personnes qui auraient mérité leur nom dans cette section, mais que j'ai malheureusement oublié. Merci à tous.

Résumé

La factorisation non-négative de matrices a pour but d'approximer une grande matrice composée d'éléments positifs ou nuls par le produit de deux matrices composées également d'éléments positifs ou nuls. Ce type de factorisation est généralement utilisé pour ressortir les caractéristiques d'un ensemble de données, qui, dans beaucoup de cas, sont positives ou nulles. La factorisation non-négative de matrices utilise généralement la norme ℓ_2 ou la divergence de KullBack-Leiber dans sa fonction objectif.

Dans ce travail, nous nous concentrerons principalement sur l'extraction de thèmes de textes ainsi que leur classification non-supervisée. Ce type de données est souvent très creux. La norme ℓ_1 est généralement mieux adaptée aux données composées de beaucoup de zéros, nous mettrons alors en place un modèle de factorisation non-négative de matrices creuses utilisant cette norme. Nous testerons l'algorithme de résolution basique, et nous proposerons de petites variantes permettant d'améliorer.

Dans des cas réels, des données creuses seront souvent composées de "faux" zéros et de "vrai" zéros, sans que nous puissions les distinguer. Ce travail étudiera également un modèle de factorisation négative en norme ℓ_1 dépénalisant les zéros afin de donner moins d'impact aux "faux" zéros dans nos prédictions. Deux variantes de ce modèle seront créées, la première effectue une dépénalisation globale dans l'ensemble des données, tandis que la seconde effectue une dépénalisation plus ciblée, qui s'appliquera texte par texte.

Table des matières

Liste des Algorithmes

Table des figures

Liste des tableaux

Liste des Abréviations

CD	*Coordinate Descent*	Descente de Coordonnées
C-CD	*Correlated Coordinate Descent*	Descente de Coordonnées par Corrélation
R-CD	*Random Coordinate Descent*	Descente Aléatoire de Coordonnées
S-CD	*Slope Coordinate Descent*	Descente de Coordonnées par Pentes
HALS	*Hierarchical Alternating Least Squares*	
MU	*Multiplicative Update*	
NMF / NNMF	*Non-Negative Matrix Factorization*	Factorisation Non-Négative de Matrices
DR-NMF	*Distributionnally Robust Non-Negative Matrix Factorization*	
KL-NMF	*KullBack-Leiber Non-Negative Matrix Factorization*	Factorisation Non-Négative de Matrices ulisant la divergence KullBack-Leiber
ℓ_1-NMF / Mah-NMF	*Manhattan Non-Negative Matrix Factorization*	Factorisation Non-Négative de Matrices en norme ℓ_1 ou distance de Manhattan
ℓ_2-NMF / FRO-NMF	*Frobenius Non-Negative Matrix Factorization*	Factorisation Non-Négative de Matrices en norme ℓ_2 ou norme de Frobenius
PCA	*Princpal Components Analysis*	
resp.	*respectively*	respectivement
s.t	*such that*	tel que
SVD	*Singular Values Decomposition*	

Chapitre 1

Introduction

1.1 Introduction

Ce travail a pour but d'effectuer de l'extraction de thèmes de différents textes ainsi que de faire de la classification non-supervisées de ces textes en fonction des thèmes qui leur ont été assignés. Nous devrons alors réfléchir à un modèle efficace pour ce cas de figure.

Dans ce premier chapitre, nous commencerons par décrire nos données, afin de comprendre au mieux les cas d'utilisation que devra respecter notre modèle. Nous analyserons ensuite les méthodes existantes pour savoir construire notre modèle dans la suite du travail.

1.2 Description du Problème

Les jeux de données sur lesquels nous serons amenés à travailler sont des ensembles de n documents. Ces documents sont représentés par le nombre d'occurrences de chaque mot dans leur texte. Nous stockerons chaque jeu de données dans une matrice X, tel que la valeur $X_{i,j}$ représentera le nombre d'occurrences du mot i dans le texte j. La figure 1.1 permet de visualiser la manière dont est représenté un jeu de données dans la matrice X.

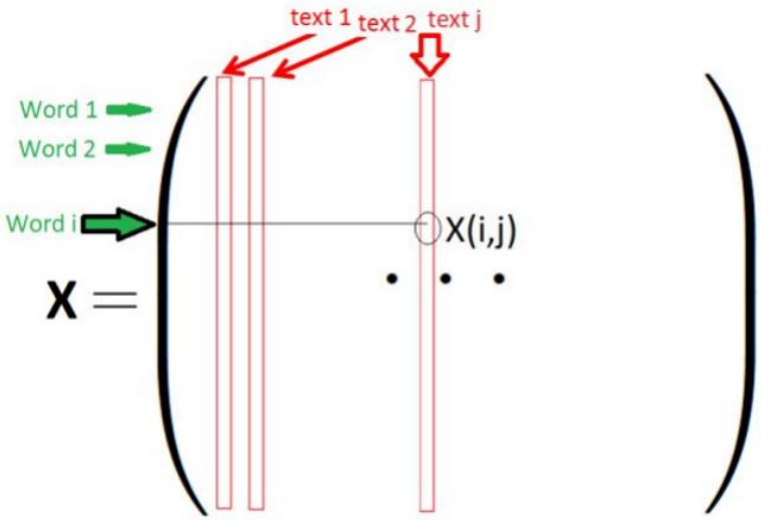


FIGURE 1.1 – Représentation du Jeu de Données dans la Matrice X [5].

Notre travail sera d'extraire plusieurs thèmes présents dans l'ensemble des textes. Chaque thème sera représenté par un sous-ensemble de mots caractéristiques, et chaque mot se verra attribuer un poids en fonction de son importance (voir figure 1.2). À partir des thèmes et d'un nouveau texte, nous pourrons alors déduire le contenu global de ce texte.

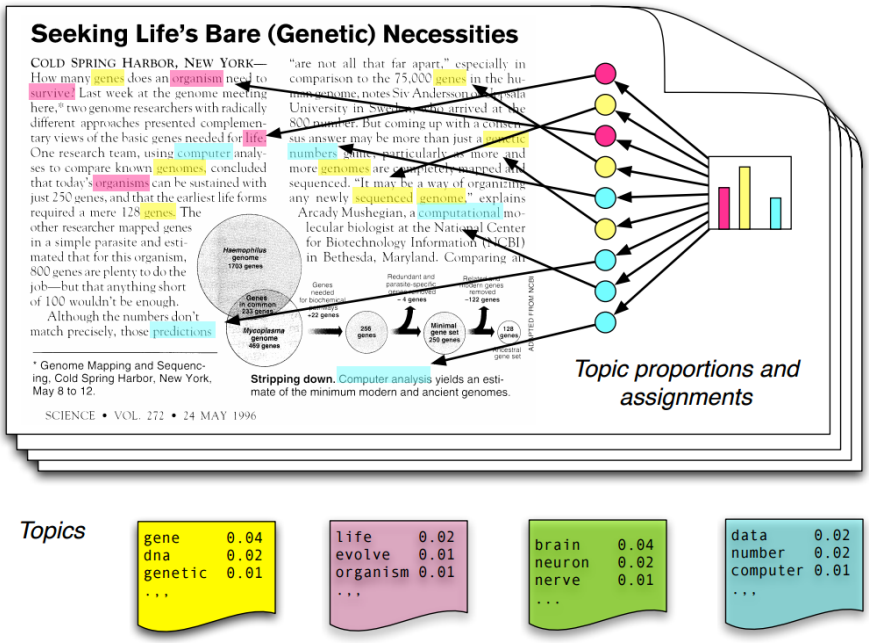


FIGURE 1.2 – Extraction des Thèmes d'un Texte [3].

Dans le cadre de ce travail, nous nous concentrerons principalement sur une méthode appelée la Factorisation Non-Négative de Matrices (*Non-Negative Matrix Factorization*, NMF ou NNMF). La NMF est une approche de factorisation d'une matrice non-négative $X \in \mathbb{R}_+^{m \times n}$ par le produit de deux matrices non-négatives $W \in \mathbb{R}_+^{m \times r}$ et $H \in \mathbb{R}_+^{r \times n}$ avec $r \ll \min(m,n)$. Une matrice non-négative ne possède aucun élément inférieur à 0, donc pour tout indice i et j : $X_{i,j} \geq 0$. La NMF est alors définie par

$$X \approx WH \qquad W,H \geq 0. \tag{1.1}$$

Par rapport à nos types de données, la matrice W contiendra alors l'ensemble des thèmes. Une colonne $W_{:,k}$ sera composée de l'importance que possède chaque mot au thème k. La matrice H contiendra pour chaque texte un coefficient propre à la pertinence de chaque thème pour ce texte. L'élément $H_{k,j}$ sera alors représentatif de l'appartenance du texte j au thème k.

L'intérêt de travailler avec des matrices non-négatives se trouve dans la signification des résultats. Il semblerait étrange qu'un thème portant sur le sport se représente en ayant, par exemple, des valeurs négatives sur les mots propres à un thème scientifique. De plus, un texte pourrait à la fois porter sur un thème sportif et scientifique. Il semble alors plus pertinent de définir un thème en fonction des mots qu'il contient, plutôt qu'en fonction des mots qu'il ne contient pas. La matrice X initiale est déjà non-négative, et il semble pertinent de garder des matrices W et H non-négatives également.

1.3 Factorisation Non-Négative de Matrices (NMF)

La NMF est définie par l'équation (1.1). Dans notre cas, la matrice X est composée de n colonnes dans un espace à m dimensions. Après la factorisation, tout point j sera alors représenté par une combinaison linéaire de r vecteurs $W_{:,k}$ (pour $k = 1,...r$) représentatifs de l'ensemble des points. Chaque vecteur k sera multiplié par un coefficient $H_{k,j}$ de manière à représenter au mieux le point j :

$$X_{:,j} \approx \sum_{k=1}^{r} W_{:,k} * H_{k,j} \qquad j = 1,...n. \tag{1.2}$$

Dans le cas de la NMF, la matrice X initiale ne possède que des valeurs supérieures ou égales à zéro. La NMF va donc imposer aux matrices W et H d'être également non-négatives. Il sera alors possible d'interpréter plus facilement les résultats obtenus. En effet, dans des cas réels, les données seront souvent représentées par des nombres non-négatifs, et il n'y aurait pas de sens physique à avoir des valeurs négatives dans la factorisation.

Le principe de la NMF sera alors de trouver le produit des deux matrices W et H le plus proche possible de X. Plusieurs fonctions permettent d'exprimer la proximité entre WH et X. Les fonctions objectifs les plus fréquentes sont la divergence Kullback-Leibler et la norme ℓ_2 (ou "norme de Frobenius" ou "norme Euclidienne"). Ces deux variantes sont respectivement robustes à la distribution de Poisson et au bruit Gaussien [12]. Dans le cas de la norme ℓ_2, la fonction d'optimisation est

$$\min_{W,H} f(W,H) = ||X - WH||_F^2 = \sum_{i,j} (X - WH)_{i,j}^2 \quad s.t. \quad W,H \geq 0. \tag{1.3}$$

1.4 Autres Applications de la NMF

La NMF a beaucoup d'applications dans différents domaines autres que l'extraction de thèmes, nous pouvons citer plusieurs exemples :

— **L'extraction de caractéristiques d'un visage**. À partir de n images de visages, le but est de ressortir les caractéristiques principales permettant de distinguer ces visages. Un visage pourra ensuite être reconstruit à partir de la présence plus ou moins importante des caractéristiques, comme illustré à la figure 1.3.

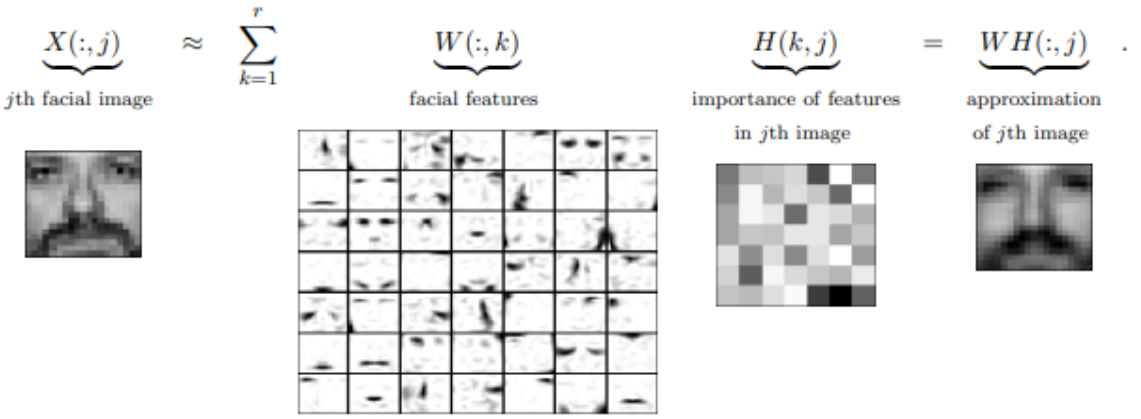


FIGURE 1.3 – Extraction des caractéristiques de visages [8].

— **La décomposition hyperspectrale d'image**. Ce type d'application illustré à la figure 1.4 a comme données une liste de n "pixels" qui seront représentés par leur spectre. La NMF ressortira alors les r spectres caractéristiques de l'image, et chaque pixel sera alors une composition de ces spectres. Les r spectres caractéristiques vont le plus souvent correspondre à un matériau et la NMF permet alors de déterminer la composition physique de chaque point d'une image.

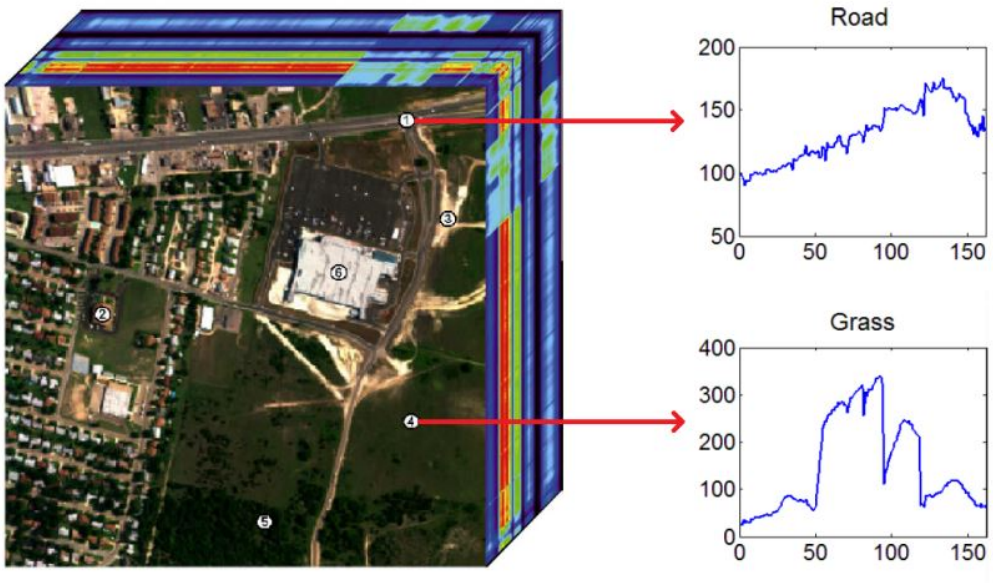


FIGURE 1.4 – Image Hyperspectrale (307 x 307 pixels, 162 bandes spectrales) [8].

— **Réduction de dimensions**. Le principe est similaire à l'algorithme *Princpal Components Analysis* (PCA), à partir de n données placées dans un espace à m dimensions, la NMF replace ces données dans un espace à r dimensions, tout en essayant de garder au mieux leurs différences et similarités entre elles.

1.5 Résolution de la ℓ_2-NMF

La fonction d'optimisation (1.3) n'est pas convexe. Résoudre le problème de la NMF revient à résoudre un problème NP-HARD [24], il est donc très complexe dans la plupart des cas de trouver la solution optimale. Cependant, en fixant une des deux matrices W ou H, le problème devient alors convexe [8]. La plupart des algorithmes de NMF sont alors basés sur une structure de "*block coordinate descent*", où à chaque itération, W sera mise à jour en fixant H, puis H sera mise à jour en fixant W et ainsi de suite (voir Algorithme 1).

Algorithme 1 $[W,H]$ = two_block_descent(X,r)

INPUT : $X \in \mathbb{R}_+^{m\times n}, r \in \mathbb{N}$
OUTPUT : $W \in \mathbb{R}_+^{m\times r}, H \in \mathbb{R}_+^{r\times n}$
$[W^{(0)},H^{(0)}] \leftarrow initializer(X,r)$
while stop criterion not satisfied **do**
    $W^{(i)} \leftarrow update\_W(X,W^{(i-1)},H^{(i-1)})$
    $H^{(i)} \leftarrow update\_H(X,W^{(i)},H^{(i-1)})$
end while

La NMF possède une symétrie pour W et H. Elle est définie par

$$||X - WH||_F = ||X^T - H^T W^T||_F. \tag{1.4}$$

Cette symétrie permet de n'avoir qu'une seule fonction de mise à jour pour W et H. Il suffit de modifier les arguments des fonctions *update* selon la symétrie. Nous pourrons alors modifier l'Algorithme 1 avec la ligne

$$W^{(i)} \leftarrow update\_H(X^T, H^{(i-1)^T}, W^{(i-1)^T})^T. \tag{1.5}$$

Plusieurs algorithmes de résolution existent dans la littérature. Nous pouvons citer l'algorithme *Multiplicative Update* (MU) [8], qui est basé sur la mise à jour suivante, où les opérateurs $\circ$ et $\div$ sont respectivement la multiplication point à point et la division point à point :

$$H^{(i)} \leftarrow H^{(i-1)} \circ \frac{W^{(i)^T} X}{W^{(i)^T} W^{(i)} H^{(i-1)}}. \tag{1.6}$$

Un autre algorithme fréquemment utilisé est le *Hierarchical Alternating Least Squares* (HALS) [8]. Il se base sur de la descente de coordonnées, c'est-à-dire qu'il optimise chaque élément de la matrice individuellement pour minimiser l'objectif.

L'Algorithme 2 HALS converge généralement plus rapidement que l'algorithme MU et garantit presque toujours de converger en un point stationnaire [9]. HALS possède de plus quelques variantes accélérant sa convergence [9].

Algorithme 2 $H^{(i)}$ = HALS$(X, W, H^{(i-1)})$

1: **INPUT :** $X \in \mathbb{R}_+^{m \times n}, W \in \mathbb{R}_+^{m \times r}, H^{(i-1)} \in \mathbb{R}_+^{r \times n}$
2: **OUTPUT :** $H \in \mathbb{R}_+^{r \times n}$
3: $H^{(i)} \leftarrow H^{(i-1)}$
4: **for** $k = 1 : r$ **do**
5: $\quad H^{(i)}_{k,:} \leftarrow \max\left(0, \frac{(W^T X)_{:,k} - \sum_{l=1, l \neq k}^{r} (W^T W)_{k,l} H^{(i-1)}_{l,:}}{(W^T W)_{k,k}}\right)$
6: **end for**

L'Algorithme 1 de descente par bloc nécessite d'avoir un itéré initial $H^{(0)}$ et $W^{(0)}$. Comme pour beaucoup de problèmes non-convexes, trouver un bon itéré initial est un problème en soi, et ce point de départ influera sur la qualité de la solution vers laquelle les algorithmes convergeront. L'initialisation n'est donc pas à négliger.

Les méthodes d'initialisation les plus simples se contentent de remplir les matrices H et W de 0, de 1 ou de nombres aléatoires. Ces méthodes peu coûteuses peuvent cependant aboutir à un résultat très médiocre en fonction du jeu de données et de la méthode de mise à jour utilisée. D'autres méthodes plus complexes existent, et nécessitent parfois d'utiliser d'autres algorithmes d'apprentissage non-supervisés tel que *k-means* ou la SVD [8].

Les critères d'arrêt de l'Algorithme 1 sont également multiples. Le choix du critère d'arrêt est généralement moins important que celui de l'initialisation, mais il doit tout de même être choisi de manière intelligente en fonction de l'objectif voulu. Les critères les plus simples sont un nombre maximal d'itérations ou un temps limite pour l'algorithme. Bien qu'utile pour s'assurer que l'algorithme ne prenne trop de temps, ils ne sont cependant pas robustes et il est possible que l'algorithme n'ait pas convergé vers un minimum.

Il existe des critères de convergence plus adaptables et robustes à divers cas. Par exemple, lorsque la différence entre les facteurs obtenus entre deux itérés successifs est faible, nous pouvons supposer que l'algorithme a convergé et qu'il n'est pas nécessaire d'effectuer plus d'itérations. Il est également possible d'utiliser la différence de la fonction d'optimisation entre deux itérations. Si celle-ci ne dépasse pas un certain seuil, nous supposons alors que l'algorithme a convergé. Pour un seuil de 10^{-6}, le critère d'arrêt à respecter sera alors le suivant :

$$\frac{||W^{(i)}H^{(i)} - W^{(i-1)}H^{(i-1)}||_F^2}{||X||_F^2} < 10^{-6}. \tag{1.7}$$

Il faut cependant noter que certaines méthodes peuvent boucler entre plusieurs solutions, et un itéré ne convergera alors jamais par rapport à l'itéré précédent. Si de telles méthodes sont utilisées, il faut alors prendre en considération ce problème lorsque nous choisirons le critère d'arrêt.

D'autres critères d'arrêt existent, parfois plus complexes ou plus robustes, mais nous n'en parlerons pas dans ce document [8].

1.6 Approches pour des Matrices Creuses

Dans l'ensemble d'un jeu de données, la plupart des textes n'utilisent qu'une petite partie du dictionnaire de mots. Beaucoup d'occurrences de mots vaudront alors zéro dans la matrice X. La matrice X sera alors dite "matrice creuse" (*sparse matrix*).

Un bon algorithme d'approximation de la matrice X devra alors donner une matrice creuse également. Par la nature non-négative des matrices W et H, elles devront également être creuses pour produire une matrice WH creuse. Nous pouvons le montrer de la manière suivante :

$$\begin{aligned}(WH)_{i,j} &= \sum_{k=1}^{r} W_{i,k} * H_{k,j} = 0 \\ &\iff W_{i,k} * H_{k,j} = 0 \iff W_{i,k} = 0 \quad \textbf{OU} \;\; H_{k,j} = 0 \qquad \forall k = 1, ...r.\end{aligned} \tag{1.8}$$

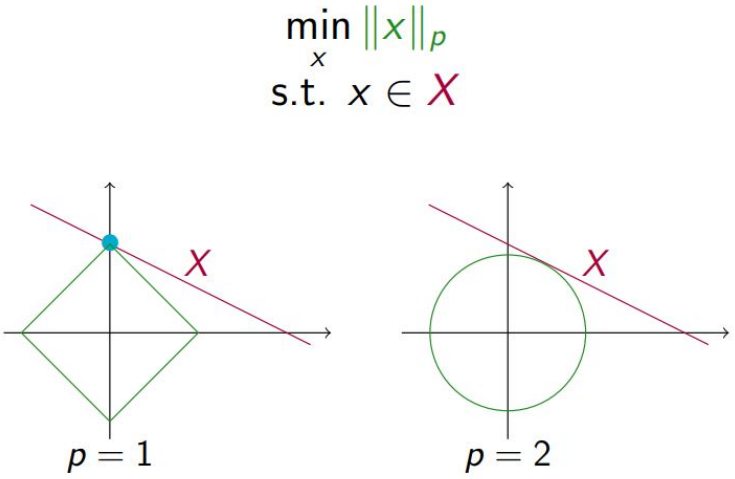
La NMF a parfois tendance à produire des matrices creuses [8]. Cependant, la ℓ_2-NMF aura dans certains cas tendance à privilégier des valeurs approchant zéros plutôt que de prédire réellement des zéros (voir figure 1.6). Il existe de nombreuses variantes de la NMF pouvant privilégier la formation de matrices creuses. Différents travaux existent et ajoutent le plus souvent une pénalité de *sparsité* dans la fonction objectif dépendant souvent d'un paramètre [25], par exemple la fonction suivante :

$$\min_{W,H} f(W,H) = ||X - WH||_F^2 - \eta ||W||_{1/2} \quad s.t. \quad W,H \geq 0, \tag{1.9}$$

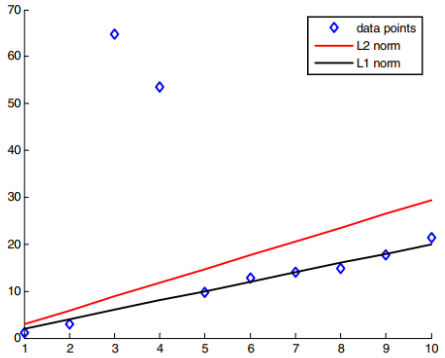
avec $||W||_{1/2} = \left(\sum_{i=1}^{n} \sum_{k=1}^{r} W_{i,k}^{\frac{1}{2}} \right)^2$ la pénalité de sparsité et $\eta \geq 0$ un paramètre de la fonction.

Une autre approche pour obtenir des matrices creuses serait d'utiliser la norme ℓ_1 comme fonction objectif. Cette norme a une forme particulière, qui présente des pics aux points contenant des zéros (voir figure 1.5a). Elle aura donc souvent tendance à donner un résultat contenant également des zéros. La norme ℓ_1 est de plus peu sensible aux valeurs aberrantes [16], contrairement à la norme ℓ_2 qui n'est pas robuste à un bruit non-gaussien (voir figure 1.5b).

Comme cité précédemment, la norme ℓ_2 permet de bonnes approximations de données soumises à un bruit gaussien. La norme ℓ_1 permet quant à elle de bonnes approximations lorsque les données sont soumises à un bruit laplacien [12, 15, 16]. Ces avantages de la norme ℓ_1 sur la norme ℓ_2 sont également observés dans le cas de la NMF, comme l'illustre la figure 1.6.



(a) Représentation Graphique des Normes ℓ_1 et ℓ_2 [23].



(b) Régressions Linéaires de Données avec les Normes ℓ_1 et ℓ_2 [16].

FIGURE 1.5 – Illustration des Avantages de la Norme ℓ_1 par rapport à la Norme ℓ_2.

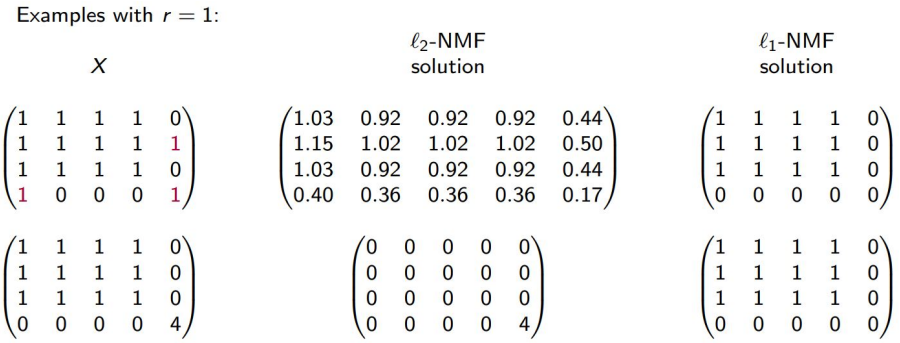


FIGURE 1.6 – Illustration des Avantages de la l1-NMF sur la l2-NMF [23].

Une variante de la NMF utilisant la norme ℓ_1 pourrait alors être un meilleur candidat afin d'effectuer la factorisation de matrices creuses ou de matrices soumises à des bruits non-gaussiens. Ce modèle ℓ_1-NMF utilisera donc la fonction d'optimisation suivante :

$$\min_{W,H} f(W,H) = ||X - WH||_1 = \sum_{i,j} |X - WH|_{i,j} \quad s.t. \quad W,H \geq 0. \tag{1.10}$$

C'est ce modèle ℓ_1-NMF que nous utiliserons dans la suite du travail. Il est à noter que cette fonction d'optimisation est NP-HARD même dans le cas où $r = 1$ [11], ce qui n'est pas le cas pour la ℓ_2-NMF, pour laquelle il existe des algorithmes de résolution polynomiaux dans le cas où $r = 1$ [1].

1.7 État de l'Art

La littérature cite souvent la norme ℓ_1 pour de la factorisation de matrices (MF). La contrainte de non-négativité n'est donc pas présente, mais nous verrons que les méthodes de résolution peuvent être tout de même utilisées avec de légères modifications. Afin de distinguer la NMF de la MF, nous utiliserons les notations suivantes dans le cas de la factorisation de matrices classiques :

$$\min_{U,V} E(U,V) = ||M - UV||_1 = \sum_{i,j} |M - UV|_{i,j}, \tag{1.11}$$

avec $M \in \mathbb{R}^{m\times n}$, $U \in \mathbb{R}^{m\times r}$ et $V \in \mathbb{R}^{r\times n}$ avec $r \ll \min(m,n)$. Cette fonction d'optimisation est évidemment équivalente à la fonction d'optimisation présentée à l'équation (1.10), sans la contrainte de non-négativité.

Cette forme de factorisation de matrices est préférée à la factorisation en norme ℓ_2 lorsque les données sont soumises à un bruit Laplacien [12, 15, 16] dont la formule générale de distribution peut s'écrire

$$f(x|\mu) \sim \exp\left(-\frac{|x-\mu|}{b}\right). \tag{1.12}$$

Les méthodes de factorisation que nous pouvons trouver utilisent principalement la descente par bloc, et donc un algorithme similaire à l'Algorithme 1 servant à résoudre la ℓ_2-NMF. Cette méthode permet de simplifier le problème en optimisant successivement les fonctions suivantes :

$$\begin{aligned} V_{t+1} &= \min_V ||M \quad - U_t V_t||_1, \\ U_{t+1} &= \min_U ||M^T - V_{t+1}^T U_t^T||_1, \end{aligned} \tag{1.13}$$

avec H_t et W_t représentant la solution obtenue après t itérations. L'algorithme de descente par bloc pourra alors continuer les itérations jusqu'à, par exemple, la convergence de la fonction objectif

$$\frac{|E(U_t,V_t) - E(U_{t+1},V_{t+1})|}{|E(0,0)|} < \xi, \tag{1.14}$$

avec ξ un paramètre de convergence, pouvant être fixé par exemple à $\xi = 0.1$ en fonction du problème.

Les deux sous-problèmes présentés à l'équation (1.13) sont symétriques et convexes [15, 16, 18] et ce même en ajoutant la contrainte de non-négativité [12]. Ces problèmes peuvent également être simplifiés en n problèmes indépendants convexes, avec v_j et m_j respectivement la j-ème colonne de V et de M :

$$v_j = \min_x ||U_{t-1}x - m_j||_1. \tag{1.15}$$

Ce problème étant convexe, il existe alors des méthodes d'optimisation permettant de trouver la solution optimale. Le minimum global peut alors être trouvé à l'aide de l'optimisation linéaire (*Linear Programming*) [16] de la manière suivante

$$\begin{aligned} \min_{x,d} \quad & \vec{e}^T d \quad s.t \\ & U_{t-1}x - m_j \geq -d, \\ & U_{t-1}x - m_j \leq d, \end{aligned} \tag{1.16}$$

avec $\vec{e}$ un vecteur colonne ne contenant que des 1.

Une méthode d'optimisation quadratique permet également de résoudre le problème en approximant la fonction norme ℓ_1 [15, 16], mais nous ne nous attarderons pas dessus ici.

D'autres méthodes d'optimisation existent et se basent sur de la descente de coordonnées. Ce type de méthodes ne permet pas d'atteindre le minimum global pour chaque bloc individuel de l'équation (1.13), mais permet d'approcher le minimum de $E(U,V)$ de l'équation (1.11) tout comme le font les méthodes exactes décrites ci-dessus.

La descente de coordonnées va alors optimiser individuellement chaque point de V noté $V_{k,j}$. En notant Z la matrice de résidu tel que $Z_{:,j} = M_{:,j} - \sum_{k=1,k\neq k}^{r} U_{:,k}V_{k,:} \quad \forall j = 1,...n$, la fonction à minimiser sera alors :

$$\min_{V_{k,j}} ||Z_{:,j} - U_{:,k}V_{k,j}||_1. \tag{1.17}$$

Cette fonction objectif est une fonction convexe non-différentiable composée de plusieurs parties linéaires. Elle peut être réécrite sous la forme générale appelée "*weighted median*" :

$$\min_{\alpha\geq 0} ||x - \alpha y||_1 = \sum_i |x_i - \alpha y_i| = \sum_i |y_i| \left| \frac{x_i}{y_i} - \alpha \right|. \tag{1.18}$$

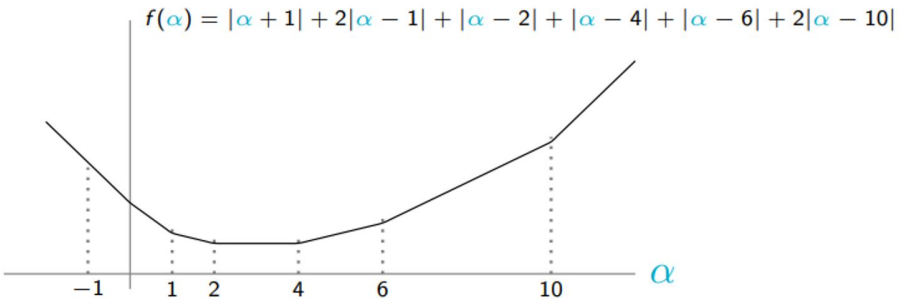


FIGURE 1.7 – Exemple de la Forme Générale *weighted median* de la Fonction d'Optimisation par Descente de Coordonnées en de la ℓ_1-NMF [23]

Cette fonction possède plusieurs *breakpoints*, aux valeurs $\frac{x_i}{y_i}$, $i = 1,...m$ et $y_i \neq 0$. Au vu de la forme de la fonction (voir la figure 1.7), nous pouvons déduire que le minimum de la fonction se trouvera à l'un de ces *breakpoints*. Une méthode pour trouver le minimum de la fonction serait alors de calculer la valeur de *weighted median* pour tous les points $\frac{x_i}{y_i}$. L'algorithme obtenu aurait alors une complexité $\mathcal{O}(n^2)$ (il faut calculer n fois la fonction d'optimisation (1.18), possédant une somme de n éléments).

Il existe cependant un algorithme plus rapide, avec une complexité $\mathcal{O}(n\log(n))$, [13] (voir Algorithme 3). Cet algorithme trie d'abord les breakpoints $\frac{x_i}{y_i}$. Il calcule ensuite la pente après avoir atteint chacun de ces breakpoints. Une fois que la pente devient positive, l'algorithme sait que le breakpoint optimal est atteint.

Un algorithme en complexité $\mathcal{O}(n)$ existe également [4]. Cet algorithme est cependant beaucoup plus lourd à mettre en place.

Algorithme 3 α = weighted_median(x, y)

1: INPUT : $x \in \mathbb{R}^n, y \in \mathbb{R}^n$
2: OUTPUT : $\text{argmin}_{\alpha\geq 0} \sum_i |x_i - \alpha y_i|$
3: $S \leftarrow \{\frac{x_i}{y_i} | y_i \neq 0\}$
4: $[S, index] \leftarrow sort(S)$
5: $y \leftarrow y[index]$
6: $k \leftarrow$ smallest k $| \sum_{i=1}^{k} y_i \geq \frac{\sum_{i=1}^{n} y_i}{2}$
7: $\alpha \leftarrow S_k$

Cette méthode peut aussi bien être utilisée pour de la factorisation de matrices en norme ℓ_1 [16] que pour de la factorisation non-négative en norme ℓ_1 [12]. Dans le cas non-négatif, la valeur $V_{k,j}$ sera bornée à zéro et donc $V_{k,j} = \max(0, weighted\_median(Z_{:,j}, U_{:,k}))$.

Une dernière méthode de résolution par descente de coordonnées existe dans la littérature [12]. Elle a pour but de lisser la fonction objectif afin qu'elle soit différentiable. Cette méthode utilise alors une fonction d'approximation de la valeur absolue [21]. Une fois que la fonction est différentiable, il est alors possible d'effectuer une descente de gradient afin de trouver le minimum de $V_{k,j}$. La fonction étant convexe, une contrainte de non-négativité peut être ajoutée en se limitant au domaine des réels positifs. La méthode de smoothing est cependant plus lourde à mettre en place que la méthode *weighted median*. De plus, nous verrons par la suite des méthodes basées sur *weighted median* et améliorant nos résultats pour nos jeux de données.

Chapitre 2

Factorisation Non-Négative de Matrices en norme ℓ_1 – ℓ_1-NMF

2.1 Introduction

Dans le chapitre précédent, nous avons déduit que la ℓ_1-NMF était un bon candidat pour l'extraction de thème et la classification non-supervisées de textes. Ce chapitre propose une première méthode de résolution de la ℓ_1-NMF. Cette méthode sera optimisée afin d'obtenir une meilleure complexité algorithmique pour des matrices creuses.

Après avoir exposé nos premiers algorithmes, nous vérifierons si notre modèle produit théoriquement un résultat creux. Nous analyserons ensuite les résultats de la ℓ_1-NMF sur des jeux de données synthétiques binaires et sur nos jeux de données réels.

2.2 Résolution de la ℓ_1-NMF

Comme vu dans la section 1.7, la fonction d'optimisation (Équation (1.10)) devient convexe si la matrice W ou la matrice H est fixée. Notre algorithme reposera alors sur de la descente par bloc, comme décrit dans l'Algorithme 1. Tout comme pour la norme ℓ_2, l'égalité $||X - WH||_1 = ||X^T - H^T W^T||_1$ est vérifiée et ajoute une symétrie entre W et H. Nous pouvons alors obtenir la fonction objectif de descente par bloc :

$$\min_{H \geq 0} f(H) = ||X - WH||_1 = \sum_{i,j} |X - WH|_{i,j} = \sum_{i,j \in \kappa_+} |X - WH|_{i,j} + \sum_{i,j \in \kappa_0} |WH|_{i,j} \tag{2.1}$$

avec $\kappa_+ = \{(i,j) | X_{i,j} > 0\}$ et $\kappa_0 = \{(i,j) | X_{i,j} = 0\}$ étant deux sous-ensembles complémentaires à l'ensemble des combinaisons d'indices (i,j). L'ensemble κ_+ représentera alors l'ensemble des indices pour lesquels les entrées de X sont non-nulles et l'ensemble κ_0 représente l'ensemble des indices pour lesquelles les entrées de X sont nulles.

Nous allons ensuite nous concentrer sur de la descente de coordonnées afin d'utiliser la méthode *weighted median* (voir Algorithme 3). Nous minimiserons individuellement chaque variable $H_{p,q}$. Afin d'obtenir une équation de la forme générale de *weighted median* (voir Équation (1.18)), nous réécrirons la fonction d'optimisation en descente de coordonnées de la manière suivante :

$$\begin{aligned}\min_{H_{p,q}\geq 0} f(H_{p,q}) &= \sum_{i\in\kappa_{+_q}} |X-WH|_{i,q} + \sum_{i\in\kappa_{0_q}} |WH|_{i,q} \\ &= \sum_{i\in\kappa_{+_q}} |X_{i,q} - \sum_{\substack{k=1\\k\neq p}}^{r} (W_{i,k}H_{k,q}) - W_{i,p}H_{p,q}| + H_{p,q}\sum_{i\in\kappa_{0_q}} W_{i,p} \\ &= \sum_{i\in\kappa_{+_q}} |a_i - b_i H_{p,q}| + |0 - cH_{p,q}|\end{aligned} \tag{2.2}$$

avec :

— $\kappa_{+_q} = \{i|(i,q)\in\kappa_+\} = \{i|X_{i,j}>0, j=q\}$ un sous-ensemble de κ_+ avec l'indice q fixé ;
— $\kappa_{0_q} = \{i|(i,q)\in\kappa_0\} = \{i|X_{i,j}=0, j=q\}$ un sous-ensemble de κ_0 avec l'indice q fixé ;
— $a_i = X_{i,q} - \sum\limits_{\substack{k=1\\k\neq p}}^{r} (W_{i,k}H_{k,p})$; $b_i = W_{i,p}$ et $c = \sum\limits_{i\in\kappa_{0_q}} W_{i,p}$.

Cette fonction convexe peut alors être résolue avec l'Algorithme 3. Nous pouvons remarquer que la complexité de l'algorithme est cette fois-ci $\mathcal{O}(n_+\log(n_+))$ avec n_+ correspondant à la taille de l'ensemble κ_{+_q}. Dans le cas d'une matrice creuse, n_+ est beaucoup plus petit que n, ce qui permet d'augmenter grandement les performances de l'algorithme. C'est principalement cette raison qui nous a motivé à séparer l'ensemble des indices de X en deux sous-ensembles complémentaires κ_+ et κ_0.

L'Algorithme 4 reprend alors l'équation (2.2) décrite ci-dessus et la fonction de résolution *weighted median* (voir Algorithme 3). De plus, certains calculs sont placés de telle sorte à ne pas recalculer inutilement certaines valeurs et ainsi continuer à améliorer le temps de traitement de l'algorithme.

Algorithme 4 $H^{(i)}$ = sparse_l1_coordinate_descent($X, W, H^{(i-1)}$)

INPUT : $X\in\mathbb{R}^{m\times n}, W\in\mathbb{R}_+^{m\times r}, H^{(i-1)}\in\mathbb{R}^{r\times n}$
OUTPUT : $H^{(i)}\in\mathbb{R}^{r\times n}$
$H^{(i)} \leftarrow H^{(i-1)}$
$s_p = \sum_{j=1}^{m} W_{j,p} \qquad \forall p = 1,...r$
for $q = 1 : n$ **do**
    $\kappa_{+_q} = \{k \mid X_{k,q}\neq 0\}$
    $v = W_{\kappa_{+_q},:}H^{(i)}_{:,q}$
    for $p = 1 : r$ **do**
        $a \leftarrow X_{\kappa_{+_q},p} - v + W_{\kappa_{+_q},p}H^{(i)}_{p,q}$
        $b \leftarrow W_{\kappa_{+_q},p}$
        $c \leftarrow s_p - \sum_{k\in\kappa_{+_q}} W_{k,p}$
        $H^{(i)}_{p,q} \leftarrow \max(0, weighted\_median([a \;\; 0],[b \;\; c]))$
        $v \leftarrow v + W_{\kappa_{+_q},p}(H^{(i)}_{p,q} - H^{(i-1)}_{p,q})$
    end for
end for

Si plusieurs itérations de l'Algorithme 4 sont effectuées, il est encore possible d'améliorer le temps de traitement. En effet, la valeur κ_{+_q} ne varie pas avec les itérations. Elle peut donc être calculée hors de la boucle de descente par bloc et être injectée dans l'Algorithme 4. Nous économisons ainsi $n*m$ opérations par itération.

2.3 Condition pour que la ℓ_1-NMF produise des Matrices W et H Creuses

Notre objectif est d'approximer une matrice X creuse par une matrice WH. Pour respecter les caractéristiques de X, nous voulons alors prédire une matrice WH creuse.

Il est alors intéressant d'analyser les conditions pour que la ℓ_1-NMF produise une matrice WH creuse, et donc des matrices W et H creuses. Par définition, la matrice H sera creuse si la plupart de ses éléments $H_{p,q}$ sont nuls (par symétrie, les conditions sont similaires pour W et H).

2.3.1 Première Condition pour des Matrices Creuses

En reprenant l'Équation (1.18) et l'Algorithme 3, nous pouvons réfléchir à une condition pour laquelle la variable α (et donc $H_{p,q}$) serait nulle. Les deux arguments de la fonction *weighted median* sont deux vecteurs x et y de même dimension.

En reprenant l'Équation (2.2) et en supposant des $H_{k,q} = 0$ pour tout $k \neq p$, nous pouvons voir que le vecteur x est composé d'un unique élément nul et des éléments non nuls de la matrice X. Le vecteur y est quant à lui composé d'un élément égale à $\sum_{i \in \kappa_{0_q}} W_{i,p}$ et des valeurs $W_{i,p}$ avec $i \in \kappa_{+_q}$.

Dû au caractère non-négatif des matrices X et W, x et y seront alors également non-négatifs, et donc $\frac{x_i}{y_i}$ sera non-négatif pour tout i. De plus, x possède un unique élément nul, il existe donc un seul $\frac{x_i}{y_i}$ nul. L'Algorithme 3 trie les éléments $\frac{x_i}{y_i}$. Donc, pour que $\alpha = 0$, il faudra que $k = 1$ vérifie l'inéquation $\sum_{i=1}^{k} y_i \geq \frac{1}{2} \sum_i y_i$, et donc :

$$y_1 \geq \frac{1}{2} \sum_i y_i. \tag{2.3}$$

Le vecteur y ayant été trié par ordre de $\frac{x_i}{y_i}$, y_1 correspond à $\sum_{i \in \kappa_{0_q}} W_{i,p}$. La condition pour que $H_{p,q}$ soit nul est alors :

$$\sum_{i \in \kappa_{0_q}} W_{i,p} \geq \frac{1}{2} \sum_{i=1}^{m} W_{i,p}. \tag{2.4}$$

Pour une matrice initiale X creuse, il est très probable que cette condition soit respectée pour une matrice W non-creuse, le sous ensemble κ_{0_q} composant une grande partie de l'ensemble des indices de $j = 1, ...n$. La ℓ_1-NMF produira alors des matrices W et H creuses si la matrice X est elle-même creuse.

Nous avons supposé que $H_{k,p} = 0$ pour $k \neq p$. Ces termes n'ont d'impacts que sur les valeurs du vecteur x. Étant donné que les matrices W et H sont non-négatives, une valeur de $H_{k,p} \neq 0$ ne pourra que diminuer les valeurs de x. Cela peut avoir pour effet de diminuer le minimum de l'Équation (2.2). En ajoutant un critère de non-négativité sur l'équation, réduire la valeur du minimum ne diminue pas les chances de produire une valeur de $H_{p,q}$ nulle, et la ℓ_1-NMF produira donc toujours des matrices creuses.

2.3.2 Deuxième Condition pour des Matrices Creuses

Nous avons vu que si la condition de l'équation (2.4) était respectée, alors la valeur de $H_{p,q}$ sera 0. Nous pouvons voir qu'il existe une autre condition équivalente, mais plus pratique dans certains cas.

En démarrant de l'Équation (2.2) à laquelle nous considérons $H_{k,q} = 0$ pour tout $k \neq p$ comme précédemment, nous obtenons la fonction d'optimisation suivante :

$$\alpha = \underset{H_{p,q} \geq 0}{\operatorname{argmin}} f(H_{p,q}) = \sum_{i \in \kappa_{+_q}} |X_{i,q} - W_{i,p} H_{p,q}| + H_{p,q} \sum_{i \in \kappa_{0_q}} W_{i,p}. \tag{2.5}$$

Nous avons montré plus haut avec l'Algorithme 3 que la solution de cette équation était à l'un des breakpoints $\frac{x_i}{y_i}$, et donc dans notre cas dans l'ensemble $\{0, \frac{X_{i,q}}{W_{i,p}} \; \forall i \in \kappa_{+_q}\}$. La fonction étant convexe et $\frac{X_{i,q}}{W_{i,p}} > 0$ pour tout $i \in \kappa_{+_q}$, nous pouvons déduire que $\alpha > 0 \iff f(0^+) < f(0) \iff \frac{\delta f}{\delta H_{p,q}}(0^+) < 0$. En $H_{p,q} = 0^+$, les valeurs absolues de la fonction peuvent être retirées. Nous pouvons alors calculer la nouvelle condition pour produire des matrices creuses :

$$\begin{aligned} \lim_{H_{p,q} \to 0^+} f(H_{p,q}) &= \sum_{i \in \kappa_{+_q}} (X_{i,q} - W_{i,p} H_{p,q}) + H_{p,q} \sum_{i \in \kappa_{0_q}} W_{i,p}, \\ \lim_{H_{p,q} \to 0^+} \frac{\delta f}{\delta H_{p,q}}(H_{p,q}) &= -\sum_{i \in \kappa_{+_q}} W_{i,p} + \sum_{i \in \kappa_{0_q}} W_{i,p} < 0, \\ &\Rightarrow \sum_{i \in \kappa_{0_q}} W_{i,p} < \sum_{i \in \kappa_{+_q}} W_{i,p}. \end{aligned} \tag{2.6}$$

La condition pour que $H_{p,q}$ soit nul sera alors

$$\sum_{i \in \kappa_{0_q}} W_{i,p} \geq \sum_{i \in \kappa_{+_q}} W_{i,p}. \tag{2.7}$$

Le sous-ensemble κ_{+_q} est beaucoup plus petit que le sous-ensemble κ_{0_q} pour des matrices X et W creuses. Dès lors, α aura une grande probabilité d'être nulle si la colonne $W_{:,p}$ n'est pas creuse. Les matrices W (et H par symétrie) auront alors de bonne chance d'être creuse.

Nous pouvons montrer l'équivalence entre la première condition (2.4) et la seconde condition (2.7). Pour rappel, par définition, l'union des ensembles κ_{+_q} et κ_{+_0} correspond à l'ensemble des indices $i = 1, ...m$:

$$\begin{aligned} \sum_{i \in \kappa_{0_q}} W_{i,p} &\geq \frac{1}{2} \sum_{i=1}^{m} W_{i,p} \\ \sum_{i \in \kappa_{0_q}} W_{i,p} &\geq \frac{1}{2} \left(\sum_{i \in \kappa_{0_q}} W_{i,p} + \sum_{i \in \kappa_{+_q}} W_{i,p} \right) \\ \sum_{i \in \kappa_{0_q}} W_{i,p} - \frac{1}{2} \sum_{i \in \kappa_{0_q}} W_{i,p} &\geq \frac{1}{2} \sum_{i \in \kappa_{+_q}} W_{i,p} \\ \sum_{i \in \kappa_{0_q}} W_{i,p} &\geq \sum_{i \in \kappa_{+_q}} W_{i,p} \end{aligned} \tag{2.8}$$

La première condition montrée par l'équation (2.4) et la deuxième condition montrée par l'équation (2.7) sont donc bien équivalentes. Elles permettent chacune de voir une approche différente de la condition pour produire des matrices creuses.

2.3.3 Cas d'une Matrice X binaire

Supposons le cas particulier où X est une matrice binaire, c'est-à-dire $X_{i,j} = \{0,1\}$ pour tout i, j. En partant du principe que la matrice W est également binaire, l'algorithme *weighted median* 3 nous montre que les seuls résultats possibles pour $H_{p,q}$ sont $\{0,1\}$. En effet, si X et W sont binaires, x et y seront binaires également, et donc $\frac{x_i}{y_i}$ sera binaire pour tout i. L'influence des $H_{k,q}$ avec $k \neq p$ peut créer des valeurs x_i inférieures à 0, et donc non-binaire. Cependant, le critère de non-négativité fait que ces x_i peuvent être remplacé par 0 sans impacter les résultats, et $H_{p,q}$ sera binaire.

Une variante de l'algorithme 3 existe avec une complexité $\mathcal{O}(n)$ dans le cas d'une matrice X binaire. L'algorithme mesure la taille de l'ensemble $\{i|W_{i,p} = 1, X_{i,q} = 1\}$ et celle de l'ensemble $\{i|W_{i,p} = 1, X_{i,q} = 0\}$. Si le premier ensemble est plus grand que le second, alors α sera égal à 1. Dans le cas contraire, α sera égal à 0.

Dans le cas de la ℓ_2-NMF, il est très probable de prédire des valeurs $H_{p,q}$ différentes de 0 et 1 pour une matrice binaire (voir Figure 1.6). Or, pour une matrice X binaire, nous aurons envie de prédire une matrice WH elle-même binaire.

2.3.4 Initialisation pour des Matrices Creuses

Prédire trop de zéros aura un inconvénient majeur sur la ℓ_1-NMF : l'algorithme aura tendance à prédire des lignes de H uniquement avec des zéros si la colonne de W correspondante n'améliore la fonction d'optimisation pour aucun texte. Si ce phénomène arrive, l'algorithme n'arrivera pas à retrouver une valeur intéressante et bouclera sur une ligne de H remplie de zéros (et, par symétrie, une colonne nulle de W).

Il est alors primordial de bien initialiser les matrices H_0 et W_0. Si elles sont initialisées uniquement avec des 1 ou des valeurs aléatoires, l'algorithme ℓ_1-NMF convergera le plus souvent vers des matrices nulles. Comme vu avec l'Équation (2.4), les éléments de H ont beaucoup de chance d'être nul si la matrice W n'est pas creuse.

Cette condition montre que $H_{p,q}$ a plus de chance d'être nul si la matrice X est creuse. Ce phénomène ne se produit pas pour une matrice X non-creuse. La ℓ_1-NMF produira alors de bons résultats, même avec une initialisation aléatoire [12].

Une méthode courante pour initialiser la ℓ_1-NMF est d'utiliser la ℓ_2-NMF. Les matrices sont alors initialisées en effectuant quelques itérations de ℓ_2-NMF avant de commencer les itérations de ℓ_1-NMF. C'est cette méthode que nous utiliserons dans un premier temps pour ce travail.

Nous verrons qu'il existe d'autres méthodes (voir Section 5.2.1), mais celles-ci ne sont pas vraiment abouties et mériteraient un travail y prêtant plus attention.

2.4 Résultats de la *l1*-NMF sur des Jeux de Données Synthétiques

En plus de l'approche théorique, nous pouvons comparer la ℓ_1-NMF et la ℓ_2-NMF d'un point de vue pratique sur des jeux de données synthétiques. Pour cela, nous avons construit une matrice X binaire composée de r clusters. Afin de tester à quel point la méthode est robuste au bruit, nous ajouterons une perturbation à la matrice. Cette perturbation sera définie par une probabilité p qu'une entrée $X_{i,j}$ soit changée en 1 (resp. 0) si elle était initialement un 0 (resp. 1). La figure 2.1 permet de visualiser les données synthétiques que nous utiliserons.

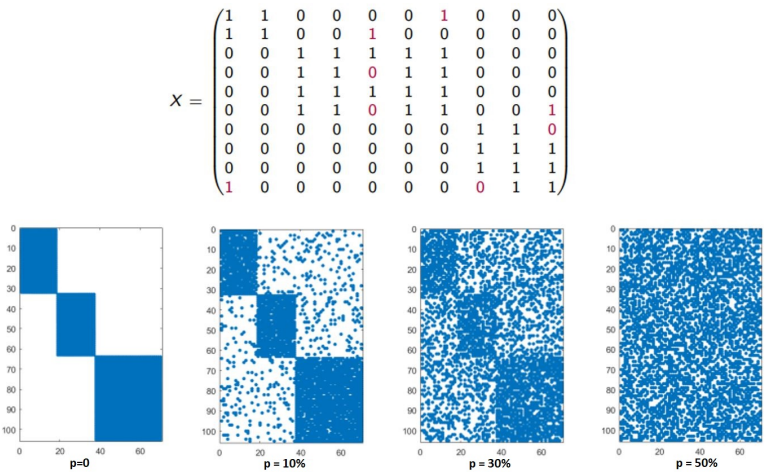


FIGURE 2.1 – Illustration du Jeu de Données Synthétiques et de l'Impact de la Perturbation [23]

Trois jeux de données synthétiques seront testés dans cette section. Ils sont tous les trois composés d'une matrice binaire $X \in \{1,0\}^{n\times m}$ avec $n = 400$ et $m = 600$. Les jeux de données possèdent respectivement $r = \{3,10,20\}$ clusters répartis équitablement dans la matrice.

Nous avons analysé l'évolution de deux métriques en fonction de la perturbation appliquée à la matrice X. La première métrique correspond à l'erreur quadratique moyenne de la matrice WH par rapport à la matrice X non-perturbée : $\frac{||X-WH||_F}{||X||_F}$ (et donc, la fonction d'optimisation de la ℓ_2-NMF pour la matrice X non-perturbée). La deuxième métrique correspond à la précision à laquelle l'algorithme assigne les points d'un même cluster ensemble. Cette métrique mérite plus d'explications et est traitée en détail dans la section 2.5.2.

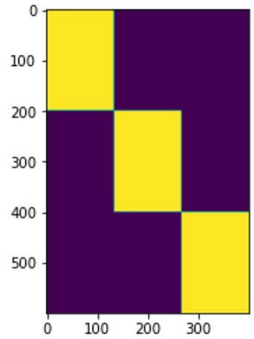
Nous testerons ces métriques sur une plage de perturbations allant de 0% (la matrice n'est pas perturbée) jusqu'à 50%, 40% et 30% pour une valeur de r respectivement de 3, 10 et 20.

Nous pouvons constater d'après les résultats sur les figures 2.2, 2.3 et 2.4 que les prédictions de la ℓ_1-NMF sont clairement meilleures que celles de la ℓ_2-NMF.

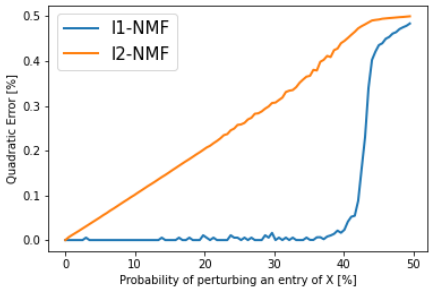
Pour les graphiques de précision, la courbe de la ℓ_1-NMF est pratiquement toujours au-dessus de celle de la ℓ_2-NMF. Nous pouvons d'ailleurs constater que cette dernière diminue plus ou moins proportionnellement à la perturbation, tandis que la courbe de précision de la ℓ_1-NMF a l'air presque constante jusqu'à un certain seuil. Ce seuil correspond plus ou moins au moment où il y a plus d'entrées perturbées dans la matrice X que d'entrées non perturbées (soit 33%, 10% et 5% respectivement pour r =3, 10 et 20). Une fois ce seuil atteint, la précision décroît cependant plus ou moins brutalement pour devenir assez vite équivalente à la précision de la ℓ_2-NMF.

D'un point de vue de l'erreur quadratique, nous pouvons remarquer que la ℓ_1-NMF produit généralement une matrice WH plus proche de la matrice X initial que la ℓ_2-NMF. Les courbes d'erreur quadratique semblent être corrélées aux courbes de précision, ce qui parait assez cohérent.

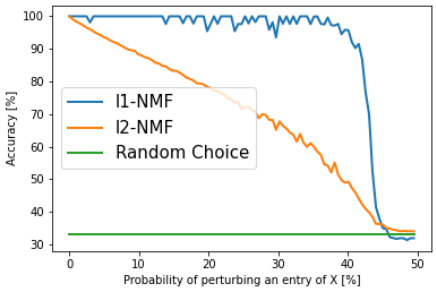
Ces résultats montrant la supériorité de la ℓ_1-NMF sont tout de même à nuancer. Les jeux de données synthétiques utilisées sont des matrices binaires, ce qui est le domaine de prédilection théorique de la ℓ_1-NMF et qui est un domaine dans lequel la ℓ_2-NMF ne s'en sort pas spécialement bien.



(a) Jeu de Données Synthétiques ($r = 3$).

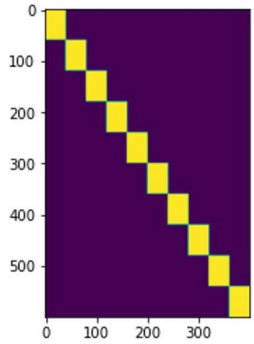


(b) Erreur Quadratique en Fonction de la Perturbation sur les Données Synthétiques ($r = 3$).

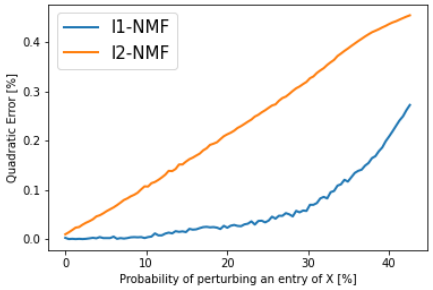


(c) Précision en Fonction de la Perturbation sur les Données Synthétiques ($r = 3$).

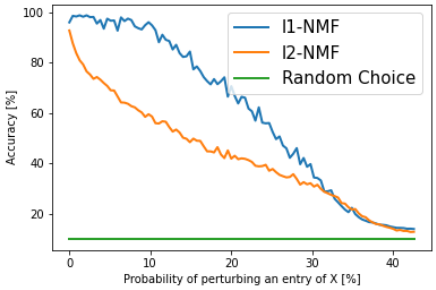
FIGURE 2.2 – Comparaison des Algorithmes ℓ_1-NMF et ℓ_2-NMF sur un Jeu de Données Synthétiques avec $r = 3$.



(a) Jeu de Données Synthétiques ($r = 10$).

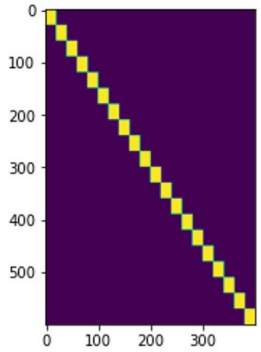


(b) Erreur Quadratique en Fonction de la Perturbation sur les Données Synthétiques ($r = 10$).

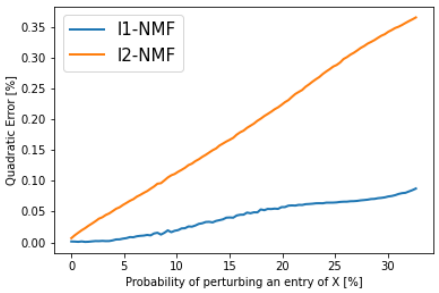


(c) Précision en Fonction de la Perturbation sur les Données Synthétiques ($r = 10$).

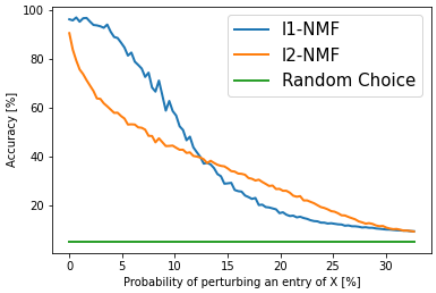
FIGURE 2.3 – Comparaison des Algorithmes ℓ_1-NMF et ℓ_2-NMF sur un Jeu de Données Synthétiques avec $r = 10$.



(a) Jeu de Données Synthétiques ($r = 20$).



(b) Erreur Quadratique en Fonction de la Perturbation sur les Données Synthétiques ($r = 20$).



(c) Précision en Fonction de la Perturbation sur les Données Synthétiques ($r = 20$)

FIGURE 2.4 – Comparaison des Algorithmes ℓ_1-NMF et ℓ_2-NMF sur un Jeu de Données Synthétiques avec $r = 20$.

2.5 Résultats de la ℓ_1-NMF sur des Jeux de Données Réels

2.5.1 Présentation des Jeux de Données

Comme précisé dans la section 1.2, le problème sur lequel nous allons nous concentrer est de l'extraction de thèmes de textes ainsi que la classification non-supervisée de ces textes. Les textes de nos jeux de données seront représentés par une matrice $X \in \mathbb{R}_+^{m \times n}$, telle que la valeur $X_{i,j}$ représente le nombre d'occurrences du mot i dans le texte j.

Dans les jeux de données que nous utiliserons, nous connaissons déjà le nombre r de thèmes présents dans ces textes. Il est à noter que dans beaucoup de cas réels, r n'est pas connu. Déduire la meilleure valeur possible pour r est alors un problème en soi [27].

De plus, pour nos jeux de données, nous aurons accès à un vecteur $y \in \mathbb{R}^n$ correspondant à la classe de chaque texte j. Ce vecteur nous permettra de mesurer à quel point nous arrivons à regrouper les textes proches entre eux dans un même groupe.

Dans le cadre de ce travail, nous travaillerons d'abord sur le jeu de données "TDT2" [7]. Ce jeu de données a comme avantage de posséder, en plus de la matrice X, un dictionnaire de mots correspondant aux index i de la matrice. Nous pourrons ainsi analyser plus facilement les résultats. Le jeu de données TDT2 ne possède cependant pas de vecteur y, nous ne pourrons donc pas mesurer la précision du modèle.

Nous travaillerons également sur un ensemble d'autres jeux de données [31], qui ne possèdent que la matrice X et le vecteur y.

Les jeux de données peuvent être décrits par les valeurs suivantes :

— n, le nombre de textes que contient le jeu de données.

— m, le nombre de mots que contient le jeu de données.

— r, le nombre de classes du jeu de données. Ils sont ici interprétés comme le nombre de thèmes des textes du jeu de données.

— S_X, la "*sparsity*" de la matrice X. Cette valeur représente à quel point la matrice X est remplie de zéros. Si S_X est 0, alors la matrice ne possédera aucun zéro. Si S_X vaut 1, alors la matrice ne sera remplie que de zéros. La "*sparsity*" peut être calculée comme suit :

$$S_X = \frac{|\kappa_0|}{n * m} \qquad with \quad \kappa_0 = \{(i,j)|X_{i,j} = 0\}. \tag{2.9}$$

— *Balance*. La *Balance* d'un jeu de données est décrite comme le ratio entre le nombre de documents de la plus petite classe et le nombre de documents de la plus grande classe. Une *Balance* de 1 décrit une répartition équilibrée des textes dans les classes, et 0 décrit une répartition très inéquitable entre les tailles des classes.

Le tableau 2.1 reprend les jeux de données qui nous serviront de comparaison ainsi que leurs caractéristiques décrites ci-dessus. Nous pouvons voir que, pour nos jeux de données, la plupart des matrices sont très creuses, souvent composées à plus de 95% de zéros. Nous pouvons également noter que certains jeux de données sont équilibrés, et d'autres non. Nous pourrons ainsi voir si la ℓ_1-NMF permet de bien classifier des jeux de données inéquilibrés.

Dataset Name	n	m	r	S_X [%]	*Balance*
TDT2	10212	36771	20	99,647	/
NG20	19949	43586	20	99,819	0.991
ng3sim	2998	15810	3	99,291	0.998
classic	7094	41681	4	99,924	0.323
ohscal	11162	11465	10	99,473	0.437
k1b	2340	21819	6	99,407	0.043
hitech	2301	10080	6	98,571	0.192
reviews	4069	18483	5	98,991	0.098
sports	8580	14870	7	99,144	0.036
tr11	414	6424	9	95,693	0.046
tr23	204	5831	6	93,409	0.066
tr41	878	7453	10	97,392	0.037
tr45	690	8261	10	96,603	0.088

TABLE 2.1 – Présentation des Jeux de données [7] [31] et de leurs caractéristiques.

2.5.2 Présentation des Métriques

Pour des algorithmes de Machine Learning, les métriques ont plusieurs utilités. Elles permettent d'évaluer l'efficacité et la performance des modèles, de comparer les modèles ou de comparer les performances d'un même modèle sur plusieurs jeux de données.

Certaines métriques sont assez usuelles, comme la valeur de la fonction objectif, appelée erreur en norme ℓ_1 (err) dans la suite du document, ou le temps de traitement (t). D'autres métriques moins habituelles méritent plus de précisions.

Précision - *Accuracy*

La première métrique que nous allons mesurer correspond à la précision à laquelle la NMF va regrouper ensemble les textes d'une même classe. Dans nos jeux de données, chaque texte q possède une classe y_q. Nous supposons que les différentes classes de y sont des nombres naturels entre 1 et r.

Dans le cas de la NMF, nous pouvons interpréter que pour un texte q, l'index du maximum de $H_{:,q}$ correspondrait à sa classe prédite : $\tilde{y}_q = \text{argmax}_p\ H_{p,q}$. Il faut cependant au préalable normaliser la matrice W, pour que les coefficients d'une colonne de H soient à la même échelle. La manière la plus simple de faire est de normaliser W après chaque $update\_W$ dans l'Algorithme 1. Nous effectuerons une normalisation *min-max* de la matrice W, en considérant que le minimum d'une colonne de W vaut zéro :

$$W_{:,p} \leftarrow \frac{W_{:,p}}{\max_i W_{i,p}} \qquad \forall p \in 1,...r. \tag{2.10}$$

Cette normalisation respecte le critère de non-négativité sur W. Une autre normalisation possible serait la mise à l'échelle, qui pose la somme d'une colonne de W à 1 : $\frac{W_{:,p}}{\sum_i W_{i,p}}$. Cependant, cette normalisation privilégie les thèmes composés d'un unique mot, contrairement à la normalisation *min-max* qui permet de privilégier les thèmes ayant plus de mots.

Avec cette définition de $\tilde{y}$, la NMF peut alors être interprétée comme de la classification. Cependant, étant donné qu'il s'agit de classification non-supervisée, nous ne pouvons pas mesurer la précision en vérifiant si $\tilde{y}_k = y_k$, car les classes assignées par la NMF ne sont le plus souvent pas dans le même ordre que les classes initiales.

Pour palier à ce problème, nous pouvons vérifier quelle est la meilleure manière d'associer les classes du vecteur $\tilde{y}$ aux classes du vecteur y. Ce problème est un problème d'affectation. La matrice de coût correspondant à ce problème vérifie le nombre de fois que la classe prédite serait incorrecte en échangeant les classes entre elles. La matrice C est alors construite de la manière suivante :

$$C_{i,j} = |\{\tilde{y}_k \neq j | y_k = i \quad \forall k = 1,...r\}| \qquad \forall i,j \in [1;r]. \tag{2.11}$$

Le problème d'affectation peut ensuite être résolu par l'algorithme *Munkres* (aussi appelé algorithme hongrois) [19].

L'algorithme 5 permet de calculer à partir du vecteur de classe y et de la matrice H la précision à laquelle un algorithme de NMF aura regroupé ensemble les textes d'une même classe. Dans cet algorithme et dans la suite du document, $0^{N\times M}$ sera considéré comme une matrice remplie de 0 de N lignes et M colonnes.

La métrique de précision sera notée "*Acc*" dans la suite du document. Elle peut varier de 0 à 100%.

Algorithme 5 Acc = Accuracy(y,H)

INPUT : $y \in \mathbb{N}^n, H \in \mathbb{R}^{r\times n}$
OUTPUT : $Acc \in \mathbb{R} \in [0;1]$
$C = 0^{r\times r}$
for $k = 1 : n$ **do**
    $i \leftarrow y_k$
    for $j = 1 : r$ **do**
        if $\max_l H_{l,k} = 0$ **or** $\text{argmax}_l H_{l,k} \neq j$ **then**
            $C_{i,j} \leftarrow C_{i,j} + 1$
        end if
    end for
end for
$best\,jobs = Munkres(C)$
$Acc = 1 - \sum_i (C_{i,best\,jobs_i})/n$

Les algorithmes de NMF étant ici interprétés comme de la classification non-supervisée, leurs précisions seront souvent inférieures aux précisions obtenues à l'aide d'algorithmes de classification supervisée. Nous comparerons alors les méthodes ℓ_1-NMF entre elles ou avec d'autres modèles de NMF. Les résultats des autres modèles sont récupérés dans l'article [10].

Proportion de colonnes non-nulles de H et de W

Cette métrique mesure la proportion de colonnes de la matrice H possédant au moins un élément non-nul. Une colonne $H_{:,j}$ remplie de zéros signifie que le texte j n'appartient à aucun thème selon l'algorithme. Le texte ne peut alors pas être classifié.

Au plus la proportion de colonnes non-nulles de H est grande, au plus notre algorithme aura réussi à classifier des textes. Une valeur élevée de cette métrique est donc positive. Cette métrique doit dans l'idéal avoir une valeur égale à 100%, mais nous verrons dans la Section 2.5.3 que ce n'est pas souvent le cas.

La métrique sera notée "$|\vec{e}H|_0$" dans la suite du document. Cette métrique peut varier entre 0 et 100%. Elle peut être calculée de la manière suivante :

$$|\vec{e}H|_0 = \frac{\left|\left\{\sum_{i=1}^{r} H_{i,j} \neq 0 \qquad \forall j = 1,...n\right\}\right|}{n}. \tag{2.12}$$

La proportion de colonnes non-nulles de W permet de vérifier qu'il n'y a aucune colonnes de W remplies de zéros. Si cela arrive, nous nous privons d'un thème pour nos textes, ce qui n'est pas souhaitable.

La métrique sera notée "$|\vec{e}W|_0$" dans la suite du document. Cette métrique peut varier entre 0 et 100%. Elle doit dans l'idéal valoir 100%. Elle peut être calculée de la manière suivante :

$$|\vec{e}W|_0 = \frac{\left|\left\{\sum_{i=1}^{m} W_{i,j} \neq 0 \quad \forall j = 1,...r\right\}\right|}{r}. \tag{2.13}$$

Précision sans les colonnes nulles de H

Les colonnes nulles de H ne peuvent pas être classifiées, elles influent donc négativement la valeur de la métrique Acc. Nous pouvons donc calculer la précision de l'algorithme si nous excluons les colonnes nulles de H du calcul.

La métrique sera notée "Acc_0" dans la suite du document. Cette métrique peut varier entre 0 et 100%. Elle peut être calculée de la manière suivante :

$$Acc_0 = \frac{Acc}{|\vec{e}H|_0}. \tag{2.14}$$

Il est à noter que cette métrique n'est pas très pertinente si la proportion de colonnes nulles de H est trop importante. Si 90% des textes ne sont pas classifiés, mais que les 10% des textes restants sont bien classifiés, cela n'apporte que très peu d'informations sur les thèmes des textes.

Proportion de colonnes non-nulles de W^TX

La matrice W^TX permet de montrer à quel point les mots des textes de X sont pris en compte dans la matrice W. Si pour un texte j, tous les éléments de la colonne $(W^TX)_{:,j}$ sont nuls, cela signifie qu'aucun mot présent dans le texte j n'est présent dans un thème de W. Dans ce cas, il est impossible d'attribuer un thème de W au texte j. Nous verrons dans la Section 3.3.1 que la matrice W^TX est liée à la corrélation entre les thèmes de W et les textes de X.

La métrique de proportion de colonnes non-nulles de W^TX mesure la proportion de textes représentés par matrice W. Une valeur élevée de cette métrique est donc positif. Cette métrique doit dans l'idéal avoir une valeur égale à 1. La métrique sera notée "$|\vec{e}W^TX|_0$" dans la suite du document. Cette métrique peut varier entre 0 et 100%. Elle peut être calculée de la manière suivante :

$$|\vec{e}W^TX|_0 = \frac{\left|\left\{\sum_{i=1}^{r} (W^TX)_{i,j} \neq 0 \quad \forall j = 1,...n\right\}\right|}{n}. \tag{2.15}$$

"*Sparsity*" de WH

La ℓ_1-NMF a pour but d'approximer la matrice X creuse par une matrice WH. Cette matrice WH doit dans l'idéal être creuse également. Nous pouvons même supposer qu'une bonne approximation de X doit être autant creuse que X.

La métrique "*sparsity*" de WH sera alors notée S_{WH} dans la suite du document. Elle doit dans l'idéal être proche de S_X (voir Équation (2.9)). Cette métrique peut être calculée de la manière suivante :

$$S_{WH} = \frac{\left|\left\{(i,j)|(WH)_{i,j} = 0\right\}\right|}{n * m}. \tag{2.16}$$

2.5.3 Résultats sur les Différents Jeux de Données

Nous avons donc testé les résultats de l'Algorithme 4 de résolution de ℓ_1-NMF sur les jeux de données présentés dans la section 2.5.1.

Nous pouvons commencer par analyser les résultats sur le jeu de données TDT2. Les résultats complets sont disponibles à l'Annexe A. Dans ces résultats, nous pouvons caractériser une colonne de W par sa "taille", c'est-à-dire le nombre d'éléments non-nulles de la colonne, et par les mots ayant le plus de poids dans la colonne, par exemple :

```
MOTS DU THÈME 2 - Taille = 13
1 : week        - 1.0
2 : time        - 0.268457
3 : world       - 0.246075
4 : washington  - 0.144194
5 : clinton     - 0.134416
6 : crisis      - 0.125087
7 : government  - 0.114184
8 : american    - 0.099650
9 : people      - 0.091705
10: officials   - 0.088360
```

Nous pouvons d'abord discuter des métriques $|\vec{e}H|_0$ et $|\vec{e}W^T X|_0$, valant respectivement 81.59% et 98.88%. Nous pouvons voir que presque tous les textes sont représentés par la matrice W. Même si la plupart des textes ont été classifiés par l'algorithme, environ 20% ne l'ont pas été, ce qui est plutôt négatif.

Nous pouvons analyser les résultats des colonnes de W plus en détails, et relever les points suivants :

— Regardons les résultats des colonnes $W_{:,1}$ et $W_{:,6}$:

```
MOTS DU THÈME 1 - Taille = 0
...
MOTS DU THÈME 6 - Taille = 0
```

Ces colonnes de W sont totalement vides. Comme vu dans la section 2.2, si une colonne $W_{:,p}$ ou (resp. ligne de $H_{p,:}$) est nulle, alors l'algorithme bouclera sur $W_{:,p} = 0^m$ et $H_{p,:} = 0^n$, sans pouvoir sortir de cette solution, avec 0^N un vecteur de taille N rempli de 0. Ce problème est caractérisé par une valeur de la métrique $|\vec{e}W|_0$ inférieure à 1 (dans ce cas-ci, $|\vec{e}W|_0 = 90\%$). Nous créerons dans la suite de ce travail un algorithme de réinstanciation des colonnes de W (resp. lignes de H) qui boucleraient sur des valeurs nulles (voir Section 3.2).

— Une deuxième remarque peut être faite en analysant les colonnes $W_{:,3}$ et $W_{:,8}$:

```
MOTS DU THÈME 3 - Taille = 1
1 : president   - 1.0
...
MOTS DU THÈME 8 - Taille = 1
1 : president   - 1.0
```

Ces deux colonnes sont parfaitement identiques. Il n'y a pas réellement de sens à avoir deux thèmes qui sont exactement les mêmes. Il serait préférable, si cela arrive, de réinstancier une des deux colonnes.

Nous pouvons même réfléchir à réinstancier deux colonnes de W "suffisamment similaires", mais cela nécessiterait une définition à "suffisamment similaire". Nous pourrions par exemple instaurer un seuil sur la différence en norme 1 des deux colonnes : si $||W_{:,p_1} - W_{:,p_2}||_1 < 1$ avec $p_1 \neq p_2$, nous réinstancierons alors une des deux colonnes.

Les autres jeux de données ont également été testés, les résultats sont disponibles dans le Tableau 2.2. Ce tableau reprend le nom des jeux de données, les différentes métriques présentées dans la section 2.5.2 et la précision qu'obtient l'algorithme ℓ_2-NMF. Cette dernière colonne reprend des données provenant de l'article [10].

		ℓ_1-NMF						ℓ_2-NMF
Dataset	r	Acc [%]	$\|\vec{e}H\|_0$ [%]	$\|\vec{e}W\|_0$ [%]	Acc_0 [%]	$\|\vec{e}W^TX\|_0$ [%]	S_{WH} [%]	Acc [%] [10]
NG20	20	9.1	67.53	90.0	13.47	99.72	99.989	23.08
ng3sim	3	17.38	48.0	100.0	36.21	99.6	99.964	36.06
classic	4	5.77	5.77	25.0	100.0	13.6	99.999	55.64
ohscal	10	22.71	86.19	90.0	26.35	92.31	99.983	30.50
k1b	6	68.42	96.37	100.0	71.0	100.0	99.953	59.19
hitech	6	41.81	81.83	100.0	51.09	98.52	99.906	46.94
reviews	5	58.76	74.2	100.0	79.2	100.0	99.918	51.19
sports	7	32.26	64.42	100.0	50.08	99.91	99.875	40.37
tr11	9	5.07	10.14	100.0	50.0	100.0	97.32	50.48
tr23	6	9.31	11.76	100.0	79.17	100.0	93.363	35.29
tr41	10	29.04	79.5	100.0	36.53	100.0	99.375	44.99
tr45	10	5.07	7.25	100.0	70.0	100.0	98.104	38.26

TABLE 2.2 – Comparaison des métriques présentées dans la section 2.5.2 de la ℓ_1-NMF et de la précision de ℓ_2-NMF sur les jeux de données provenant de [31].

Nous pouvons commencer par analyser les métriques de précisions. En comparant la précision de l'algorithme ℓ_1-NMF à la précision l'algorithme ℓ_2-NMF, nous voyons que celle de la ℓ_1-NMF est le plus souvent inférieure, sauf pour les jeux de données *k1b* et *reviews*.

Cela peut d'abord s'expliquer par une métrique $|\vec{e}H|_0$ souvent éloignée des 100%. Si la métrique $|\vec{e}H|_0$ ne vaut pas 1, ça signifie qu'il existe des textes qui ne peuvent pas être classifiés, et ça fera baisser la précision.

La métrique Acc_0 permet de mesurer la précision sans compter les colonnes nulles de H. Nous voyons alors des résultats bien plus proches de ceux de la ℓ_2-NMF, voire meilleurs dans beaucoup de cas. Cependant, rien ne prouve que nous aurions réussi à classifier les textes restants avec autant de précision. Il n'est donc pas forcément pertinent de juger les résultats sur cette colonne, mais elle reste représentative du potentiel de la ℓ_1-NMF.

Comme déjà spécifié, la métrique $|\vec{e}H|_0$ ne vaut jamais 100% dans notre cas. Elle est même très basse dans le cas des jeux de données *classic*, *tr11*, *tr23* et *tr45*. Avoir une proportion de colonnes nulles de H est très mauvais, car elle signifie que nous ne pouvons pas donner de thème à notre texte pour le classifier.

La métrique $|\vec{e}W|_0$ vaut quant à elle souvent 100%, ce qui est une bonne chose. Cependant, les cas où elle ne vaut pas 100% sont problématiques également. Cela signifie qu'un thème n'est composé d'aucun mot, et que nous nous privons virtuellement d'un thème pour classifier nos textes.

Pour finir, nous voyons que les métriques $|\vec{e}W^TX|_0$ et S_{WH} sont plutôt bonnes (sauf pour le jeu de données *classic*) et qu'elles sont proches des résultats attendus pour ces métriques.

2.6 Conclusions

Après avoir élaboré et testé la ℓ_1-NMF, nous pouvons commencer par identifier une liste de problèmes principaux du modèle :

1. La ℓ_1-NMF produit des colonnes de W totalement nulles, et boucle sur ce résultat de colonnes. Ce problème sera discuté dans la Section 3.2 et une solution sera proposée.
2. La ℓ_1-NMF a tendance à produire des colonnes de W identiques ou presque.
3. La ℓ_1-NMF produit une trop grande proportion de colonnes nulles de H. Ce problème sera discuté dans le Chapitre 4 et plusieurs solutions seront suggérées.

Ces problèmes ont un impact négatif sur la classification non-supervisée, qui est notre principal élément de comparaison. Même si la NMF a pour but principal la factorisation de matrices et pas la classification non-supervisée, les problèmes exposés ci-dessus impactent tout autant négativement cette factorisation.

Nous pouvons cependant noter que la ℓ_1-NMF produit bien des matrices W et H creuses si la matrice X est creuse, comme avancé dans les premières parties de ce travail.

La ℓ_1-NMF produit également de très bons résultats sur des jeux de données synthétiques binaires. Il pourrait être intéressant d'analyser les résultats de l'algorithme sur des matrices binaires représentant des données réelles.

Chapitre 3

Améliorations et Variantes du Modèle de Base de la ℓ_1-NMF

3.1 Motivations

Dans ce chapitre, nous effectuerons des améliorations minimes au modèle de base décrit par l'Algorithme 4.

Nous proposerons d'abord une solution au problème des colonnes nulles de W. Ce problème, décrit dans la Section 2.5.3, illustre la tendance de la ℓ_1-NMF à boucler sur des colonnes nulles de W (et, par symétrie, des lignes de H). Ce problème diminue virtuellement la valeur de r et est négatif autant pour la factorisation de matrices que pour la classification non-supervisée.

Nous proposerons ensuite plusieurs modèles de variantes de descente de coordonnées. La création de ces modèles est motivée par l'ordre arbitraire d'optimisation des lignes dans l'Algorithme 4. Nous proposerons également un modèle susceptible d'améliorer la vitesse de convergence de l'algorithme ainsi que son temps de traitement.

3.2 Réinstanciation de W

Le modèle initial produit beaucoup de zéros. Comme spécifié dans la section 2.5.3, il arrive même que toute une colonne de W et une ligne de H bouclent sur des valeurs nulles. Lorsque ce phénomène se produit, cela a un impact très négatif sur nos prédictions, et il vaut mieux y remédier. Nous verrons dans cette section une méthode permettant de palier à ce problème.

Supposons le cas où la colonne $w = W_{:,p}$ est nulle, et par symétrie la ligne $h = H_{p,:}$ est nulle également. Une méthode pour réinstancier les valeurs de w et h serait d'effectuer la ℓ_1-NMF avec $r = 1$ et en prenant comme matrice à factoriser la matrice de résidu $Z = X - WH$. Nous pouvons alors écrire l'Algorithme 6, fonctionnant de manière similaire à l'algorithme 4.

Il est à noter qu'une légère simplification de l'Algorithme 6 peut être effectuée en considérant que $\{k \mid Z_{k,q} > 0\} \in \{k \mid X_{k,q} \neq 0\}$ (affirmation vérifiée dû au caractère non-négatif de X et WH) et en mettant l'ensemble $\kappa_+ = \{k \mid X_{k,q} \neq 0\}$ comme argument de l'algorithme.

L'Algorithme 6 nécessite en entrée un vecteur w. Ce vecteur ne peut pas être choisi aléatoirement si la matrice X est creuse, car il entraînerait certainement un vecteur h nul (voir Section 2.3). Nous utilisions avant la ℓ_2-NMF pour initialiser les matrices W et H. Cependant, c'est notamment à cause de cette méthode d'initialisation que la ℓ_1-NMF a produit des colonnes et lignes nulles de W et H. Il faut donc utiliser une méthode d'initialisation différente. De manière générale, il faudra utiliser une méthode d'initialisation différente pour la réinstanciation que pour obtenir les matrices W_0 et H_0.

Algorithme 6 h = rank1_l1_coordinate_descent(Z, w)

INPUT : $Z \in \mathbb{R}^{m \times n}, w \in \mathbb{R}^m_+$
OUTPUT : $h \in \mathbb{R}^n_+$
$h \leftarrow 0^n$
$s = \sum_{k=1}^m w_k$
for $q = 1 : n$ **do**
    $\kappa = \{k \mid Z_{k,q} > 0\}$
    $a \leftarrow Z_{\kappa,p}$
    $b \leftarrow w_\kappa$
    $c \leftarrow s - \sum_{k \in \kappa} w_k$
    $h_q \leftarrow \max(0, weighted\_median([a \quad 0], [b \quad c]))$
end for

Une méthode *greedy* d'initialisation pour w et h peut être imaginée (voir Algorithme 7). Nous pouvons sélectionner l'index i^* tel que la somme de la ligne $Z_{i^*,:}$ est la plus grande des sommes des lignes $Z_{i,:}$ avec $i = 1, ...m$. Nous pouvons alors initialiser w avec des zéros partout, sauf à l'index i^* où $w_{i^*} = 1$. La solution pour h sera alors $h = Z_{i^*,:}$. Quelques itérations de ℓ_1-NMF de rang 1 permettent alors de trouver un nouveau minimum. Ce minimum convergera inévitablement vers une solution non-nulle. Il est en effet simple de montrer que $w_i = 0$ pour tout $i \neq i^*$, $w_{i^*} = 1$ et $h = Z_{i^*,:}$ sont une meilleure solution que $w = 0^m$ et $h = 0^n$. Dans notre cas, la descente de coordonnées ne peut qu'améliorer la solution initiale, et il est donc impossible que l'Algorithme 7 nous donne des valeurs nulles de w et h.

Algorithme 7 w, h = greedy_rank1_l1_nmf(Z)

INPUT : $Z \in \mathbb{R}^{m \times n}$
OUTPUT : $w \in \mathbb{R}^m_+, h \in \mathbb{R}^n_+$
$w \leftarrow 0^m$
$i^* \leftarrow \text{argmax}_{i=1,...m} \sum_{j=1,...n} Z_{i,j}$
$w_{i^*} = 1$
$h \leftarrow Z_{i^*,:}$
while stop criterion not satisfied **do**
    $w =$ rank1_l1_coordinate_descent(Z', h)
    $h =$ rank1_l1_coordinate_descent(Z, w)
end while

Nous pouvons voir sur le Tableau 3.1 une comparaison entre la ℓ_1-NMF de base (voir Tableau 2.2) et la ℓ_1-NMF en réinstanciant les colonnes de W et les lignes de H nulles. Nous utiliserons les métriques de précision (Acc) et de proportion de colonnes non-nulles de H ($|\vec{e}H|_0$) pour effectuer la comparaison. Seuls les jeux de données ayant une métrique $|\vec{e}W|_0$ inférieure à 100% ont été testées.

Pour tous les jeux de données, la réinstanciation a augmenté la métrique $|\vec{e}W|_0$ à 1, ce qui était l'effet principal souhaité. N'avoir aucune colonne de W remplie de zéros nous permet de voir des améliorations générales sur la proportion de colonnes non-nulles de H, ce qui a inévitablement des conséquences positives sur la précision de l'algorithme. L'amélioration est cependant assez peu marquée pour le jeu de données "ohscal". Ce problème est peut-être dû à une fonction de réinstanciation (dans notre cas, l'Algorithme 7) peu adaptée à ce jeu de données.

		ℓ_1-NMF without reinstance			ℓ_1-NMF with reinstance		
Dataset Name	r	*Acc* [%]	$\lvert\vec{e}H\rvert_0$ [%]	*time* [s]	*Acc* [%]	$\lvert\vec{e}H\rvert_0$ [%]	*time* [s]
NG20	20	9.1	67.53	69.88	11.46	72.36	540.27
classic	4	5.77	5.77	1.98	15.99	28.31	13.51
ohscal	10	22.71	86.19	5.19	23.35	88.8	8.69

TABLE 3.1 – Comparaison de la précision et de la proportion de colonnes non-nulles de H sur les méthodes de ℓ_1-NMF avec et sans réinstanciation sur les jeux de données "NG20", "classic" et "ohscal" provenant de [31].

Cependant, nous voyons que la réinstanciation a un effet très négatif sur le temps de traitement de l'algorithme. Cette augmentation du facteur temps est notamment due au temps de traitement de l'Algorithme 7, mais également au temps qu'il faut pour vérifier si une colonne de W est à réinstancier. Le calcul explicite de la matrice $Z = X - WH$ est également coûteux en complexité et en mémoire, ce qui peut aussi provoquer une hausse du temps de traitement.

Il est donc parfois préférable de ne pas activer la réinstanciation afin d'accélérer le temps de traitement de l'algorithme. Il peut être également intéressant de trouver des méthodes plus rapides et/ou plus efficaces pour la réinstanciation.

Dans la suite de ce chapitre, la réinstanciation sera toujours activée afin d'améliorer les résultats. Nous verrons cependant dans le chapitre suivant une méthode rendant la réinstanciation obsolète pour nos jeux de données.

3.3 Variantes du Modèle de Base de la ℓ_1-NMF

3.3.1 Modifications de l'Ordre des Itérations de la Descente de Coordonnées

La méthode de résolution de la ℓ_1-NMF dans la section 2.2 utilise la descente de coordonnées. Nous pouvons voir dans l'Algorithme 4 que nous itérons toujours dans l'ordre des lignes de H. La descente de coordonnées n'assure cependant pas la convergence vers le minimum global, et l'ordre dans lequel nous minimisons les coordonnées peut avoir un impact sur le minimum atteint. Il semble donc mauvais de choisir un ordre arbitraire dans lequel itérer, et choisir un ordre intelligent ou aléatoire peut être préférable tout en donnant également de bons résultats [26].

Descente Aléatoire de Coordonnées - *Random Coordinate Descent* - R-CD

La descente aléatoire de coordonnées consiste à parcourir les coordonnées dans un ordre aléatoire. Les colonnes de H n'ayant pas d'impact entre elles, l'ordre aléatoire de descente de coordonnées peut se faire uniquement lors des itérations de 1 à r. L'Algorithme 8 permet de traiter les coordonnées dans un ordre aléatoire.

Dans notre cas, au début de chaque modification d'une colonne $H_{:,q}$, nous mélangeons la liste d'index {1, ..., r} pour obtenir notre ordre aléatoire. Cet ordre pour modifier les coordonnées permet de ne pas minimiser deux fois le même index sans avoir au préalable minimiser au moins une fois tous les autres index. L'ordre aléatoire sans remplacement est meilleur dans beaucoup de contextes que d'autres descentes aléatoires de coordonnées possibles [26].

Algorithme 8 $H^{(i)}$ = sparse_l1_random_coordinate_descent($X, W, H^{(i-1)}$)

INPUT : $X \in \mathbb{R}^{m\times n}, W \in \mathbb{R}_+^{m\times r}, H^{(i-1)} \in \mathbb{R}^{r\times n}, \lambda \in [1;0]$
OUTPUT : $H^{(i)} \in \mathbb{R}^{r\times n}$
$H^{(i)} \leftarrow H^{(i-1)}$
$s_p = \sum_{j=1}^{m} W_{j,p} \qquad \forall p = 1,...r$
for $q = 1 : n$ **do**
    $\kappa_{+_q} = \{k \mid X_{k,q} \neq 0\}$
    $v = W_{\kappa_{+_q},:} H^{(i)}_{:,q}$
    $random\_index = shuffle([1:r])$
    for $p \in random\_index$ **do**
        $a \leftarrow X_{\kappa_{+_q},p} - v + W_{\kappa_{+_q},p} H^{(i)}_{p,q}$
        $b \leftarrow W_{\kappa_{+_q},p}$
        $c \leftarrow s_p - \sum_{k\in\kappa_{+_q}} W_{k,p}$
        $H^{(i)}_{p,q} \leftarrow weighted\_median([a \;\; 0],[b \;\; c])$
        $v \leftarrow v + W_{\kappa_{+_q},p}(H^{(i)}_{p,q} - H^{(i-1)}_{p,q})$
    end for
end for

Descente de Coordonnées par Pentes - *Slope Coordinate Descent* - S-CD

Il peut être intéressant de privilégier les index les plus susceptibles de diminuer la fonction objectif en les traitant en premier dans la descente de coordonnées. Il est cependant complexe de savoir dès le début quel index minimisera le mieux la fonction objectif.

Une manière de simplifier le problème serait de regarder la pente la plus négative en $H_{p,q} = 0$ pour chaque p. Nous pouvons supposer que les pentes les plus négatives seront susceptibles d'avoir un meilleur impact sur la diminution de la fonction objectif. Cela n'est cependant pas toujours vrai, mais l'approximation peut suffire pour des cas réels de descente de coordonnées.

La fonction objectif est le plus souvent non-différentiable en $H_{p,q} = 0$, mais nous pouvons calculer la pente en $H_{p,q} = 0^+$, ce qui a déjà été fait pour l'Équation (2.6).

$$\begin{aligned}\lim_{H_{p,q}\to 0^+} \frac{\delta f}{\delta H_{p,q}}(H_{p,q}) &= \sum_{i\in\kappa_{0_q}} W_{i,p} - \sum_{i\in\kappa_{+,q}} W_{i,p} \\ &= \sum_{i=1}^{m} W_{i,p} - 2\sum_{i\in\kappa_{+_q}} W_{i,p}.\end{aligned} \tag{3.1}$$

Les index de la descente de coordonnées sont alors triés par ordre croissant de pentes en 0^+, afin de minimiser d'abord les coordonnées avec les pentes les plus négatives. L'Algorithme 9 calcule alors les pentes pour chaque $p = 1,...r$ et traite les coordonnées dans l'ordre croissant des pentes.

Il est intéressant de noter que, dans le cas d'une matrice binaire, étant donné qu'il n'existe que deux solutions possibles pour $H_{p,q}$, l'indice p de la pente la plus négative en 0^+ est bien l'indice qui minimisera le plus la fonction objectif. Cette propriété peut permettre d'améliorer encore plus la complexité d'un algorithme de ℓ_1-NMF binaire.

Algorithme 9 $H^{(i)}$ = sparse_l1_slope_coordinate_descent($X, W, H^{(i-1)}$)

INPUT : $X \in \mathbb{R}^{m \times n}, W \in \mathbb{R}_+^{m \times r}, H^{(i-1)} \in \mathbb{R}^{r \times n}, \lambda \in [1;0]$
OUTPUT : $H^{(i)} \in \mathbb{R}^{r \times n}$
$H^{(i)} \leftarrow H_0^{(i-1)}$
$s_p = \sum_{j=1}^{m} W_{j,p} \qquad \forall p = 1, ...r$
for $q = 1 : n$ **do**
    $\kappa_{+_q} = \{k \mid X_{k,q} \neq 0\}$
    $v = W_{\kappa_{+_q},:} H^{(i)}_{:,q}$
    for $p = 1 : r$ **do**
        $\Delta_p = 2 \sum_{i \in \kappa_{+_q}} W_{i,p} - s_p$
    end for
    $[\Delta, index] \leftarrow sort(\Delta)$
    for $p \in index$ **do**
        $a \leftarrow X_{\kappa_{+_q},p} - v + W_{\kappa_{+_q},p} H^{(i)}_{p,q}$
        $b \leftarrow W_{\kappa_{+_q},p}$
        $c \leftarrow s_p - \sum_{k \in \kappa_{+_q}} W_{k,p}$
        $H^{(i)}_{p,q} \leftarrow constrained\_weighted\_median([a \quad 0], [b \quad c])$
        $v \leftarrow v + W_{\kappa_{+_q},p}(H^{(i)}_{p,q} - H^{(i-1)}_{p,q})$
    end for
end for

Descente de Coordonnées par Corrélation - *Correlated Coordinate Descent* - C-CD

Une autre approche intelligente dans l'ordre des index serait de les trier par ordre décroissant des corrélations entre les thèmes $W_{:,p}$ et le texte $X_{:,q}$. La corrélation $\rho_{p,q}$ entre un texte q et un thème p est calculée de la manière suivante :

$$\rho_{p,q} = W_{:,p}^T X_{:,q}. \tag{3.2}$$

Nous voyons alors le lien entre la corrélation et la métrique $|\vec{e}W^T X|_0$ présentée dans la Section 2.5.2. $W^T X$ donne en effet une matrice $\mathbb{R}_+^{r \times n}$ telle que $(W^T X)_{p,q} = \rho_{p,q}$. La métrique $|\vec{e}W^T X|_0$ vérifie alors la proportion de textes corrélés à au moins un thème. Il est donc important pour cette métrique de valoir 1.

L'Algorithme 10 calcule pour un texte sa corrélation avec chaque thème et traite les coordonnées dans l'ordre décroissant de corrélation.

Ce modèle de descente de coordonnées devrait être celui qui prendra le plus de temps pour effectuer une itération. Le produit $W^T X_{:,q}$ prend presque autant de temps à calculer que le temps de traitement de la descente de coordonnées. Ce produit a une complexité de $\mathcal{O}(r * m)$ tandis que la boucle d'itération sur $p = 1, ...r$ a une complexité de $\mathcal{O}(r * m_+ \log(m_+))$, mais m_+ est beaucoup plus petit que m.

Algorithme 10 $H^{(i)}$ = sparse_l1_correlated_coordinate_descent($X, W, H^{(i-1)}$)

INPUT : $X \in \mathbb{R}^{m \times n}, W \in \mathbb{R}_+^{m \times r}, H^{(i-1)} \in \mathbb{R}^{r \times n}, \lambda \in [1;0]$
OUTPUT : $H^{(i)} \in \mathbb{R}^{r \times n}$
$H^{(i)} \leftarrow H^{(i-1)}$
$s_p = \sum_{j=1}^{m} W_{j,p} \qquad \forall p = 1, ...r$
for $q = 1 : n$ **do**
    $\kappa_{+_q} = \{k \mid X_{k,q} \neq 0\}$
    $v = W_{\kappa_{+_q},:} H^{(i)}_{:,q}$
    $\rho = W^T X_{:,q}$
    $[\rho, index] \leftarrow sort(\rho)$
    for $p \in index$ **do**
        $a \leftarrow X_{\kappa_{+_q},p} - v + W_{\kappa_{+_q},p} H^{(i)}_{p,q}$
        $b \leftarrow W_{\kappa_{+_q},p}$
        $c \leftarrow s_p - \sum_{k \in \kappa_{+_q}} W_{k,p}$
        $H^{(i)}_{p,q} \leftarrow weighted\_median([a \;\; 0], [b \;\; c])$
        $v \leftarrow v + W_{\kappa_{+_q},p}(H^{(i)}_{p,q} - H^{(i-1)}_{p,q})$
    end for
end for

3.3.2 Modèle Accéléré par Itérations Internes Multiples

Dans le modèle de descente de coordonnées de base CD (voir Algorithme 4), le calcul de $v = W_{\kappa,:} H_{:,q}$ pour chaque texte $q = 1, ...n$ augmente un peu le temps de traitement. Il peut être intéressant d'effectuer plusieurs fois la minimisation de $H_{:,q}$ après avoir calculé v, afin de gagner en temps de traitement.

Une méthode similaire existe déjà pour accélérer les modèles de ℓ_2-NMF MU et HALS [9]. Ces méthodes fonctionnent cependant sur des méthodes heuristiques en norme ℓ_2, prenant souvent beaucoup d'itérations pour converger. La norme ℓ_1 converge cependant beaucoup plus rapidement, généralement en moins de 10 itérations dans notre cas. La variante accélérée pourrait alors avoir un impact minime sur nos résultats.

L'Algorithme 11 permet de mettre en place cette variante accélérée pour le modèle de descente de coordonnées de base CD (voir Algorithme 4). De la même manière, cette variante d'accélération peut être mise en place pour les modèles R-CD, S-CD et C-CD. Les algorithmes accélérés des autres modèles de descente de coordonnées sont explicités dans l'Annexe B.

Le critère d'arrêt pour les itérations internes est un point important de cette variante accélérée. Dans notre cas, nous limiterons le critère d'arrêt à un seuil sur la différence entre $H_{:,q}$ et sa valeur avant l'itération interne. Si la différence est trop petite, alors les itérations internes s'arrêteront. Nous ajouterons également un nombre maximum d'itérations interne afin d'éviter d'attendre trop longtemps la convergence de l'algorithme.

Des critères d'arrêt plus complexes pourraient être mis en place [9], mais nous nous concentrerons d'abord sur les résultats de cette méthode avant de réfléchir à l'améliorer.

Algorithme 11 $H^{(i)}$ = accelerated_sparse_l1_coordinate_descent($X, W, H^{(i-1)}$)

INPUT : $X \in \mathbb{R}^{m \times n}, W \in \mathbb{R}_+^{m \times r}, H^{(i-1)} \in \mathbb{R}^{r \times n}, \lambda \in [1;0]$
OUTPUT : $H^{(i)} \in \mathbb{R}^{r \times n}$
$H^{(i)} \leftarrow H^{(i-1)}$
$s_p = \sum_{j=1}^{m} W_{j,p} \qquad \forall p = 1, ...r$
for $q = 1 : n$ **do**
    $\kappa_{+_q} = \{k \mid X_{k,q} \neq 0\}$
    $v = W_{\kappa_{+_q},:} H^{(i)}_{:,q}$
    while stop criterion not satisfied **do**
        $h_{old} \leftarrow H^{(i)}_{:,q}$
        for $p = 1, ...n$ **do**
            $a \leftarrow X_{\kappa_{+_q},p} - v + W_{\kappa_{+_q},p} H^{(i)}_{p,q}$
            $b \leftarrow W_{\kappa_{+_q},p}$
            $c \leftarrow s_p - \sum_{k \in \kappa_{+_q}} W_{k,p}$
            $H^{(i)}_{p,q} \leftarrow constrained\_weighted\_median([a \quad 0], [b \quad c])$
            $v \leftarrow v + W_{\kappa_{+_q},p} (H^{(i)}_{p,q} - h_{old_p})$
        end for
    end while
end for

3.3.3 Résultats et Comparaisons des modèles

Le modèle de base, les trois modèles modifiants l'ordre des itérations et leur variante accélérée composent les huit premiers modèles de la ℓ_1-NMF que nous allons comparer. Ces modèles seront testés sur les jeux de données provenant de [31]. Nous comparerons les modèles par le temps de traitement (t), l'erreur en norme ℓ_1 ($err = \frac{||X - WH||_1}{||X||_1}$) et la précision ($Acc$). Les résultats sont présents dans le tableau 3.2. L'itération initiale W_0 et H_0 est la même pour tous les modèles.

Afin d'avoir une vue globale sur les résultats, une dernière ligne montre la moyenne des métriques normalisées. Cette normalisation se fait en divisant les métriques d'une même catégorie d'un même jeu de données par la métrique la plus élevée de cette catégorie pour ce jeu de données. Cette moyenne permet une meilleure comparaison des modèles entre eux.

Pour un même jeu de données et pour les moyennes, le meilleur temps est mis en gras et le deuxième meilleur temps est souligné. Les variations sur l'erreur et sur la précision ont été jugées trop faibles pour qu'il soit pertinent de mettre en évidence les meilleures erreurs et précisions.

Une première conclusion à faire sur le Tableau 3.2 porte sur la faible variation en fonction des modèles des erreurs en norme ℓ_1 (err) et des précisions (Acc). En regardant les moyennes normalisées, nous pouvons voir que la plus grande différence est de 0.6% pour les erreurs et de 3.8% pour les précisions. Aucun modèle ne semble être significativement plus performant qu'un autre sur base de ces métriques.

Une autre conclusion rapide à tirer porte sur la descente de coordonnées par corrélation C-CD et sa variante accélérée. Nous voyons que pour un même jeu de données, le temps de traitement est souvent significativement plus long pour les modèles C-CD et acc-C-CD que pour les autres modèles.

		CD			acc-CD			R-CD			acc-R-CD		
Dataset	r	t [s]	err	Acc [%]	t [s]	err	Acc [%]	t [s]	err	Acc [%]	t [s]	err	Acc [%]
NG20	20	**540**	1.43e-2	11.46	573	1.43e-2	10.13	738	1.43e-2	12.86	572	1.43e-2	10.21
ng3sim	3	6.89	5.63e-2	17.38	11.63	5.63e-2	17.38	9.65	5.67e-2	17.21	**4.11**	5.67e-2	17.04
classic	4	13.51	6.85e-3	15.99	14.02	6.85e-3	15.99	13.6	6.85e-3	15.99	14.02	6.85e-3	15.99
ohscal	10	**8.69**	3.30e-2	23.35	10.9	3.30e-2	23.36	9.47	3.30e-2	21.94	10.86	3.30e-2	23.0
k1b	6	**7.69**	4.99e-2	68.42	8.9	4.99e-2	68.93	7.77	4.99e-2	67.69	9.04	4.99e-2	68.59
hitech	6	**4.54**	1.02e-1	41.81	5.46	1.02e-1	39.16	4.68	1.02e-1	40.94	5.61	1.02e-1	37.9
reviews	5	7.04	6.99e-2	58.76	8.78	6.99e-2	58.76	7.13	6.99e-2	59.3	8.93	6.99e-2	58.69
sports	7	22.39	4.69e-2	32.26	28.74	4.69e-2	33.53	24.76	4.69e-2	33.55	29.18	4.69e-2	33.43
tr11	9	17.54	4.32e-1	5.07	13.24	4.30e-1	4.59	**7.08**	4.40e-1	6.52	14.82	4.30e-1	4.59
tr23	6	**4.14**	1.11	9.31	14.3	1.09	9.31	16.8	1.11	4.9	9.46	1.10	8.82
tr41	10	**6.93**	2.13e-1	29.04	14.75	2.11e-1	31.78	10.26	2.10e-1	33.6	7.3	2.09e-1	34.28
tr45	10	17.23	3.28e-1	5.07	10.75	3.25e-1	5.22	**10.44**	3.30e-1	5.07	14.06	3.25e-1	5.22
Normalized Mean		0,263	0,996	0,939	0,336	0,993	0,933	0,281	0,998	0,934	0,278	0,993	0,927
		S-CD			**acc-S-CD**			**C-CD**			**acc-C-CD**		
NG20	20	805	1.43e-2	12.38	740	1.43e-2	10.44	3930	1.43e-2	10.77	5208	1.43e-2	10.23
ng3sim	3	9.13	5.68e-2	17.11	12.83	5.67e-2	17.38	20.32	5.68e-2	17.08	49.91	5.63e-2	17.38
classic	4	13.61	6.85e-3	15.99	14.4	6.85e-3	15.99	64.73	6.85e-3	15.99	65.97	6.85e-3	15.99
ohscal	10	8.91	3.30e-2	22.45	11.44	3.30e-2	23.29	102	3.30e-2	23.09	109	3.30e-2	23.08
k1b	6	8.19	4.99e-2	68.25	9.42	4.99e-2	68.93	41.04	4.99e-2	67.69	43.3	4.99e-2	68.5
hitech	6	5.45	1.02e-1	40.5	7.17	1.02e-1	41.76	19.37	1.02e-1	47.5	24.29	1.02e-1	37.59
reviews	5	**5.95**	6.99e-2	59.65	9.21	6.99e-2	58.69	49.92	6.99e-2	58.96	44.77	6.99e-2	58.76
sports	7	**20.54**	4.69e-2	33.58	29.68	4.69e-2	33.46	173	4.69e-2	33.68	152	4.69e-2	33.47
tr11	9	11.57	4.37e-1	4.83	12.95	4.30e-1	4.59	10.95	4.37e-1	4.83	19.63	4.30e-1	4.59
tr23	6	5.12	1.09	5.88	17.59	1.08	8.82	10.54	1.05	3.92	22.0	1.08	8.82
tr41	10	7.95	2.09e-1	33.94	8.16	2.08e-1	34.17	12.69	2.14e-1	35.54	13.72	2.09e-1	33.94
tr45	10	15.43	3.30e-1	5.07	11.67	3.25e-1	5.22	38.44	3.24e-1	5.22	15.94	3.25e-1	5.07
Normalized Mean		**0,253**	0,995	0,921	0,326	0,992	0,940	0,810	0,993	0,912	0,927	0,992	0,927

TABLE 3.2 – Comparaison du temps de traitement, de la précision et de l'erreur en norme ℓ_1 des modèles présentés dans les Sections 3.3.1 et 3.3.2 sur les jeux de données provenant de [31].

Nous pouvons également voir que les variantes accélérées ne semblent pas être plus rapides que les variantes originales. Elles sont même régulièrement plus lentes. Ce phénomène s'explique sans doute par le fait que la fonction de mise à jour en ℓ_1-NMF (voir Algorithme 4) a tendance à vite converger vers un minimum du bloc, parfois en une itération. Les variantes accélérées causent alors souvent une itération supplémentaire, et donc une hausse du temps de traitement total de l'algorithme.

Les autres différences de temps n'ont pas l'air significatives. Nous pouvons cependant voir que le modèle de descente de coordonnées par pentes S-CD semble légèrement plus rapide que les autres modèles.

Pour la suite des tests, nous exclurons dans un premier temps les modèles C-CD et acc-C-CD dû à leur mauvaise performance en temps. Nous garderons cependant les autres modèles, les différences en temps, erreurs et précisions n'étant pas jugées suffisantes pour exclure d'autres modèles.

3.4 Conclusions

Dans ce chapitre, nous avons dans un premier temps testé une méthode de réinstanciation de colonnes de W. La réinstanciation nous a poussé à mettre en place une nouvelle méthode d'initialisation : la *greedy*-ℓ_1-NMF (voir Algorithme 7).

La réinstanciation améliore généralement bien les résultats, et résout le problème de colonnes nulles de W, caractérisé par la métrique $|\vec{e}W|_0$ inférieure à 1. Elle a pour effet positif d'augmenter la proportion de colonnes non-nulles de H, et par extension la précision des résultats. Cependant, la réinstanciation possède un coût en temps assez élevé, allant jusqu'à multiplier par dix le temps de traitement sur certains jeux de données.

Nous avons ensuite mis en place des variantes de descente de coordonnées pour l'algorithme initial de la ℓ_1-NMF (Algorithme 4). Cependant, ces variantes ne semblent pas avoir d'impact significatif sur les résultats. Une variante d'accélération a également été mis en place, mais elle ne semble pas modifier significativement le temps de traitement de l'algorithme.

Certains problèmes de la ℓ_1-NMF n'ont pas encore été résolus par les modèles déjà présentés, notamment la trop grande proportion de colonnes nulles de H. Le prochain chapitre propose une solution à ce problème, en plus de proposer une alternative à la réinstanciation, coûteuse en temps.

Chapitre 4

Modèles de ℓ_1-NMF Dépénalisant les Zéros

4.1 Motivations

Nous avons vu dans la Section 2.5.3 que les résultats de la ℓ_1-NMF étaient souvent encombrés de zéros. Cela provoque une grande quantité de données non-classifiées, caractérisées par la métrique $|\vec{e}H|_0$ inférieure à 1. Des données non-classifiées ne sont pas souhaitables, surtout pour notre méthode de comparaison se basant principalement sur la précision des données classifiées.

De plus, dans la plupart des cas pratiques, les données collectées sont fréquemment encombrés de zéros, qu'ils soient faux ou vrais, de manière indiscernable [17, 28]. Il pourrait alors être intéressant d'imaginer un modèle pénalisant moins les zéros, de sorte que le modèle continue de prédire des résultats creux sans pour autant ne prédire que des zéros.

Le modèle imaginé dans le cadre de ce travail vise à diminuer la pénalité des zéros dans la fonction objectif. En reprenant la fonction objectif de la descente de coordonnées d'Équation (2.1), nous pouvons y ajouter un paramètre λ dépénalisant les zéros :

$$\min_{W,H\geq 0} f(W,H,\lambda) = \sum_{i,j\in\kappa_+} |X-WH|_{i,j} + \lambda \sum_{i,j\in\kappa_0} |WH|_{i,j}, \tag{4.1}$$

avec λ un réel compris entre 0 et 1 inclus, $\kappa_+ = \{(i,j)|X_{i,j} > 0\}$ et $\kappa_0 = \{(i,j)|X_{i,j} = 0\}$ deux sous-ensembles complémentaires à l'ensemble des combinaisons d'indices (i,j).

Si le paramètre λ a une valeur de 1, la fonction objectif correspond à celle du modèle initial. Si le paramètre λ a une valeur de 0, la fonction objectif dépénalise complètement les zéros de la matrice initiale X et le modèle essayera de prédire uniquement les valeurs de X supérieures à 0.

Le paramètre λ peut permettre de prédire moins de zéros et d'avoir des meilleurs résultats finaux. Cependant, rien ne permet de déterminer une valeur optimale à ce paramètre. Nous pouvons dans un premier temps supposer que la meilleure valeur de λ est propre au jeu de données et aux résultats souhaités.

Il est à noter que le paramètre λ ne retire pas la symétrie entre W et H. Nous pourrons ainsi n'utiliser qu'une seule fonction de mise à jour de H pour notre modèle :

$$\sum_{i,j\in\kappa_+} |X-WH|_{i,j} + \lambda \sum_{i,j\in\kappa_0} |WH|_{i,j} = \sum_{i,j\in\kappa_+} |X^T - H^T W^T|_{j,i} + \lambda \sum_{i,j\in\kappa_0} |H^T W^T|_{j,i}. \tag{4.2}$$

4.2 Modèle Dépénalisant – λ-ℓ_1-NMF

4.2.1 Résolution du Modèle λ-ℓ_1-NMF

L'ajout du paramètre λ ne modifie pas beaucoup l'implémentation du modèle. La fonction de descente de coordonnées (voir Équation (2.2)) avec le paramètre λ devient :

$$\begin{aligned}\min_{H_{p,q}\geq 0} f(H_{p,q}) &= \sum_{i\in\kappa_{+_q}} |X-WH|_{i,q} + \lambda \sum_{i\in\kappa_{0_q}} |WH|_{i,q} \\ &= \sum_{\substack{i \\ X_{i,q}>0}} |X_{i,q} - \sum_{\substack{k=1 \\ k\neq p}}^{r} (W_{i,k}H_{k,p}) - W_{i,p}H_{p,q}| + \lambda H_{p,q} \sum_{\substack{i \\ X_{i,q}=0}} W_{i,p} \\ &= \sum_{\substack{i \\ X_{i,q}>0}} |a_i - b_i H_{p,q}| + |0 - \lambda c H_{p,q}|,\end{aligned} \tag{4.3}$$

avec :

- $\lambda = [0;1]$ le paramètre du modèle ;
- $\kappa_{+_q} = \{i|(i,q)\in\kappa_+\} = \{i|X_{i,j}>0, j=q\}$ un sous-ensemble de κ_+ avec l'indice q fixé ;
- $\kappa_{0_q} = \{i|(i,q)\in\kappa_0\} = \{i|X_{i,j}=0, j=q\}$ un sous-ensemble de κ_0 avec l'indice q fixé ;
- $a_i = X_{i,q} - \sum_{\substack{k=1 \\ k\neq p}}^{r} (W_{i,k}H_{k,p})$; $b_i = W_{i,p}$ et $c = \sum_{i\in\kappa_{0_q}} W_{i,p}$.

La seule différence avec le modèle initial est alors l'ajout du facteur λ à la variable c. Nous pouvons alors facilement modifier l'Algorithme 4 pour en faire l'Algorithme 12 qui prendra en compte notre paramètre λ.

Algorithme 12 $H^{(i)}$ = depenalizing_l1_coordinate_descent($X, W, H^{(i-1)}, \lambda$)

INPUT : $X \in \mathbb{R}^{m\times n}, W \in \mathbb{R}_+^{m\times r}, H^{(i-1)} \in \mathbb{R}^{r\times n}, \lambda \in [1;0]$
OUTPUT : $H \in \mathbb{R}^{r\times n}$
$H^{(i)} \leftarrow H^{(i-1)}$
$s_p = \sum_{j=1}^{m} W_{j,p} \qquad \forall p = 1,...r$
for $q = 1 : n$ **do**
    $\kappa_{+_q} = \{k \mid X_{k,q} \neq 0\}$
    $v = W_{\kappa_{+_q},:} H^{(i)}_{:,q}$
    for $p = 1 : r$ **do**
        $a \leftarrow X_{\kappa_{+_q},p} - v + W_{\kappa_{+_q},p} H^{(i)}_{p,q}$
        $b \leftarrow W_{\kappa_{+_q},p}$
        $c \leftarrow \lambda * (s_p - \sum_{k\in\kappa_{+_q}} W_{k,p})$
        $H^{(i)}_{p,q} \leftarrow weighted\_median([a \quad 0], [b \quad c])$
        $v \leftarrow v + W_{\kappa_{+_q},p}(H^{(i)}_{p,q} - H^{(i)}_{p,q})$
    end for
end for

Il est à noter que le modèle λ-ℓ_1-NMF présenté dans l'Algorithme 12 est compatible avec les variantes de descentes de coordonnées présentées dans la Section 3.3.1. Seul le modèle S-CD subit une légère modification dans le calcul de la pente en 0^+, qui devient alors :

$$\lim_{H_{p,q}\to 0^+} \frac{\delta f}{\delta H_{p,q}}(H_{p,q}) = \lambda * \sum_{i=1}^{m} W_{i,p} - (1+\lambda) \sum_{i\in\kappa_{+_q}} W_{i,p}. \tag{4.4}$$

4.2.2 Modèle λ-ℓ_1-NMF pour des Matrices Creuse

Notre travail porte sur la factorisation de matrices creuses. Comme dit précédemment, il est plus souhaitable de produire un résultat creux dans notre cas. Le modèle λ-ℓ_1-NMF dépénalise la prédiction de zéros. Il semble alors logique qu'il produira des résultats avec moins de zéros et donc moins creux. Nous pouvons nous intéresser à la condition pour prédire des zéros de ce nouveau modèle.

En reprenant l'équation 2.6, nous pouvons facilement y ajouter le paramètre λ en facteur de la somme sur l'ensemble κ_{0_q}. Nous obtenons alors la condition suivante pour prédire un $H_{p,q}$ nul :

$$\lambda * \sum_{i \in \kappa_{0_q}} W_{i,p} \geq \sum_{i \in \kappa_{+_q}} W_{i,p}. \tag{4.5}$$

Pour un λ petit, l'Équation (4.5) montre qu'il y a beaucoup moins de chance de produire de zéro que pour un λ grand. Le résultat WH sera alors moins creux au plus λ est petit.

Dans le cas limite $\lambda = 0$, la condition pour que $H_{p,q}$ soit nul devienne $0 \geq \sum_{i \in \kappa_{+_q}} W_{i,p}$, ce qui, pour un cas réel, n'est possible que si $W_{i,p} = 0$ pour tous les indices i. Avec la réinstanciation activée, pour $\lambda = 0$, ça montre que pour tous les indices i et j, $(WH)_{i,j} > 0$. Le résultat n'est donc pas du tout creux.

Dans notre étude des résultats, nous nous concentrerons uniquement sur les valeurs de λ supérieures à 0, ce qui permettra de garder des résultats un minimum creux.

4.2.3 Résultats du Modèle λ-ℓ_1-NMF

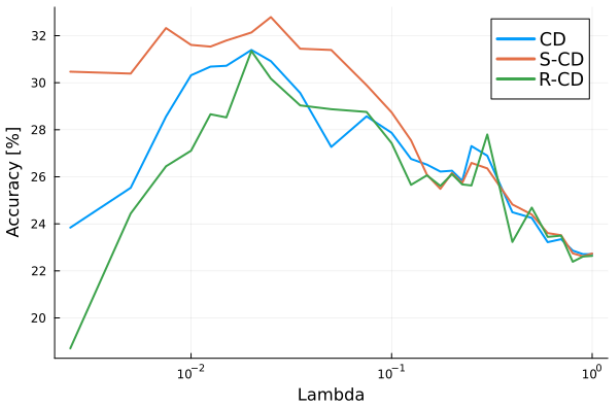
Afin de gagner du temps pour tester le modèle, nous ne comparerons que 4 jeux de données entre eux : *ng3sim*, *ohscal*, *sports* et *tr45*. Ces jeux de données ne sont pas trop grands, et assez diversifiés dans leurs caractéristiques et leurs résultats obtenus précédemment. Ils semblent alors être un bon choix comme échantillon représentatif de nos jeux de données. Nous testerons les modèles CD, R-CD et S-CD pour la descente de coordonnées. Beaucoup de valeurs de λ seront testées afin d'essayer de trouver un optimum.

Nous comparerons d'abord les méthodes entre elles. Nous comparerons leur précision (Acc) et leur temps de traitement (t) en fonction du paramètre λ. Nous récupérerons ensuite la méthode ayant le mieux performé en précision et/ou en temps, pour comparer les métriques de la méthode en fonction du paramètre λ. Nous nous concentrerons sur les métriques suivantes :

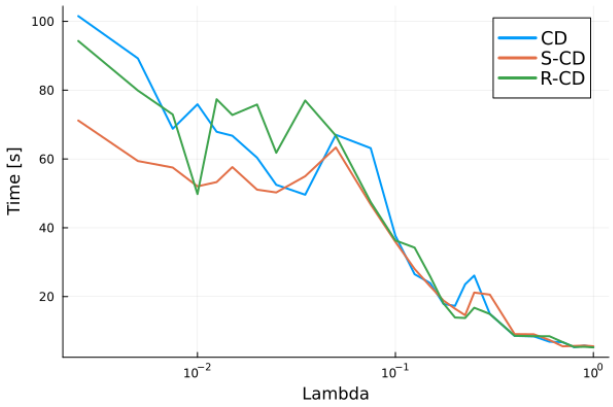
- La précision (Acc) et Précision sans colonnes nulles de H (Acc_0) (voir Section 2.5.2 pour plus de détails).
- Le temps de traitement (t) jusqu'à ce que le modèle ait convergé (ou effectué 100 itérations).
- La proportion de colonnes non-nulles de H ($|\vec{e}H|_0$) (voir Section 2.5.2 pour plus de détails).
- La *sparsity* de WH (voir Équation (2.16)).

Dans cette section, nous ne regarderons que les résultats graphiques du jeu de données *ohscal* provenant de [31], ceux-ci sont disponibles aux Figures 4.1 et 4.2. Les résultats graphiques des autres jeux de données sont disponibles à l'Annexe C.

En testant les différents modèles de descente de coordonnées, une des premières choses que nous avons remarquées porte sur les variantes accélérées. Celles-ci semblent être systématiquement plus lentes que leur variante de base. Nous avons donc décidé d'abandonner temporairement les variantes accélérées, en supposant que celles-ci ne sont pas adaptées à nos modèles ou à nos jeux de données.

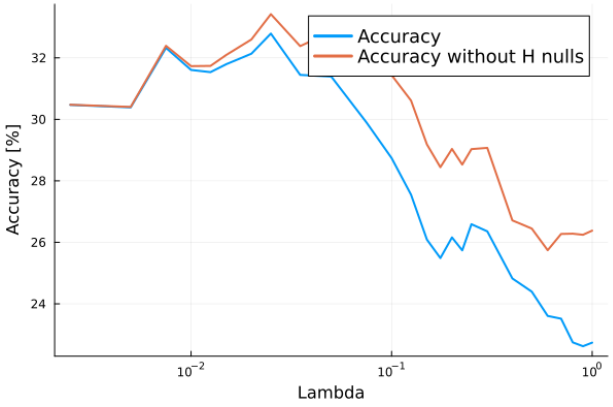


(a) Comparaison de la précision des différents modèles de descente de coordonnées de la λ-ℓ_1-NMF en fonction du paramètre λ.

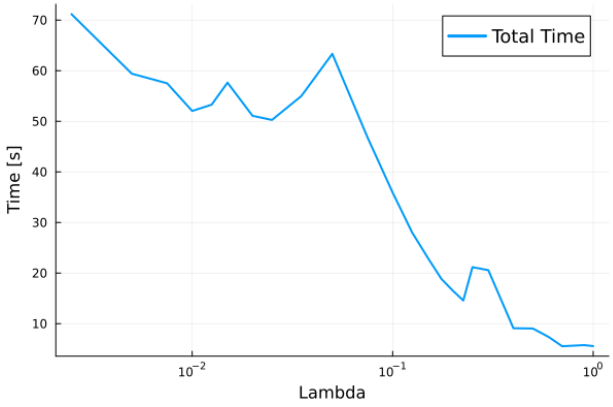


(b) Comparaison du temps de traitement jusqu'à convergence des différents modèles de descente de coordonnées de la λ-ℓ_1-NMF en fonction du paramètre λ.

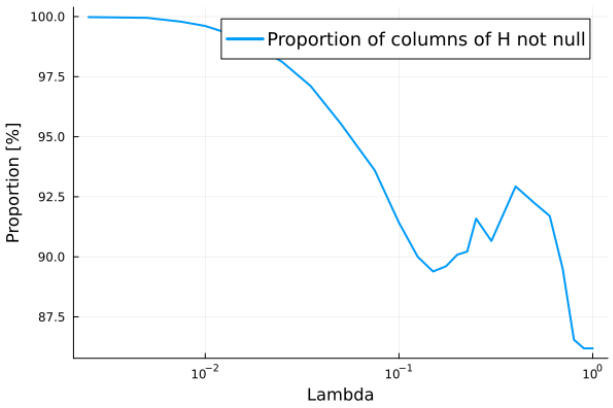
FIGURE 4.1 – Comparaison des différents modèles de descente de coordonnées sur le modèle λ-ℓ_1-NMF sur le Jeu de Données *ohscal* [31].



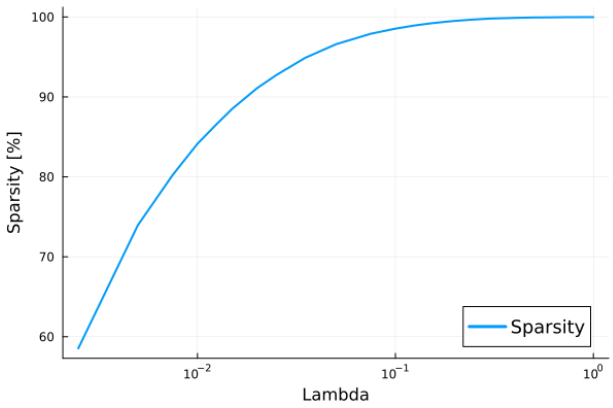
(a) Comparaison de la précision et de la précision sans colonnes nulles de H en fonction du paramètre λ.



(b) Comparaison du temps de traitement jusqu'à convergence en fonction du paramètre λ.



(c) Comparaison de la proportion de colonnes non-nulles de H en fonction du paramètre λ.



(d) Comparaison de la *sparsity* de WH en fonction du paramètre λ.

FIGURE 4.2 – Comparaison de la précision, du temps de traitement, de la proportion de colonnes non-nulles de H et de la "*sparsity*" en descente de coordonnées par pentes (S-CD) sur le modèle λ-ℓ_1-NMF pour le Jeu de Données *ohscal* [31].

En regardant les graphiques de comparaison des modèles de descentes de coordonnées pour le jeu de données *ohscal* (voir Figure 4.1), nous pouvons voir que la précision varie beaucoup en fonction du paramètre λ. Celle-ci semble croître jusqu'à un certain seuil et, à partir d'une certaine valeur seuil de λ (pour *ohscal*, le seuil est environ à 0.02), la précision semble décroître. Cette valeur seuil λ^* est cependant difficile à prédire sans tester toutes les valeurs possibles, et elle semble varier en fonction des jeux de données analysés (voir Annexe C).

Nous pouvons également voir que le temps de traitement du modèle augmente lorsque λ diminue. Ce phénomène est sûrement dû à deux choses :

— L'Algorithme 3 est plus rapide lorsque la solution α est nulle. Diminuer λ diminue le nombre de zéros prédits et donc augmente le temps de traitement.

— Le nombre d'itérations nécessaire pour converger augmente au plus λ diminue, ce qui augmente inévitablement le temps de traitement.

Le modèle S-CD semble assez fiable en termes de précision et performant en temps. Nous regarderons les résultats de ce modèle dans la suite du travail.

Nous pouvons voir que la proportion de colonnes non-nulles de H augmente au plus λ diminue et la "*sparsity*" diminue au plus λ diminue. Ces deux phénomènes sont liés et étaient attendus (voir Section 4.2.2). La fonction objectif dépénalise les zéros prédits, il est donc normal de voir apparaître moins de zéros dans les résultats.

La courbe de précision augmente également lorsque λ diminue (pour λ supérieur à λ^*). Cela pourrait être interprété par une plus grande proportion de colonnes classifiées de H, cependant nous voyons que la courbe de précision sans les colonnes nulles de H augmente également. Les courbes de Acc et Acc_0 se rejoignent peu après avoir dépassé la valeur de λ^*.

Nous voyons que les résultats sont meilleurs que ceux obtenus dans la Section 2.5.3, signifiant que le modèle λ-ℓ_1-NMF semble meilleur que le modèle de base ℓ_1-NMF. Cette amélioration vient notamment du fait que de la proportion de colonnes non-nulles de H soit beaucoup plus élevée.

Ce modèle de λ-ℓ_1-NMF est intéressant, mais a un problème : la valeur seuil λ^* permettant d'obtenir une précision optimale est inconnue. Cette valeur dépend certainement du jeu de données, mais beaucoup de caractéristiques pourraient être liées à la détermination de λ^*.

4.2.4 Choix du paramètre λ

λ fonction de la matrice X

Pour résoudre le problème du choix du paramètre, nous pouvons essayer d'établir une corrélation entre la valeur optimale de λ et S_X, la "*sparsity*" de la matrice X (voir Équation (2.9)). Nous possédons peu d'informations et de jeux de données différents permettant d'établir une fonction empirique correcte $\lambda = f(S_X)$. Ce problème pourrait composer un travail part entière.

Nous savons cependant que la valeur optimale de λ a un ordre de grandeur assez faible. Nous pouvons essayer la fonction d'approximation $\tilde{\lambda} = \sqrt{1-S_X}/2$. Celle-ci n'a aucune base fondée, mais l'ordre de grandeur de $\tilde{\lambda}$ approche celui des λ^* pour lesquels les meilleurs résultats ont été trouvés. Nous pouvons essayer cette valeur de λ pour tous les jeux de données et analyser les résultats. Ceux-ci sont présents sur le Tableau 4.1.

Nous voyons que, pour une valeur de $\lambda = \sqrt{1-S_X}/2$, les résultats sont assez corrects pour la plupart des jeux de données. Beaucoup de jeux de données atteignent une proportion de colonnes non-nulles de H supérieure à 90% et une précision assez haute, dépassant parfois la ℓ_2-NMF.

		λ-ℓ_1-NMF						ℓ_2-NMF
Dataset	r	λ	Acc [%]	$\|\vec{e}H\|_0$ [%]	Acc_0 [%]	S_{WH} [%]	$Time$ [s]	Acc [%]
NG20	20	0.0212	16.34	84.02	19.45	95.444	971.47	23.08
ng3sim	3	0.0420	23.08	59.67	38.68	93.883	14.68	36.06
classic	4	0.0137	41.51	53.33	77.85	98.714	11.05	55.64
ohscal	10	0.0362	31.66	96.98	32.65	95.125	59.3	30.50
k1b	6	0.0385	67.48	91.37	73.85	95.276	14.24	59.19
hitech	6	0.0597	39.29	88.96	44.16	90.786	22.71	46.94
reviews	5	0.0502	56.03	81.54	68.72	93.424	60.83	51.19
sports	7	0.0462	42.67	92.73	46.02	94.057	69.69	40.37
tr11	9	0.1037	40.1	92.27	43.46	41.807	4.2	50.48
tr23	6	0.1283	23.04	67.16	34.31	48.107	12.68	35.29
tr41	10	0.0807	32.23	96.81	33.29	72.252	16.15	44.99
tr45	10	0.0921	32.17	91.3	35.24	44.732	67.6	38.26

TABLE 4.1 – Comparaison des métriques présentées dans la section 2.5.2 de la λ-ℓ_1-NMF pour $\lambda = \sqrt{1-S_X}/2$ et de la précision de la ℓ_2-NMF [10] sur les jeux de données provenant de [31]

Cependant, nous constatons également que beaucoup de jeux de données gardent une proportion de colonnes nulles de H élevée. Ce problème pourrait sûrement être résolu en choisissant un λ plus petit. Il faudrait alors une fonction permettant de déduire un λ plus adapté à l'ensemble des jeux de données, mais rien ne nous permet actuellement de déduire une telle fonction.

Test de plusieurs λ jusqu'à respect d'une condition

Nous pouvons essayer une autre méthode pour choisir un meilleur λ, qui serait adaptée pour tous les jeux de données. La méthode testera les résultats pour une valeur de λ élevée, et si les résultats ne respectent pas un critère d'arrêt, alors nous baisserons la valeur de λ jusqu'à respecter ce critère. Un exemple d'algorithme possible est proposé à l'Algorithme 13.

Algorithme 13 $[W,H,\lambda]$ = l1_NMF_decrising_lambda(X,r,β,λ_0)

INPUT : $X \in \mathbb{R}^{m\times n}, r \in \mathbb{N}, \beta \in]1;0[, \lambda_0 \in [1;0[$
OUTPUT : $W \in \mathbb{R}_+^{m\times r}, H \in \mathbb{R}^{r\times n}, \lambda \in [1;0[$
$[W_0,H_0] \leftarrow initializer(X,r)$
$\lambda \leftarrow \lambda_0$
while stop criterion not satisfied **do**
    $W,H \leftarrow \lambda\_\ell_1\_\text{NMF}\_block\_descent(X,W_0,H_0,\lambda)$
    $\lambda \leftarrow \lambda * \beta$
end while

Cette méthode peut cependant s'avérer très coûteuse en temps si le critère d'arrêt n'est pas adapté, si le jeu de données a besoin d'une faible valeur de λ pour respecter le critère d'arrêt ou si le facteur de pas β choisi est trop grand.

Dans notre cas, nous choisirons le critère d'arrêt suivant : $|\vec{e}H|_0 > 0.9$. D'autres critères auraient pu être choisis, par exemple $|\vec{e}W^TX|_0 = 1$. Nous prendrons un λ initial de 0.5 et un pas β de $\frac{2}{3}$. Les résultats obtenus avec ces paramètres sont visibles sur le Tableau 4.2.

		dec-λ-ℓ_1-NMF						ℓ_2-NMF
Dataset	r	λ	Acc [%]	$\|\vec{e}H\|_0$ [%]	Acc_0 [%]	S_{WH} [%]	$Time$ [s]	Acc [%]
NG20	20	0.014	19.08	91.53	20.84	91.714	1596.99	23.08
ng3sim	3	0.0084	35.02	92.03	38.06	72.257	149.75	36.06
classic	4	0.0011	51.13	91.51	55.87	83.772	115.96	55.64
ohscal	10	0.5	24.4	92.24	26.45	99.936	8.93	30.50
k1b	6	0.5	66.41	90.77	73.16	99.874	6.21	59.19
hitech	6	0.039	39.03	92.05	42.4	86.316	127.28	46.94
reviews	5	0.023	60.7	90.83	66.83	86.508	189.86	51.19
sports	7	0.039	44.3	94.98	46.64	93.143	295.73	40.37
tr11	9	0.108	39.13	95.17	41.12	43.722	47.89	50.48
tr23	6	0.039	36.27	97.55	37.19	13.561	44.25	35.29
tr41	10	0.108	38.27	95.44	40.1	73.245	104.9	44.99
tr45	10	0.065	34.78	98.12	35.45	34.219	162.83	38.26

TABLE 4.2 – Comparaison des métriques présentées dans la Section 2.5.2 de la λ-ℓ_1-NMF pour λ obtenu à l'aide de l'Algorithme 13 et de la précision de la ℓ_2-NMF [10] sur les jeux de données provenant de [31]

Nous voyons que la méthode choisit généralement des valeurs de λ assez faibles pour respecter notre critère d'arrêt. Cette méthode pour choisir le paramètre λ est assez utile, car elle permet de résoudre la plupart des problèmes que nous pourrions rencontrer dans nos résultats. Ici, plus de 90% de nos textes ont été classifiés, et nous voyons que la précision est donc également améliorée.

La métrique de précision est d'ailleurs très proche des résultats de la ℓ_2-NMF. Nous obtenons même de meilleurs résultats pour 4 jeux de données (*k1b*, *reviews* , *sports* et *tr23*) alors même que 100% des textes ne sont pas classifiés.

Nous voyons cependant que la "*sparsity*" de la matrice WH (S_{WH}) est parfois assez faible. Elle est souvent très différente de S_X. Nous voyons que même si nous améliorons nos résultats, nous sacrifions parfois notre volonté d'obtenir des matrices creuses pour approximer une matrice X creuse.

4.3 Modèle Dépénalisant par Colonnes – λ_q-ℓ_1-NMF

Le modèle λ-ℓ_1-NMF part du postulat que les données sont souvent encombrées de zéros, vrais ou faux. Dans notre cas, ces données sont des textes décrits par le nombre d'occurrences de chaque mot. Cependant, un texte de 3000 mots devrait avoir moins de faux-zéros qu'un texte de 150 mots. Le premier texte aura eu le temps de traiter son thème en détail, tandis que le second n'est qu'un résumé, et ne traitera que quelques mots du thème.

Le modèle λ-ℓ_1-NMF dépénalise les zéros de la même manière sur l'ensemble du jeu de données. Il pourrait sembler pertinent d'utiliser une dépénalisation différente en fonction des textes, représentés par les colonnes de X. Nous pouvons alors retravailler l'Équation (4.1) pour obtenir la fonction d'optimisation suivante :

$$\min_{W,H\geq 0} f(W,H,\lambda) = \sum_{i,j\in\kappa_+} |X-WH|_{i,j} + \sum_{i,j\in\kappa_0} \lambda_j |WH|_{i,j}, \tag{4.6}$$

avec λ un vecteur composé de n réels λ_q supérieurs à 0, $\kappa_+ = \{(i,j)|X_{i,j}>0\}$ et $\kappa_0 = \{(i,j)|X_{i,j}=0\}$ deux sous-ensembles complémentaires à l'ensemble des combinaisons d'indices (i,j).

4.3.1 Résolution du Modèle λ_q-ℓ_1-NMF

Détermination des valeurs de λ_q

Pour le modèle initial λ-ℓ_1-NMF, trouver la valeur de λ optimale était complexe. Pour ce nouveau modèle, il faut déterminer m valeurs λ_q. Il est donc impossible de tester chaque valeur individuellement pour atteindre les meilleurs résultats ou pour estimer des valeurs intéressantes.

Nous allons alors chercher une fonction pour définir λ_q en fonction de X. La fonction suivante considère que λ_q est proportionnel au ratio du nombre de zéros dans le texte j et du nombre de zéros moyen de tous les textes :

$$\lambda_q = \alpha * \frac{n|\kappa_{+_q}|}{|\kappa_+|}, \tag{4.7}$$

avec $\kappa_+ = \{(i,j)|X_{i,j} > 0\}$ l'ensemble des indices pour lesquelles les entrées de X sont non-nulles ; $\kappa_{+_q} = \{i|(i,q) \in \kappa_0\} = \{i|X_{i,q} > 0\}$ un sous-ensemble de κ_+ ; et $\alpha \in \mathbb{R}$ un paramètre nous permettant de faire varier l'ordre de grandeur des λ, α représentant la moyenne des λ_q. Dans notre cas, il vaut mieux travailler avec un α inférieur à 1. Ce paramètre a cependant de forte chance de dépendre de la matrice X.

Il est à noter que cette fonction peut permettre d'obtenir des λ_q supérieurs à 1. Ce phénomène se produit d'office si α est supérieur à 1, et est très probable au plus α se rapproche de 1. Si λ_q est supérieur à 1, ça veut dire que le modèle appliquera une plus grande pénalité sur les 0. Les textes qui posséderont un λ_q supérieur à 1 seront cependant des textes représentés par plus de mots que la moyenne (si α est inférieur ou égal à 1), et ces textes sont généralement déjà classifiés. Ce phénomène n'est donc pas un problème, et il peut même être avantageux de pénaliser plus les 0 pour ces textes.

Asymétrie de W et H

Contrairement aux modèles vu précédemment, le modèle λ_q-ℓ_1-NMF ne possède pas de symétrie entre W et H. Cette asymétrie est visible en regardant la fonction d'optimisation de descente de coordonnées pour H :

$$\min_{H_{p,q} \geq 0} f(H_{p,q}) = \sum_{i \in \kappa_{+_q}} |X - WH|_{i,q} + \lambda_q \sum_{i \in \kappa_{0_q}} |WH|_{i,q}, \tag{4.8}$$

et ensuite en regardant la fonction d'optimisation de descente de coordonnées pour W^T :

$$\min_{W^T_{p,s} \geq 0} f(W^T_{p,s}) = \sum_{j \in \kappa^T_{+_s}} |X^T - H^T W^T|_{j,s} + \sum_{j \in \kappa^T_{0_s}} \lambda_j |H^T W^T|_{j,s}, \tag{4.9}$$

avec :

— $\kappa_{+_q} = \{k|X_{k,j} > 0, j = q\}$ et $\kappa_{0_q} = \{k|X_{k,j} = 0, j = q\}$;

— $\kappa^T_{+_s} = \{k|X^T_{k,i} > 0, i = s\}$ et $\kappa^T_{0_s} = \{k|X^T_{k,i} = 0, i = s\}$.

Les différences sont assez maigres, mais il sera nécessaire de faire deux algorithmes, légèrement différents, pour résoudre la λ_q-ℓ_1-NMF.

Algorithmes de résolution

Les algorithmes de résolution sont principalement basés sur l'Algorithme 12. L'algorithme de mise à jour de $H_{p,q}$ est pratiquement identique à cet algorithme. Les seules différences sont l'argument λ qui devient un vecteur de dimension m, et la valeur de c qui devient $\lambda_q * (s_p - \sum_{k \in \kappa_{+,q}} W_{k,p})$.

L'algorithme de mise à jour de W est quant à lui légèrement différent. Celui-ci est montré en détail dans l'Algorithme 14.

Algorithme 14 $W^{(i)}$ = column_depenalizing_l1_W_coordinate_descent($X, W^{(i-1)}, H, \lambda$)

1: INPUT : $X \in \mathbb{R}^{m\times n}, W^{(i-1)} \in \mathbb{R}_+^{m\times r}, H \in \mathbb{R}^{r\times n}, \lambda \in [1;0]$
2: OUTPUT : $W^{(i)} \in \mathbb{R}^{m\times r}$
3: $W^{(i)} \leftarrow W^{(i-1)}$
4: $s_p = \sum_{j=1}^{n} H_{p,j} * \lambda_j \qquad \forall p = 1,...r$
5: **for** $s = 1 : m$ **do**
6: $\quad \kappa_{+_s}^T = \{k \mid X_{k,s}^T \neq 0\}$
7: $\quad v = H^T_{\kappa_{+_s}^T,:} W_{:,s}^{(i)^T}$
8: $\quad$ **for** $p = 1 : r$ **do**
9: $\quad\quad a \leftarrow X^T_{\kappa_{+_s}^T,p} - v + H^T_{\kappa_{+_s}^T,p} W_{p,s}^{(i)^T}$
10: $\quad\quad b \leftarrow H^T_{\kappa_{+_s}^T,p}$
11: $\quad\quad c \leftarrow s_p - \sum_{k\in\kappa_{+_s}^T} (H_{k,p}^T * \lambda_k)$
12: $\quad\quad W_{s,p}^{(i)} \leftarrow weighted\_median([a \quad 0], [b \quad c])$
13: $\quad\quad v \leftarrow v + H^T_{\kappa_{+_s}^T,p} (W_{s,p}^{(i)} - W_{s,p}^{(i-1)})$
14: $\quad$ **end for**
15: **end for**

En dehors des différences dues aux matrices transposées et aux changements d'indice, les différences notables sont surtout les lignes 4 et 11 de l'Algorithme 14. Ces deux lignes gèrent l'influence du vecteur λ sur le modèle.

Par soucis d'optimisation et de simplicité de code, il vaut mieux mettre directement en argument de l'Algorithme 14 les valeurs X^T, W^T et H^T, et modifier l'algorithme pour que les matrices transposées soient gérées directement.

4.3.2 Résultat du modèle lambda par colonne

Nous avons testé nos jeux de données provenant de [31] en utilisant la méthode λ_q-ℓ_1-NMF. Les λ_q seront déterminés par l'Équation (4.7) avec le paramètre $\alpha = 0.25$ pour le Tableau 4.3 et $\alpha = 0.1$ pour le Tableau 4.4. Nous comparerons les métriques de précision (Acc et Acc_0), proportion de colonnes non-nulles de H ($|\vec{e}H|_0$) et la "*sparsity*" de WH (S_{WH}).

Nous voyons sur ces tableaux que nos résultats de précisions sont le plus souvent assez similaires que les résultats pour la λ-ℓ_1-NMF classique (voir Tableaux 4.1 et 4.2). Chaque modèle performe plus ou moins bien, en fonction des paramètres et des jeux de données.

Nous pouvons cependant voir un avantage pour le modèle λ_q-ℓ_1-NMF : celui-ci produit généralement une matrice WH plus creuse que le modèle avec un λ global. Ce modèle est donc peut-être à préférer dans le cas de matrices creuses.

Un problème évident de ce modèle est que celui-ci dépend d'un paramètre α. Nous n'avons pas beaucoup traité le choix de ce paramètre dans le cadre de ce travail, mais il pourrait être intéressant de s'y attarder plus en détail si ce modèle venait à être repris dans d'autres travaux. L'Équation (4.7) déterminant la valeur des λ_q est également un paramètre du modèle, et cette fonction pourrait certainement être améliorée pour être efficace sur plus de jeux de données.

		λ_q-ℓ_1-NMF (α = 0.25)					ℓ_2-NMF
Dataset	r	*Acc* [%]	$\lvert\vec{e}H\rvert_0$ [%]	Acc_0 [%]	S_{WH} [%]	*Time* [s]	*Acc* [%]
NG20	20	10.99	87.03	12.63	99.942	53.88	23.08
ng3sim	3	29.29	84.42	34.69	99.844	2.14	36.06
classic	4	18.79	30.98	60.65	99.996	2.61	55.64
ohscal	10	27.25	94.33	28.89	99.74	27.43	30.50
k1b	6	64.83	93.29	69.49	99.624	11.08	59.19
hitech	6	48.59	96.09	50.57	98.969	36.53	46.94
reviews	5	73.29	86.63	84.6	99.391	13.94	51.19
sports	7	44.11	93.6	47.13	99.187	27.2	40.37
tr11	9	39.86	99.52	40.05	73.584	7.82	50.48
tr23	6	42.65	95.1	44.85	50.194	5.43	35.29
tr41	10	47.15	99.77	47.26	95.001	12.24	44.99
tr45	10	30.58	77.1	39.66	80.532	10.75	38.26

TABLE 4.3 – Comparaison des métriques présentées dans la section 2.5.2 de la λ_q-ℓ_1-NMF pour des valeurs de λ_q déterminées par l'Équation (4.7) avec $\alpha = 0.25$ et de la précision de la ℓ_2-NMF [10] sur les jeux de données provenant de [31]

		λ_q-ℓ_1-NMF (α = 0.1)					ℓ_2-NMF
Dataset	r	*Acc* [%]	$\lvert\vec{e}H\rvert_0$ [%]	Acc_0 [%]	S_{WH} [%]	*Time* [s]	*Acc* [%]
NG20	20	17.27	96.43	17.91	99.68	689.34	23.08
ng3sim	3	32.79	90.29	36.31	98.974	5.52	36.06
classic	4	32.62	66.45	49.09	99.945	4.89	55.64
ohscal	10	27.57	98.58	27.97	98.593	49.53	30.50
k1b	6	69.83	99.36	70.28	98.273	19.61	59.19
hitech	6	46.76	99.78	46.86	95.167	32.42	46.94
reviews	5	74.15	99.71	74.37	97.192	31.28	51.19
sports	7	56.85	98.95	57.46	97.196	86.0	40.37
tr11	9	28.99	100.0	28.99	51.585	18.77	50.48
tr23	6	41.18	100.0	41.18	21.041	2.17	35.29
tr41	10	56.15	100.0	56.15	81.468	72.33	44.99
tr45	10	33.19	100.0	33.19	54.141	7.86	38.26

TABLE 4.4 – Comparaison des métriques présentées dans la section 2.5.2 de la λ_q-ℓ_1-NMF pour des valeurs de λ_q déterminées par l'Équation (4.7) avec $\alpha = 0.1$ et de la précision de la ℓ_2-NMF [10] sur les jeux de données provenant de [31]

Finalement, nous pouvons regarder les résultats du modèle λ_q-ℓ_1-NMF avec le paramètre α à 0.1 sur le jeu de données TDT2. Ces résultats sont disponibles en détail dans l'Annexe D.

Nous pouvons noter quelques différences entre les résultats actuels et les résultats que nous avions obtenus dans la Section 2.5.3. Un des premiers points notable est la proportion de colonnes non-nulles de H, atteignant près de 100% pour le modèle λ_q-ℓ_1-NMF.

Nous constatons également qu'il y a beaucoup plus de mots dans les thèmes du dernier résultat que dans les thèmes du modèle initial. De plus, les thèmes semblent plus diversifiés. Pour la ℓ_1-NMF classique, nous obtenions beaucoup de thèmes similaires, voire identique. Hors, nos obtenons ici des thèmes qui semblent variés, avec quelques points communs de temps en temps.

4.4 Conclusions

Dans ce chapitre, nous avons utilisé une variante de la fonction objectif en norme ℓ_1 en y ajoutant un paramètre λ dépénalisant les zéros de la matrice X (voir Équation (4.1). Cette nouvelle fonction d'optimisation a permis d'obtenir de meilleurs résultats, permettant notamment de classifier une grande majorité des thèmes avec plus de précision.

Cette variante de la ℓ_1-NMF propose également une alternative à la réinstanciation. Avec un bon paramètre λ, nous obtenons alors des résultats meilleurs et plus rapides.

Nous avons ensuite construit une variante à ce nouveau modèle λ-ℓ_1-NMF, pour construire un modèle prenant en compte la diversité des textes d'un même jeu de données. Les textes courts, traitant moins de mots, sont alors moins pénalisés que des textes longs, qui peuvent traiter la majorité des mots de leurs thèmes. Ce modèle λ_q-ℓ_1-NMF a montré des résultats très satisfaisants sur la plupart des jeux de données.

Ces deux nouveaux modèles ont cependant un problème majeur, ils dépendent d'un ou plusieurs paramètres. Ces paramètres permettent d'obtenir des résultats différents, mais nous ne savons pas déduire les meilleurs paramètres permettant d'obtenir les résultats les plus probants en fonction du jeu de données que nous analysons.

Chapitre 5

Conclusions

5.1 Synthèse des Résultats

Dans ce travail, nous avons eu l'occasion de créer et d'analyser trois modèles principaux :

— ℓ_1-NMF : Le modèle de base de factorisation non-négative de matrices utilisant une fonction d'optimisation en norme ℓ_1.

— λ-ℓ_1-NMF : Une variante du modèle de base, dépénalisant les "zéros" de la matrice X pour éviter les faux-zéros. Ce modèle dépend du paramètre réel λ.

— λ_q-ℓ_1-NMF : Une variant du modèle dépénalisant les zéros. Ce modèle dépénalise plus ou moins les zéros en fonction du nombre de mots que chaque texte a pu traiter. Ce modèle dépend de deux paramètres, la fonction $\lambda_q = f_q(X, \alpha)$ (voir Équation 4.7) et le réel α.

D'autres variantes plus minimes ont également été mises en place. Une synthèse de l'ensemble des résultats est disponible à l'Annexe E.

Nous allons comparer une dernière fois nos résultats entre eux. Nous utiliserons cette fois-ci la métrique de "précision redressée". Cette métrique reprend la précision du modèle en considérant que tous les textes ont été classifiés. Pour les textes qui n'ont pas été classifiés par le modèle, nous considérerons que leur classe est aléatoire.

Le Tableau 5.1 reprend l'ensemble des résultats que nous avons obtenus ainsi que ceux résultants d'autres méthodes de factorisation non-négative de matrices : la NMF en norme ℓ_2 (ℓ_2-NMF ou FRO-NMF) [8], la NMF utilisant la divergence Kullback-Leibler (KL-NMF) [14] et la "*Distributionally Robust*" NMF (DR-NMF) [10]. Les résultats de ces trois méthodes proviennent de [10].

Nous comparons ici ces formes de NMF avec le modèle ℓ_1-NMF (avec réinstanciation et descente de coordonnées classique CD) (voir Tableau 3.2) ; le modèle λ-ℓ_1-NMF avec $\lambda = \sqrt{1 - S_X}/2$ (voir Tableau 4.1) et avec λ obtenu en descendant progressivement sa valeur (voir Tableau 4.2) ; le modèle λ_q-ℓ_1-NMF utilisant l'Équation (4.7) pour calculer les λ_q pour des valeurs de α de 0.25 et 0.1.

Sur le Tableau 5.1, pour un même jeu de données, la plus grande précision sera en gras, et la deuxième plus grande précision sera soulignée.

Nous pouvons observer que c'est le modèle λ_q-ℓ_1-NMF qui s'en sort assez bien dans la comparaison avec les autres modèles de NMF en termes de précision. La KL-NMF semble cependant globalement meilleure, et surtout beaucoup plus stable que la ℓ_1-NMF et ses variantes, qui s'en sortent assez mal sur les premiers jeux de données.

		Clustering Accuracy [%]			Clustering Redressed Accuracy [%]				
		ℓ_2-NMF	KL-NMF	DR-NMF	ℓ_1-NMF	λ-ℓ_1-NMF		λ_q-ℓ_1-NMF	
Dataset	r					$\lambda = \frac{\sqrt{1-S_X}}{2}$	λ Decrising	$\alpha = 0.25$	$\alpha = 0.1$
NG20	20	23.08	**42.15**	28.74	12.84	17.14	19.5	11.64	17.45
ng3sim	3	38.06	**63.48**	49.87	34.71	36.52	37.68	34.48	36.02
classic	4	55.64	**83.66**	78.46	33.91	53.18	53.25	36.04	41.01
ohscal	10	30.50	**37.45**	32.13	24.47	31.96	25.17	27.82	27.71
k1b	6	59.19	64.27	60.13	69.02	68.92	67.95	65.95	**69.94**
hitech	6	46.94	43.29	48.02	44.84	41.13	40.35	**49.24**	46.8
reviews	5	51.19	75.65	74.88	63.92	59.72	62.54	**75.96**	74.2
sports	7	40.37	43.93	50.26	37.34	43.71	45.02	45.03	**57.0**
tr11	9	50.48	**62.32**	51.45	15.06	40.96	39.67	39.91	28.99
tr23	6	35.29	34.80	38.73	24.02	28.51	36.68	**43.46**	41.18
tr41	10	44.99	54.33	53.08	31.09	32.55	38.72	47.18	**56.15**
tr45	10	38.26	**46.81**	39.13	14.35	33.04	34.97	32.87	33.19
Average		42.83	54.34	50.41	33.80	40.61	41.79	42.47	44.14

TABLE 5.1 – Comparaison des précisions de la ℓ_2-NMF, KL-NMF et DR-NMF [10] avec les précisions redressées de la ℓ_1-NMF et de ses variantes sur les jeux de données provenant de [31].

Les meilleures précisions sont très partagées entre les modèles λ_q-ℓ_1-NMF et KL-NMF, mais le modèle DR-NMF finit souvent second, et montre une grande stabilité dans la classification non-supervisée de texte. La ℓ_1-NMF classique produit quant à elle les moins bons résultats de tous les modèles, tandis que la λ-ℓ_1-NMF produit des résultats légèrement en dessous de ceux de la variante λ_q-ℓ_1-NMF, mais de meilleurs résultats pour les jeux de données où le modèle dépénalisant par colonnes sous-performe.

La précision de la ℓ_1-NMF dépend de la manière dont les textes non-classifiés sont redistribués dans des classes. La précision des modèles pourrait être encore légèrement améliorée si ces textes étaient redistribués de manière intelligente.

5.2 Perspectives d'Améliorations

5.2.1 Initialisation

L'initialisation de la ℓ_1-NMF est très importante car, si la matrice X est creuse, nous ne pouvons pas initialiser W et H par des matrices aléatoires, remplies de 0 ou remplies de 1, comme c'est fait habituellement. Si nous faisions ça, l'algorithme convergerait vers le minimum local $W = 0^{m\times r}$ et $H = 0^{r\times n}$ (voir Section 2.3).

Les résultats obtenus dans ce travail utilisent tous la ℓ_2-NMF pour initialiser les matrices W et H, mais il pourrait être intéressant de se pencher sur d'autres méthodes d'initialisation.

L'initialisation et la réinstanciation (voir Section 3.2) sont deux choses liées. Pour réinstancier une colonne, il nous faut une méthode d'initialisation que nous appliquerons sur $Z = X - WH$ pour une valeur r de 1. Nous avons eu l'occasion de traiter la méthode d'initialisation *greedy*-ℓ_1-NMF (voir Algorithme 7) pour la réinstanciation.

Après quelques tests, cette méthode ne semble pas montrer de très bons résultats pour l'initialisation. Le vecteur h créé par la *greedy*-ℓ_1-NMF n'est généralement pas assez creux et provoque des résultats moyens pour une matrice X creuse. Peut-être que cette méthode pourrait être améliorée en forçant le vecteur h à être plus creux.

La Section 4.2.2 montre que pour un $\lambda = 0$, la méthode λ-ℓ_1-NMF produira un résultat non-creux (sauf si les matrices W_0 et H_0 sont remplies de 0) et n'atteindra pas le minimum local $W = 0^{m\times r}$ et $H = 0^{r\times n}$. L'Algorithme 15 propose alors une méthode d'initialisation, avec $1^{N\times M}$ une matrice remplie de 1 avec N lignes et M colonnes.

Algorithme 15 $[W,H]$ = initialize_with_l1(X,r)

INPUT : $X \in \mathbb{R}^{m\times n}, r \in \mathbb{N}$
OUTPUT : $W \in \mathbb{R}_+^{m\times r}, H \in \mathbb{R}^{r\times n}$
$[W,H] \leftarrow [1^{m\times r}, 1^{r\times n}]$
while stop criterion not satisfied **do**
    $W \leftarrow depenalizing\_l1\_coordinate\_descent(X^T, H^T, W^T, \lambda = 0.0)^T$
    $H \leftarrow depenalizing\_l1\_coordinate\_descent(X, W, H, \lambda = 0.0)$
end while

Cependant, l'Algorithme 15 produit également un résultat non-creux, et la ℓ_1-NMF risquera toujours de converger vers des colonnes de W et des lignes de H nulles. La Section 4.2.3 nous montre que les matrices W et H sont de plus en plus creuses au plus λ augmente. Une méthode pour que les matrices W_0 et H_0 soient creuses serait alors d'augmenter progressivement la valeur de λ pour obtenir un résultat de plus en plus creux.

D'autres méthodes de NMF pourraient être utilisées pour initialiser la ℓ_1-NMF, comme la KL-NMF ou la DR-NMF qui produisent déjà de meilleurs résultats que la ℓ_2-NMF. La ℓ_1-NMF améliorerait peut-être les résultats de ces méthodes pour des jeux de données creux.

5.2.2 Autres Variantes de la ℓ_1-NMF

1-sparse-ℓ_1-NMF

La NMF permet de factoriser une matrice X non-négative. Hors, dans ce travail, nous évaluons nos modèles de NMF par leur capacité à classifier les textes de la matrice X. Il semblerait pertinent d'imaginer un modèle qui n'attribuerait alors qu'une seule classe à chaque colonne de H, c'est-à-dire que $|\{H_{i,q} \neq 0 | i = 1,...r\}| = 1$ pour tout $q = 1,...n$.

Ce genre de modèle pourrait être appelé 1-sparse-NMF si la contrainte est appliquée aux matrices W et H, ou semi-1-sparse-NMF si cette contrainte n'est appliquée qu'à la matrice H. Dans le cas de la semi-1-sparse-ℓ_1-NMF, la fonction objectif peut être réécrite de la manière suivante :

$$\begin{aligned} \min_{W,H} f(W,H) = &||X - WH||_1 = \sum_{i,j} |X - WH|_{i,j} \\ s.t. \quad & W,H \geq 0, \\ & |\{H_{i,q} \neq 0 | i = 1,...r\}| = 1 \qquad \forall q = 1,...n. \end{aligned} \tag{5.1}$$

L'Algorithme 16 propose une méthode de résolution à la 1-sparse-ℓ_1-NMF. Cette méthode sélectionne l'indice p^* améliorant le plus la fonction objectif de la colonne q de H. Il pose ensuite $H_{p,q} = 0$ pour $p \neq p^*$ et met à jour uniquement $H_{p^*,q}$.

Comme pour la ℓ_1-NMF, le paramètre λ peut être ajouté pour dépénaliser les éléments nuls de X. Nous produirons ainsi moins de zéros dans les matrices W et H, tout en pouvant atteindre des résultats creux.

Algorithme 16 $H^{(i)}$ = depenalizing_1-sparse_l1_coordinate_descent(X, W, λ)

INPUT : $X \in \mathbb{R}^{m\times n}, W \in \mathbb{R}_+^{m\times r}, \lambda \in [1;0]$
OUTPUT : $H^{(i)} \in \mathbb{R}^{r\times n}$
$H^{(i)} \leftarrow 0^{r\times n}$
$s_p = \sum_{j=1}^{m} W_{j,p} \qquad \forall p = 1,...r$
for $q = 1 : n$ **do**
    $\kappa_{+_q} = \{k \mid X_{k,q} \neq 0\}$
    for $p = 1 : r$ **do**
        $a \leftarrow X_{\kappa_{+_q},p}$
        $b \leftarrow W_{\kappa_{+_q},p}$
        $c \leftarrow \lambda * (s_p - \sum_{k\in\kappa_{+_q}} W_{k,p})$
        $h_p \leftarrow weighted\_median([a \quad 0],[b \quad c])$
        $obj_p \leftarrow |X_{:,q} - W_{:,p} * h_p|_1$
    end for
    $p^* = \text{argmin}_p\, obj_p$
    $H_{p^*,q} \leftarrow h_{p^*}$
end for

La semi-1-sparse-ℓ_1-NMF a le potentiel d'être meilleure pour la classification.

Cependant, après quelques tests, il semble que la méthode a tendance à ne pas converger. Elle boucle en effet souvent entre deux solutions, ce qui n'est habituellement pas le cas de la ℓ_1-NMF, qui garantit d'obtenir à chaque itération une solution meilleure ou identique à la précédente. Ce phénomène vient sûrement du fait que la matrice H est réinitialisée à chaque itération. Pourtant, ça ne devrait pas provoquer la non-convergence si seul un élément de h est non-nul.

Une étude plus approfondie et plus rigoureuse de la convergence de la ℓ_1-NMF pourrait résoudre ce problème.

increasing-λ-ℓ_1-NMF

Nous avons énoncé dans la Section 5.2.1 une méthode d'initialisation commençant avec $\lambda = 0$ et augmentant progressivement la valeur de λ. Cette méthode d'initialisation pourrait être une méthode à part entière. Cette méthode augmenterait la valeur de λ jusqu'à atteindre un critère d'arrêt. Ce critère d'arrêt pourrait par exemple être posé sur la "sparsity" de WH : $S_{WH} \geq threshold$.

Autres Modèles

Le travail [12] porte également sur la ℓ_1-NMF pour des cas généraux, alors que nous nous sommes concentrés sur les matrices creuses. Ce travail propose plusieurs variantes intéressantes de la ℓ_1-NMF qui valent la peine d'être étudiées en détails.

Il est également possible de s'inspirer des travaux existants sur la ℓ_2-NMF [22, 25] qui proposent de nouvelles variantes en norme de Frobenius, parfois adaptables à la ℓ_1-NMF.

5.2.3 Choix des Paramètres de la λ-ℓ_1-NMF

En regardant le Tableau 5.1, les méthodes λ-ℓ_1-NMF et λ_q-ℓ_1-NMF sont les deux méthodes en norme ℓ_1 ayant le plus de potentiel sur nos jeux de données. Ces deux méthodes dépendent de paramètres λ et λ_q influençant les résultats.

Il pourrait être intéressant de mener des études plus rigoureuses sur le choix de ces paramètres. Dans ce travail, ils ont été sélectionnés de manière arbitraire. Nous avons choisi des valeurs permettant de donner de bons résultats pour nos jeux de données. Cependant, pour d'autres jeux de données, nos fonctions déterminant les paramètres pourraient ne pas fonctionner. Il serait alors intéressant de trouver des méthodes moins arbitraires permettant de sélectionner nos paramètres en fonction des jeux de données qui sont analysés.

5.2.4 Matrices Binaires

Nous avons vu dans la Section 2.4 que la ℓ_1-NMF produisait de très bons résultats sur des jeux de données synthétiques binaires. Nous n'avons cependant pas essayé nos méthodes sur des jeux de données binaires réels. Il pourrait être intéressant de s'intéresser au potentiel de la ℓ_1-NMF pour la factorisation de matrices binaires. Des travaux comme [29, 30] traitent de factorisation de matrices binaire et pourraient servir de comparaison.

De plus, dans des cas binaires, il existe plusieurs méthodes permettant d'améliorer le temps de traitement de la ℓ_1-NMF. Nous avions montré dans la Section 2.3.3 l'existence d'un Algorithme de descente de gradient avec une complexité $\mathcal{O}(n)$. De plus, une étude plus approfondie de la méthode S-CD (voir Section 3.3.1) pourrait également augmenter la complexité d'un algorithme de résolution de ℓ_1-NMF pour une matrice X binaire.

5.2.5 Autres

Dans la Section 2.5.3, nous avons soulevé le problème des clusters similaires de la ℓ_1-NMF. Ce problème devrait toucher toutes les formes de factorisation de matrices. Il pourrait être assez intéressant de se pencher sur une solution à ce problème. Des termes de régularisation ajoutant une pénalité sur les lignes de W ou des contraintes limitant la similarité entre les colonnes de W pourraient être ajoutés pour résoudre ce problème.

Une caractéristique de nos jeux de données présentée dans la section 2.5.1 était la "*balance*", qui permettait de voir si les textes étaient répartis de manière équilibrée ou non dans les classes pour un même jeu de données. Nous n'avons pas exploité cette caractéristique dans notre travail. Cependant, il pourrait être intéressant de voir si la "*balance*" a un impact sur nos résultats, et analyser dans quelle mesure la ℓ_1-NMF est capable de classifier un jeu de données équilibré ou non.

De plus, un jeu de données dont les textes sont répartis inéquitablement entre les classes peut provoquer des biais dans la mesure de précision. Par exemple, pour un jeu de données à 5 classes, si les 50% des textes appartiennent à la classe 1, un algorithme classifiant tous les textes dans la classe 1 est assuré d'avoir 50% de précision. Pourtant, nous ne devrions pas nous satisfaire d'un tel algorithme.

Même pour les méthodes λ-ℓ_1-NMF, nous n'atteignons généralement pas une proportion de textes classifiés de 100%. Nous avons utilisé dans le Tableau 5.1 la notion de "précision redressée" où nous classions les textes non-classifiés aléatoirement. Nous pourrions cependant leur attribuer une classe plus intelligemment. Un exemple simple serait d'attribuer à un texte non-classifié le thème pour lequel il aurait le plus de corrélation (voir Équation 3.2).

Nous pourrions aussi nous dire que la matrice H n'est pas idéale pour mesurer la précision. Nous pourrions utiliser la matrice de corrélation $W^T X$ pour attribuer l'ensemble des classes. Il suffirait d'injecter dans la mesure de précision (voir Algorithme 5) la matrice $W^T X$ à la place de la matrice H. Même pour cette méthode de mesure de précision, il faut garder la matrice W normalisée, comme elle l'était déjà initialement (voir Section 2.5.2).

Pour finir, une dernière amélioration possible à citer serait l'amélioration des critères d'arrêt. Dans plusieurs algorithmes (voir Algorithmes 1, 7, 11, 13 et 15), nous avons introduit la notion de critère d'arrêt, sans jamais réellement la développer. Nous nous contentons souvent d'un critère de convergence ou d'un nombre maximum d'itérations. Il pourrait être intéressant de chercher des critères d'arrêt permettant d'améliorer les résultats, diminuer le nombre d'itérations effectuées, ...

5.3 Applications de la ℓ_1-NMF

Les applications de la NMF sont assez vastes. Nous avons déjà pu traiter certaines de ces applications dans la Section 1.4. La ℓ_1-NMF permet évidemment de traiter toutes ces applications. Cependant, elle devrait avoir plus de potentiel dans le cas de jeux de données creux ou binaires. Parmi ce genre de données, nous retrouvons évidemment l'extraction de thèmes et le clustering de textes, mais nous pouvons citer d'autres applications :

- Dans la bio-informatique, les données génomiques sont représentées par des entrées binaires, représentant la présence ou l'absence d'un gène dans une séquence d'ADN [6]. La ℓ_1-NMF pourrait regrouper les séquences similaires entre elles pour permettre de les étudier plus facilement.
- Les systèmes de recommandation sont des problèmes communs dans beaucoup de branches, tels que les sites d'achats en ligne [2]. Les données à traiter dans ces domaines sont souvent creuses ou binaires. La ℓ_1-NMF peut être utilisée pour regrouper les utilisateurs ou les produits similaires et ainsi améliorer le système de recommandation.
- Un problème similaire aux systèmes de recommandation est la segmentation de clientèle par habitude d'achats. Les clients peuvent être caractérisés par le nombre d'achats qu'ils effectuent dans certaines catégories de produits [20]. La ℓ_1-NMF peut regrouper des thèmes d'achats récurrents et y assigner des clients.

Nous pouvons parler d'une dernière application plus théorique à la ℓ_1-NMF. L'initialisation des problèmes d'apprentissage automatique (*Machine Learning*) est souvent un problème en soi, et il est complexe de trouver de bonnes méthodes d'initialisation. De la même manière que nous avons utilisé la ℓ_2-NMF pour initialiser nos matrices W et H, nous pourrions utiliser la ℓ_1-NMF pour beaucoup d'algorithmes de *machine learning*, tel que les problèmes "k-means", les poids des réseaux de neurones ou même d'autres types de NMF.

5.4 Conclusions

Dans ce travail, nous nous sommes concentrés sur l'élaboration de différents modèles de Factorisation Non-Négative de Matrices utilisant la norme ℓ_1 comme fonction objectif. Ces modèles avaient principalement pour but de factoriser des matrices creuses ou binaires. Les modèles proposés ont alors été construits pour réduire leur complexité algorithmique dans le cas de matrices creuses.

Le premier modèle a été appelé ℓ_1-NMF. Les expériences sur des jeux de données synthétiques binaires ont permis de montrer le potentiel de ce modèle. Nous avons ensuite pu expérimenter sur des jeux de données réels, composés de textes représentés par le nombre d'occurrences des mots qu'ils contiennent. Nous avons ainsi pu identifier les défauts de nos modèles.

La suite du travail s'est concentrée sur la correction de ces défauts. Nous avons d'abord proposé des variantes mineures au modèle de base, améliorant sensiblement les résultats. Un modèle accéléré a été conçu, mais il n'a pas produit de résultats convaincants et a été rejeté.

Les jeux de données réels sont souvent contaminés par de faux zéros. Deux méthodes dépénalisant les zéros ont été proposées afin d'éviter que notre modèle se contraigne à prédire les faux zéros. Les nouvelles variantes, appelées λ-ℓ_1-NMF et λ_q-ℓ_1-NMF ont été testées sur les jeux de données réels. Nous avons alors pu comparer nos méthodes avec différents modèles de NMF. Nos modèles se sont avérés performants pour certains jeux de données, mais sous-performants pour d'autres, ce qui montre leur potentiel, mais montre qu'ils peuvent encore être améliorés.

Nos modèles finaux ont été plutôt satisfaisants et ont montré un bon potentiel pour la factorisation de matrices creuses et pour la classification non-supervisée. Ils pourraient certainement être encore améliorés et des travaux futurs pourraient être menés sur les modèles présentés afin de les étudier de manière plus théorique.

Bibliographie

[1] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization–provably. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 145–162, 2012.

[2] Irina Beregovskaya and Mikhail Koroteev. Review of clustering-based recommender systems. *arXiv preprint arXiv :2109.12839*, 2021.

[3] David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4) :77–84, 2012.

[4] Chaya Bleich and Michael L Overton. A linear-time algorithm for the weighted median problem. Technical report, 1983.

[5] P. De Handschutter. Project in first-order methods for large-scale machine learning. Slides, 2021.

[6] Rui Dong, Lily He, Rong Lucy He, and Stephen S-T Yau. A novel approach to clustering genome sequences using inter-nucleotide covariance. *Frontiers in Genetics*, 10 :234, 2019.

[7] Jon Fiscus, George Doddington, John Garofolo, and Alvin Martin. Nist's 1998 topic detection and tracking evaluation (TDT2). In *Proceedings of the 1999 DARPA Broadcast News Workshop*, pages 19–24, 1999.

[8] Nicolas Gillis. The why and how of nonnegative matrix factorization. *arXiv preprint arXiv :1401.5226*, 2014.

[9] Nicolas Gillis and François Glineur. Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization. *Neural Computation*, 24(4) :1085–1105, 2012.

[10] Nicolas Gillis, Valentin Leplat, Vincent YF Tan, et al. Distributionally robust and multi-objective nonnegative matrix factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8) :4052–4064, 2021.

[11] Nicolas Gillis and Stephen A Vavasis. On the complexity of robust PCA and ℓ_1-norm low-rank matrix approximation. *Mathematics of Operations Research*, 43(4) :1072–1084, 2018.

[12] Naiyang Guan, Dacheng Tao, Zhigang Luo, and John Shawe-Taylor. MahNMF : Manhattan non-negative matrix factorization. *arXiv preprint arXiv :1207.3438*, 2012.

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

[14] Le Thi Khanh Hien and Nicolas Gillis. Algorithms for nonnegative matrix factorization with the kullback–leibler divergence. *Journal of Scientific Computing*, 87(3) :93, 2021.

[15] Qifa Ke and Takeo Kanade. Robust subspace computation using l1 norm. 2003.

[16] Qifa Ke and Takeo Kanade. Robust ℓ_1 norm factorization in the presence of outliers and missing data by alternative convex programming. *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1 :739 – 746 vol. 1, 2005.

[17] Oscar F López, Daniel M Dunlavy, and Richard B Lehoucq. Zero-truncated poisson regression for sparse multiway count data corrupted by false zeros. *Information and Inference : A Journal of the IMA*, 12(3) :iaad016, 2023.

[18] Deyu Meng. Divide-and-conquer method for $\ell_1 1$ norm matrix factorization in the presence of outliers and missing data. *CoRR*, abs/1202.5844, 2012.

[19] James Munkres. Algorithms for the assignment and transportation problems. *Society for Industrial and Applied Mathematics*, 15 :196–210, 1962.

[20] Morteza Namvar, Mohammad R Gholamian, and Sahand KhakAbi. A two phase clustering method for intelligent customer segmentation. In *2010 International Conference on Intelligent Systems, Modelling and Simulation*, pages 215–219. IEEE, 2010.

[21] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103 :127–152, 2005.

[22] Ankan Saha and Vikas Sindhwani. Learning evolving and emerging topics in social media : a dynamic nmf approach with temporal regularization. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 693–702, 2012.

[23] Arnaud Vandaele, François Moutier, and Nicolas Gillis. ℓ_1-nmf for sparse data. Slides, 2022. Seminar COLORAMAP.

[24] Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3) :1364–1377, 2010.

[25] Jonghye Woo, Jerry L. Prince, Maureen Stone, Fangxu Xing, Arnold D. Gomez, Jordan R. Green, Christopher J. Hartnick, Thomas J. Brady, Timothy G. Reese, Van J. Wedeen, and Georges El Fakhri. A sparse non-negative matrix factorization framework functional units of tongue behavior from MRI. *IEEE Transactions on Medical Imaging*, 38(3) :730–740, 2019.

[26] Stephen J Wright. Coordinate descent algorithms. *Mathematical programming*, 151(1) :3–34, 2015.

[27] Chunhui Yuan and Haitao Yang. Research on K-value selection method of K-means clustering algorithm. *J — Multidisciplinary Scientific Journal*, 2(2) :226–235, 2019.

[28] Nik Sarah Nik Zamri and Zamira Hasanah Zamzuri. A review on models for count data with extra zeros. In *AIP Conference Proceedings*, volume 1830. AIP Publishing LLC, 2017.

[29] Zhong-Yuan Zhang, Tao Li, Chris Ding, Xian-Wen Ren, and Xiang-Sun Zhang. Binary matrix factorization for analyzing gene expression data. *Data Mining and Knowledge Discovery*, 20 :28–52, 2010.

[30] Zhongyuan Zhang, Tao Li, Chris Ding, and Xiangsun Zhang. Binary matrix factorization with applications. In *Seventh IEEE international conference on data mining (ICDM 2007)*, pages 391–400. IEEE, 2007.

[31] Shi Zhong and Joydeep Ghosh. Generative model-based document clustering : a comparative study. *Knowledge and Information Systems*, 8 :374–384, 2005.

Annexes

Annexe A : Résultats de l'algorithme ℓ_1-NMF sur le jeu de données TDT2

```
MOTS DU THÈME 1     - Taille = 0

MOTS DU THÈME 2     - Taille = 13
1 : week            - 1.0
2 : time            - 0.268457
3 : world           - 0.246075
4 : washington      - 0.144194
5 : clinton         - 0.134416
6 : crisis          - 0.125087
7 : government      - 0.114184
8 : american        - 0.099650
9 : people          - 0.091705
10: officials       - 0.088360

MOTS DU THÈME 3     - Taille = 1
1 : president       - 1.0

MOTS DU THÈME 4     - Taille = 12
1 : government      - 1.0
2 : week            - 0.833981
3 : world           - 0.832969
4 : clinton         - 0.625795
5 : officials       - 0.524195
6 : crisis          - 0.450783
7 : united          - 0.439019
8 : american        - 0.437333
9 : people          - 0.375069
10: news            - 0.218085

MOTS DU THÈME 5     - Taille = 21
1 : week            - 1.0
2 : officials       - 0.636897
3 : washington      - 0.520871
4 : minister        - 0.454188
5 : news            - 0.348669
6 : ap              - 0.233072
7 : told            - 0.220589
8 : clinton         - 0.219654
9 : government      - 0.210453
10: crisis          - 0.170194

MOTS DU THÈME 6     - Taille = 0

MOTS DU THÈME 7     - Taille = 13
1 : clinton         - 1.0
2 : crisis          - 0.886139
3 : officials       - 0.429441
4 : american        - 0.417842
5 : president       - 0.345588
6 : washington      - 0.304718
7 : united          - 0.183464
8 : time            - 0.139699
9 : government      - 0.139302
10: week            - 0.133186

MOTS DU THÈME 8     - Taille = 1
1 : president       - 1.0

MOTS DU THÈME 9     - Taille = 41
1 : minister        - 1.0
2 : dont            - 0.552132
3 : called          - 0.503914
4 : day             - 0.462743
5 : economic        - 0.411125
6 : united          - 0.366567
7 : military        - 0.333370
8 : weapons         - 0.290518
9 : days            - 0.276117
10: foreign         - 0.231339

MOTS DU THÈME 10    - Taille = 2
1 : people          - 1.0
2 : president       - 0.080755

MOTS DU THÈME 11    - Taille = 32
1 : american        - 1.0
2 : united          - 0.671413
3 : clinton         - 0.667719
4 : people          - 0.578695
5 : international   - 0.571530
6 : president       - 0.485455
7 : weapons         - 0.316100
8 : secretary       - 0.282006
9 : iraq            - 0.279786
10: country         - 0.277233

MOTS DU THÈME 12    - Taille = 8
1 : time            - 1.0
2 : people          - 0.492688
3 : president       - 0.445799
4 : government      - 0.402833
5 : news            - 0.330727
```

```
6 : clinton         - 0.252983
7 : united          - 0.204819
8 : world           - 0.029895

MOTS DU THÈME 13    - Taille = 5
1 : people          - 1.0
2 : president       - 0.501849
3 : united          - 0.072529
4 : news            - 0.047207
5 : time            - 0.044285

MOTS DU THÈME 14    - Taille = 26
1 : news            - 1.0
2 : american        - 0.603324
3 : world           - 0.444430
4 : crisis          - 0.357812
5 : day             - 0.301152
6 : people          - 0.273359
7 : house           - 0.266884
8 : economic        - 0.202344
9 : minister        - 0.200825
10: secretary       - 0.190105

MOTS DU THÈME 15    - Taille = 2
1 : people          - 1.0
2 : president       - 0.010604

MOTS DU THÈME 16    - Taille = 18
1 : crisis          - 1.0
2 : week            - 0.270965
3 : minister        - 0.179861
4 : day             - 0.173297
5 : clinton         - 0.167563
6 : economic        - 0.140004
7 : world           - 0.086482
8 : washington      - 0.080739
9 : house           - 0.080096
10: american        - 0.064017

MOTS DU THÈME 17    - Taille = 18
1 : time            - 1.0
2 : world           - 0.678765
3 : crisis          - 0.400058
4 : minister        - 0.233743
5 : clinton         - 0.233160
6 : economic        - 0.229184
7 : american        - 0.199071
8 : officials       - 0.194341
9 : ap              - 0.178186
10: government      - 0.166570

MOTS DU THÈME 18    - Taille = 27
1 : house           - 1.0
2 : economic        - 0.859534
3 : country         - 0.803637
4 : told            - 0.650125
5 : nations         - 0.498041
6 : iraq            - 0.482764
7 : washington      - 0.458252
8 : day             - 0.313297
9 : american        - 0.277939
10: officials       - 0.277208

MOTS DU THÈME 19    - Taille = 0

MOTS DU THÈME 20    - Taille = 21
1 : crisis          - 1.0
2 : economic        - 0.770584
3 : day             - 0.740927
4 : house           - 0.418416
5 : ap              - 0.206371
6 : government      - 0.165834
7 : minister        - 0.154238
8 : told            - 0.105962
9 : united          - 0.093779
10: international   - 0.088636

Proportion de colonnes de H
non-nulles : 82.10%
Proportion de colonnes de WtX
non-nulles : 98.79%
```

Annexe B : Variantes Accélérées des Modèles de la ℓ_1-NMF

$H^{(i)}$ = accelerated_random_coordinate_descent($X, W, H^{(i-1)}$)

INPUT : $X \in \mathbb{R}^{m \times n}, W \in \mathbb{R}_+^{m \times r}, H^{(i-1)} \in \mathbb{R}^{r \times n}, \lambda \in [1;0]$
OUTPUT : $H^{(i)} \in \mathbb{R}^{r \times n}$
$s_p = \sum_{j=1}^{m} W_{j,p} \qquad \forall p = 1, ...r$
for $q = 1 : n$ **do**
    $\kappa = \{k \mid X_{k,q} \neq 0\}$
    $v = W_{\kappa,:} H^{(i)}_{:,q}$
    $index = shuffle([1 : r])$
    while stop crierion not satisfied **do**
        $h_{old} \leftarrow H_{:,q}$
        for $p \in index$ **do**
            $a \leftarrow X_{\kappa,p} - v + W_{\kappa,p} H^{(i)}_{p,q}$
            $b \leftarrow W_{\kappa,p}$
            $c \leftarrow s_p - \sum_{k \in \kappa} W_{k,p}$
            $H^{(i)}_{p,q} \leftarrow constrained\_weighted\_median([a \quad 0], [b \quad c])$
            $v \leftarrow v + W_{\kappa,p}(H^{(i)}_{p,q} - h_{old_p})$
        end for
    end while
end for

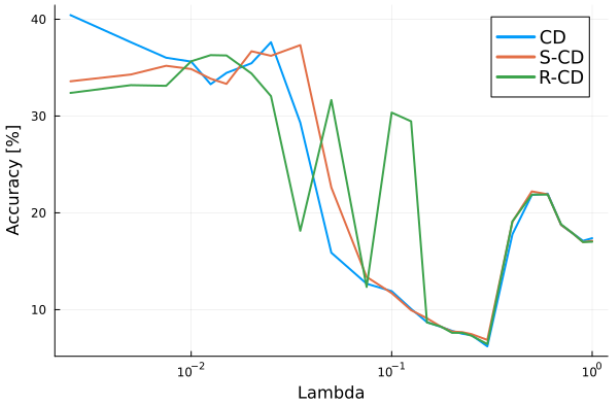
$H^{(i)}$ = accelerated_slope_coordinate_descent($X, W, H^{(i-1)}$)

INPUT : $X \in \mathbb{R}^{m \times n}, W \in \mathbb{R}_+^{m \times r}, H^{(i-1)} \in \mathbb{R}^{r \times n}, \lambda \in [1;0]$
OUTPUT : $H^{(i)} \in \mathbb{R}^{r \times n}$
$s_p = \sum_{j=1}^{m} W_{j,p} \qquad \forall p = 1, ...r$
for $q = 1 : n$ **do**
    $\kappa = \{k \mid X_{k,q} \neq 0\}$
    $v = W_{\kappa,:} H^{(i)}_{:,q}$
    for $p = 1 : r$ **do**
        $\Delta_p = 2 \sum_{i \in \kappa} W_{i,p} - s_p$
    end for
    $[\Delta, index] \leftarrow sort(\Delta)$
    while stop crierion not satisfied **do**
        $h_{old} \leftarrow H_{:,q}$
        for $p \in index$ **do**
            $a \leftarrow X_{\kappa,p} - v + W_{\kappa,p} H^{(i)}_{p,q}$
            $b \leftarrow W_{\kappa,p}$
            $c \leftarrow s_p - \sum_{k \in \kappa} W_{k,p}$
            $H^{(i)}_{p,q} \leftarrow constrained\_weighted\_median([a \quad 0], [b \quad c])$
            $v \leftarrow v + W_{\kappa,p}(H^{(i)}_{p,q} - h_{old_p})$
        end for
    end while
end for

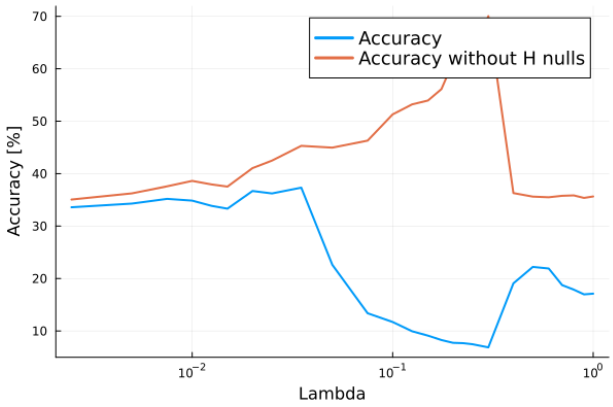
$H^{(i)}$ = accelerated_correlated_coordinate_descent($X, W, H^{(i-1)}$)

INPUT : $X \in \mathbb{R}^{m \times n}, W \in \mathbb{R}_+^{m \times r}, H^{(i-1)} \in \mathbb{R}^{r \times n}, \lambda \in [1;0]$
OUTPUT : $H^{(i)} \in \mathbb{R}^{r \times n}$
$s_p = \sum_{j=1}^{m} W_{j,p} \qquad \forall p = 1, ...r$
for $q = 1 : n$ **do**
    $\kappa = \{k \mid X_{k,q} \neq 0\}$
    $v = W_{\kappa,:} H^{(i)}_{:,q}$
    $\rho = W^T X_{:,q}$
    $[\rho, index] \leftarrow sort(\rho)$
    while stop crierion not satisfied **do**
        $h_{old} \leftarrow H_{:,q}$
        for $p \in index$ **do**
            $a \leftarrow X_{\kappa,p} - v + W_{\kappa,p} H^{(i)}_{p,q}$
            $b \leftarrow W_{\kappa,p}$
            $c \leftarrow s_p - \sum_{k \in \kappa} W_{k,p}$
            $H^{(i)}_{p,q} \leftarrow constrained\_weighted\_median([a \quad 0], [b \quad c])$
            $v \leftarrow v + W_{\kappa,p}(H^{(i)}_{p,q} - h_{old_p})$
        end for
    end while
end for

Annexe C : Résultats des Différents Modèles de Descente de Coordonnées sur le Modèle la λ-ℓ_1-NMF sur les Jeux de Données *ng3sim*, *sports* et *tr45*

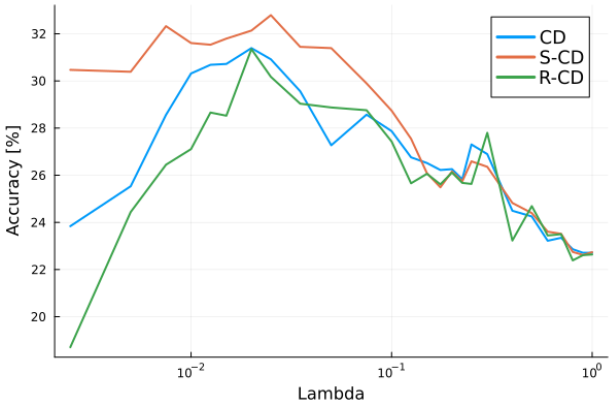


Comparaison de la précision des différents modèles de descente de coordonnées en fonction du paramètre λ.

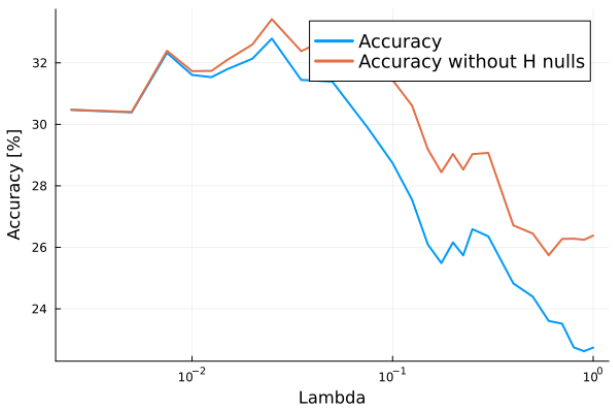


Comparaison de la précision et de la précision sans colonnes nulles de H en fonction du paramètre λ sur le modèle de descente de coordonnées S-CD.

Comparaison des différents modèles de descente de coordonnées et évolution de la précision en fonction du paramètre λ sur le modèle λ-ℓ_1-NMF sur le Jeu de Données *ng3sim* [31].

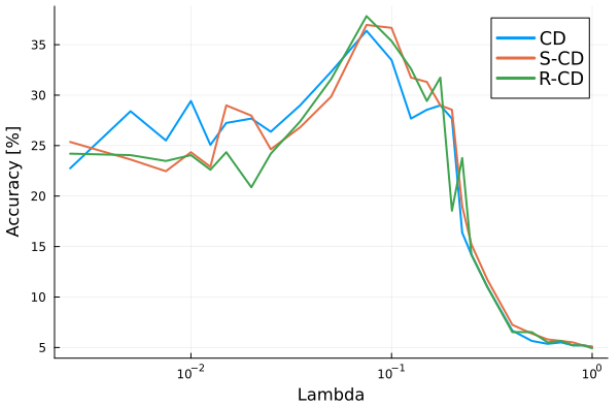


Comparaison de la précision des différents modèles de descente de coordonnées en fonction du paramètre λ.

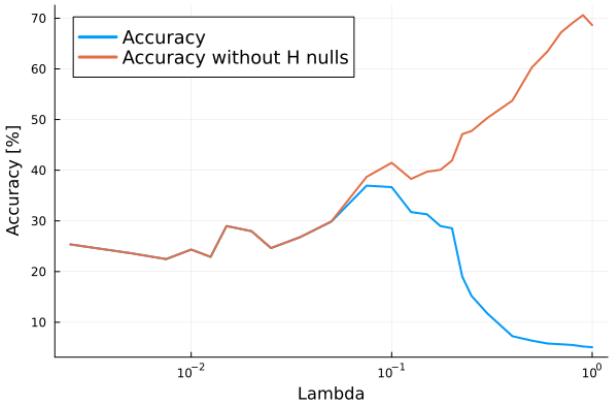


Comparaison de la précision et de la précision sans colonnes nulles de H en fonction du paramètre λ sur le modèle de descente de coordonnées S-CD.

Comparaison des différents modèles de descente de coordonnées et évolution de la précision en fonction du paramètre λ sur le modèle λ-ℓ_1-NMF sur le Jeu de Données *ohscal* [31].

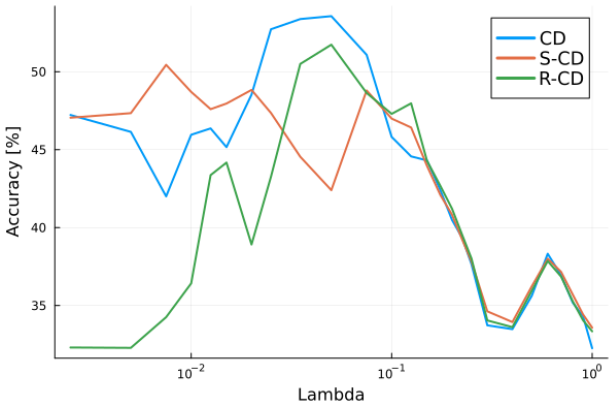


Comparaison de la précision des différents modèles de descente de coordonnées en fonction du paramètre λ.

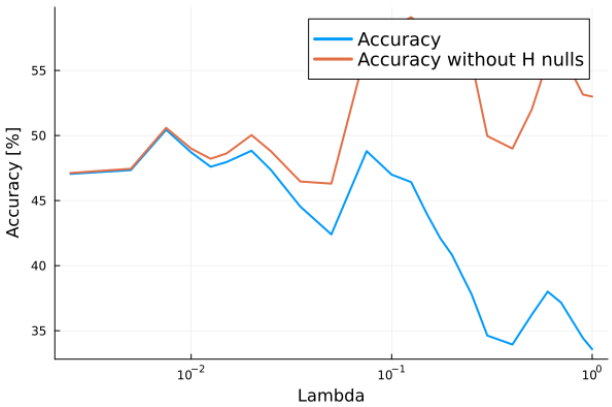


Comparaison de la précision et de la précision sans colonnes nulles de H en fonction du paramètre λ sur le modèle de descente de coordonnées S-CD.

Comparaison des différents modèles de descente de coordonnées et évolution de la précision en fonction du paramètre λ sur le modèle λ-ℓ_1-NMF sur le Jeu de Données *tr45* [31].



Comparaison de la précision des différents modèles de descente de coordonnées en fonction du paramètre λ.



Comparaison de la précision et de la précision sans colonnes nulles de H en fonction du paramètre λ sur le modèle de descente de coordonnées S-CD.

Comparaison des différents modèles de descente de coordonnées et évolution de la précision en fonction du paramètre λ sur le modèle λ-ℓ_1-NMF sur le Jeu de Données *sports* [31].

Annexe D : Résultats de l'algorithme λ_q-ℓ_1-NMF sur le jeu de données TDT2 avec $\alpha = 0.1$

```
MOTS DU THÈME 1     - Taille = 217
1 : 30              - 1.0
2 : team            - 0.856261
3 : voa             - 0.676966
4 : evidence        - 0.619192
5 : control         - 0.617687
6 : five            - 0.590890
7 : free            - 0.584682
8 : saying          - 0.517011
9 : saddam          - 0.495640
10: iraqi           - 0.492439

MOTS DU THÈME 2     - Taille = 279
1 : jury            - 1.0
2 : efforts         - 0.992474
3 : line            - 0.945485
4 : killed          - 0.924912
5 : saturday        - 0.878218
6 : times           - 0.867495
7 : win             - 0.857717
8 : statement       - 0.838084
9 : matter          - 0.805748
10: 20              - 0.734898

MOTS DU THÈME 3     - Taille = 209
1 : voa             - 1.0
2 : market          - 0.996132
3 : talks           - 0.953393
4 : night           - 0.947610
5 : 20              - 0.940757
6 : agreement       - 0.938026
7 : business        - 0.884788
8 : force           - 0.871504
9 : close           - 0.866419
10: five            - 0.759805

MOTS DU THÈME 4     - Taille = 250
1 : evidence        - 1.0
2 : currency        - 0.949502
3 : third           - 0.870333
4 : americans       - 0.667686
5 : sites           - 0.627920
6 : held            - 0.600261
7 : coming          - 0.476057
8 : doing           - 0.469401
9 : china           - 0.459824
10: 1996            - 0.456886

MOTS DU THÈME 5     - Taille = 226
1 : reporters       - 1.0
2 : agreed          - 0.877263
3 : able            - 0.855661
4 : senior          - 0.764329
5 : fund            - 0.693691
6 : mass            - 0.622900
7 : matter          - 0.590404
8 : led             - 0.531505
9 : city            - 0.487157
10: third           - 0.408433

MOTS DU THÈME 6     - Taille = 267
1 : outside         - 1.0
2 : cnn             - 0.746733
3 : press           - 0.545470
4 : investigation   - 0.537737
5 : talk            - 0.534475
6 : jury            - 0.534084
7 : meet            - 0.510922
8 : reported        - 0.486586
9 : matter          - 0.461567
10: currency        - 0.418642

MOTS DU THÈME 7     - Taille = 198
1 : story           - 1.0
2 : found           - 0.879993
3 : led             - 0.845065
4 : hope            - 0.672367
5 : clintons        - 0.651873
6 : plan            - 0.592067
7 : half            - 0.559257
8 : follows         - 0.548269
9 : current         - 0.475699
10: job             - 0.474326

MOTS DU THÈME 8     - Taille = 137
1 : international   - 1.0
2 : power           - 0.479077
3 : report          - 0.413618
4 : war             - 0.205826
5 : night           - 0.186504
6 : visit           - 0.163646
7 : bill            - 0.138038
8 : expected        - 0.134829
9 : close           - 0.126222
10: city            - 0.124659

MOTS DU THÈME 9     - Taille = 215
1 : 30              - 1.0
2 : reporters       - 0.914053
3 : americans       - 0.690371
4 : efforts         - 0.632358
5 : able            - 0.401724
6 : senior          - 0.393209
7 : hope            - 0.384555
8 : won             - 0.379978
9 : thats           - 0.362818
10: clintons        - 0.346550

MOTS DU THÈME 10    - Taille = 120
1 : defense         - 1.0
2 : talks           - 0.870547
3 : south           - 0.850040
4 : nations         - 0.724044
5 : asia            - 0.663788
6 : major           - 0.636644
7 : saddam          - 0.503792
8 : power           - 0.476227
9 : war             - 0.469620
```

```
10: bill            - 0.467561

MOTS DU THÈME 11    - Taille = 254
1 : bank            - 1.0
2 : little          - 0.967015
3 : economic        - 0.744379
4 : washington      - 0.707931
5 : clinton         - 0.676773
6 : added           - 0.642142
7 : continue        - 0.639149
8 : independent     - 0.636712
9 : law             - 0.629367
10: led             - 0.621450

MOTS DU THÈME 12    - Taille = 194
1 : reporters       - 1.0
2 : spokesman       - 0.635909
3 : six             - 0.593162
4 : attack          - 0.525049
5 : press           - 0.452950
6 : final           - 0.442310
7 : current         - 0.413916
8 : president       - 0.335605
9 : seen            - 0.315515
10: won             - 0.312924

MOTS DU THÈME 13    - Taille = 256
1 : reporters       - 1.0
2 : un              - 0.880025
3 : leaders         - 0.664175
4 : system          - 0.662797
5 : economy         - 0.590352
6 : iraq            - 0.576449
7 : iraqi           - 0.474768
8 : past            - 0.456340
9 : situation       - 0.452782
10: security        - 0.437444

MOTS DU THÈME 14    - Taille = 210
1 : iraq            - 1.0
2 : chief           - 0.443670
3 : leader          - 0.373932
4 : weapons         - 0.364560
5 : allow           - 0.300960
6 : called          - 0.273469
7 : agreement       - 0.256140
8 : military        - 0.253322
9 : added           - 0.251159
10: united          - 0.232097

MOTS DU THÈME 15    - Taille = 259
1 : security        - 1.0
2 : saddam          - 0.554366
3 : allow           - 0.508854
4 : eight           - 0.479011
5 : own             - 0.475400
6 : sites           - 0.411066
7 : jury            - 0.383053
8 : city            - 0.364198
9 : strong          - 0.346717
10: iraqs           - 0.340508

MOTS DU THÈME 16    - Taille = 240
1 : late            - 1.0
2 : theres          - 0.854239
3 : independent     - 0.598284
4 : taking          - 0.437898
5 : presidential    - 0.275513
6 : capital         - 0.235447
7 : move            - 0.234992
8 : win             - 0.205618
9 : found           - 0.204327
10: nations         - 0.199047

MOTS DU THÈME 17    - Taille = 218
1 : report          - 1.0
2 : seen            - 0.774838
3 : fund            - 0.749244
4 : late            - 0.665663
5 : earlier         - 0.623677
6 : cnn             - 0.573497
7 : saturday        - 0.530646
8 : investigation   - 0.528516
9 : agreed          - 0.457302
10: 10              - 0.437280

MOTS DU THÈME 18    - Taille = 268
1 : indonesia       - 1.0
2 : thats           - 0.276036
3 : agreement       - 0.264691
4 : run             - 0.246315
5 : eight           - 0.240807
6 : agreed          - 0.239534
7 : current         - 0.219254
8 : strong          - 0.202457
9 : recent          - 0.192247
10: reporters       - 0.168180

MOTS DU THÈME 19    - Taille = 203
1 : head            - 1.0
2 : economic        - 0.808394
3 : leader          - 0.719852
4 : crisis          - 0.630602
5 : story           - 0.628555
6 : percent         - 0.585947
7 : countries       - 0.545247
8 : headline        - 0.535892
9 : believe         - 0.531272
10: month           - 0.514539

MOTS DU THÈME 20    - Taille = 99
1 : people          - 1.0
2 : clinton         - 0.822120
3 : expected        - 0.703182
4 : team            - 0.573163
5 : weapons         - 0.558294
6 : told            - 0.547811
7 : un              - 0.509188
8 : i'm             - 0.498642
9 : times           - 0.490088
10: president       - 0.485162

Proportion de colonnes de H
non-nulles 99.39%
Proportion de colonnes de WtX
non-nulles 100.0%
```

Annexe E : Récapitulatif de l'ensemble des résultats sur les modèles ℓ_1-NMF, λ-ℓ_1-NMF et λ_q-ℓ_1-NMF, ainsi que les variantes de descente de coordonnées et d'accélération sur les jeux de données provenant de [31]

REINSTANCE = FALSE

dataset	r	m	n	Sx	Acc	H0	W0	Acc0	WtX0	Swh	Time	LastError	Maxiter
							CD						
NG20	20	43586	19949	99.819	9.1	67.53	90.0	13.47	99.72	99.989	69.88	0.0143694	9
ng3sim	3	15810	2998	99.291	17.38	48.0	100.0	36.21	99.6	99.964	3.51	0.0563389	11
classic	4	41681	7094	99.924	5.77	5.77	25.0	100.0	13.6	100.0	1.98	0.00692	4
ohscal	10	11465	11162	99.473	22.71	86.19	90.0	26.35	92.31	99.983	5.19	0.0330449	4
k1b	6	21819	2340	99.407	68.42	96.37	100.0	71.0	100.0	99.953	5.23	0.0498613	9
hitech	6	10080	2301	98.571	41.81	81.83	100.0	51.09	98.52	99.906	3.56	0.1021313	8
reviews	5	18483	4069	98.991	58.76	74.2	100.0	79.2	100.0	99.918	4.6	0.0699243	6
sports	7	14870	8580	99.144	32.26	64.42	100.0	50.08	99.91	99.875	14.29	0.0468873	10
tr11	9	6424	414	95.693	5.07	10.14	100.0	50.0	100.0	97.32	16.86	0.4316042	38
tr23	6	5831	204	93.409	9.31	11.76	100.0	79.17	100.0	93.363	4.06	1.1095599	16
tr41	10	7453	878	97.392	29.04	79.5	100.0	36.53	100.0	99.375	6.22	0.2134878	14
tr45	10	8261	690	96.603	5.07	7.25	100.0	70.0	100.0	98.104	15.98	0.3267829	22

REINSTANCE = TRUE

dataset	r	m	n	Sx	Acc	H0	Acc0	Swh	Time	LastError	Maxiter
							CD				
NG20	20	43586	19949	99.819	11.46	72.39	15.83	99.989	540.27	0.0143323	9
ng3sim	3	15810	2998	99.291	17.38	48.0	36.21	99.964	6.89	0.0563389	11
classic	4	41681	7094	99.924	15.99	28.31	56.47	99.999	13.51	0.0068529	4
ohscal	10	11465	11162	99.473	23.35	88.8	26.29	99.981	8.69	0.0329734	4
k1b	6	21819	2340	99.407	68.42	96.37	71.0	99.953	7.69	0.0498613	9
hitech	6	10080	2301	98.571	41.81	81.83	51.09	99.906	4.54	0.1021313	8
reviews	5	18483	4069	98.991	58.76	74.2	79.2	99.918	7.04	0.0699243	6
sports	7	14870	8580	99.144	32.26	64.42	50.08	99.875	22.39	0.0468873	10
tr11	9	6424	414	95.693	5.07	10.14	50.0	97.32	17.54	0.4316042	38
tr23	6	5831	204	93.409	9.31	11.76	79.17	93.363	4.14	1.1095599	16
tr41	10	7453	878	97.392	29.04	79.5	36.53	99.375	6.93	0.2134878	14
tr45	10	8261	690	96.603	5.07	7.25	70.0	98.104	17.23	0.3267829	22
							acc-CD				
NG20	20	43586	19949	99.819	10.13	77.51	13.07	99.987	573.91	0.014298	8
ng3sim	3	15810	2998	99.291	17.38	48.0	36.21	99.964	11.63	0.0563397	17
classic	4	41681	7094	99.924	15.99	28.31	56.47	99.999	14.02	0.0068529	4
ohscal	10	11465	11162	99.473	23.36	88.8	26.31	99.981	10.9	0.0329753	4
k1b	6	21819	2340	99.407	68.93	96.37	71.53	99.947	8.9	0.0498665	9
hitech	6	10080	2301	98.571	39.16	77.14	50.76	99.919	5.46	0.10237	8
reviews	5	18483	4069	98.991	58.76	74.2	79.2	99.918	8.78	0.0699245	6
sports	7	14870	8580	99.144	33.53	63.24	53.02	99.878	28.74	0.0468737	10
tr11	9	6424	414	95.693	4.59	10.14	45.24	97.162	13.24	0.4304049	17
tr23	6	5831	204	93.409	9.31	11.76	79.17	93.07	14.3	1.0919639	20
tr41	10	7453	878	97.392	31.78	78.82	40.32	99.429	14.75	0.2111625	20
tr45	10	8261	690	96.603	5.22	7.25	72.0	98.016	10.75	0.3251731	8
							R-CD				
NG20	20	43586	19949	99.819	12.86	72.27	17.79	99.989	738.68	0.0142797	11
ng3sim	3	15810	2998	99.291	17.21	48.0	35.86	99.969	9.65	0.0567303	15
classic	4	41681	7094	99.924	15.99	28.31	56.47	99.999	13.6	0.0068529	4
ohscal	10	11465	11162	99.473	21.94	87.97	24.94	99.98	9.47	0.0329729	4
k1b	6	21819	2340	99.407	67.69	96.37	70.24	99.95	7.77	0.049861	9
hitech	6	10080	2301	98.571	40.94	79.49	51.5	99.916	4.68	0.1022791	8
reviews	5	18483	4069	98.991	59.3	75.13	78.93	99.924	7.13	0.0699196	6
sports	7	14870	8580	99.144	33.55	63.38	52.94	99.878	24.76	0.0468748	11
tr11	9	6424	414	95.693	6.52	16.18	40.3	96.784	7.08	0.440422	16
tr23	6	5831	204	93.409	4.9	11.76	41.67	93.362	16.8	1.1101515	58
tr41	10	7453	878	97.392	33.6	74.49	45.11	99.338	10.26	0.2096898	20
tr45	10	8261	690	96.603	5.07	7.39	68.63	97.804	10.44	0.3299612	14

---------- acc-R-CD ----------

NG20	20	43586	19949	99.819	10.21	75.89	13.46	99.988	572.09	0.014297	8
ng3sim	3	15810	2998	99.291	17.04	48.0	35.51	99.966	4.11	0.056747	6
classic	4	41681	7094	99.924	15.99	28.31	56.47	99.999	14.02	0.0068529	4
ohscal	10	11465	11162	99.473	23.0	88.8	25.9	99.979	10.86	0.0329748	4
k1b	6	21819	2340	99.407	68.59	96.37	71.18	99.948	9.04	0.049868	9
hitech	6	10080	2301	98.571	37.9	74.32	50.99	99.927	5.61	0.1024848	8
reviews	5	18483	4069	98.991	58.69	74.2	79.1	99.918	8.93	0.0699241	6
sports	7	14870	8580	99.144	33.43	63.23	52.87	99.878	29.18	0.0468737	10
tr11	9	6424	414	95.693	4.59	9.9	46.34	97.036	14.82	0.4300873	19
tr23	6	5831	204	93.409	8.82	11.76	75.0	93.102	9.46	1.0954865	11
tr41	10	7453	878	97.392	34.28	75.4	45.47	99.192	7.3	0.2088852	10
tr45	10	8261	690	96.603	5.22	7.25	72.0	97.823	14.06	0.3252675	9

---------- S-CD ----------

NG20	20	43586	19949	99.819	12.38	75.28	16.44	99.989	805.99	0.0142979	11
ng3sim	3	15810	2998	99.291	17.11	48.0	35.65	99.966	9.13	0.0567529	14
classic	4	41681	7094	99.924	15.99	28.31	56.47	99.999	13.61	0.0068529	4
ohscal	10	11465	11162	99.473	22.45	87.97	25.52	99.981	8.91	0.0329739	4
k1b	6	21819	2340	99.407	68.25	96.37	70.82	99.952	8.19	0.049862	9
hitech	6	10080	2301	98.571	40.5	80.1	50.57	99.903	5.45	0.1022108	9
reviews	5	18483	4069	98.991	59.65	75.42	79.08	99.925	5.95	0.0699234	5
sports	7	14870	8580	99.144	33.58	63.36	53.0	99.877	20.54	0.0468728	9
tr11	9	6424	414	95.693	4.83	10.63	45.45	97.026	11.57	0.4356649	25
tr23	6	5831	204	93.409	5.88	11.76	50.0	92.981	5.12	1.0863689	18
tr41	10	7453	878	97.392	33.94	75.28	45.08	99.332	7.95	0.2087436	15
tr45	10	8261	690	96.603	5.07	7.39	68.63	98.387	15.43	0.3299963	20

---------- acc-S-CD ----------

NG20	20	43586	19949	99.819	10.44	75.87	13.76	99.988	740.02	0.0143032	10
ng3sim	3	15810	2998	99.291	17.38	48.0	36.21	99.964	12.83	0.056339	18
classic	4	41681	7094	99.924	15.99	28.31	56.47	99.999	14.4	0.0068529	4
ohscal	10	11465	11162	99.473	23.29	88.8	26.23	99.981	11.44	0.0329753	4
k1b	6	21819	2340	99.407	68.93	96.37	71.53	99.947	9.42	0.0498649	9
hitech	6	10080	2301	98.571	41.76	80.62	51.81	99.92	7.17	0.1023078	10
reviews	5	18483	4069	98.991	58.69	74.2	79.1	99.918	9.21	0.0699241	6
sports	7	14870	8580	99.144	33.46	63.19	52.95	99.878	29.68	0.0468735	10
tr11	9	6424	414	95.693	4.59	9.9	46.34	97.037	12.95	0.4304279	16
tr23	6	5831	204	93.409	8.82	11.76	75.0	92.947	17.59	1.0849798	21
tr41	10	7453	878	97.392	34.17	75.17	45.45	99.42	8.16	0.2084387	11
tr45	10	8261	690	96.603	5.22	7.25	72.0	98.194	11.67	0.3252374	7

---------- C-CD ----------

NG20	20	43586	19949	99.819	10.77	68.97	15.62	99.989	3930	0.014325	11
ng3sim	3	15810	2998	99.291	17.08	48.0	35.58	99.966	20.32	0.0567637	8
classic	4	41681	7094	99.924	15.99	28.31	56.47	99.999	64.73	0.0068529	4
ohscal	10	11465	11162	99.473	23.09	89.34	25.84	99.979	102.19	0.0329219	5
k1b	6	21819	2340	99.407	67.69	96.37	70.24	99.952	41.04	0.0498594	9
hitech	6	10080	2301	98.571	47.5	85.7	55.43	99.906	19.37	0.1020512	8
reviews	5	18483	4069	98.991	58.96	74.54	79.1	99.919	49.92	0.0699192	7
sports	7	14870	8580	99.144	33.68	66.75	50.46	99.872	173.34	0.0468888	12
tr11	9	6424	414	95.693	4.83	14.98	32.26	97.254	10.95	0.4369997	16
tr23	6	5831	204	93.409	3.92	12.25	32.0	92.57	10.54	1.0493853	30
tr41	10	7453	878	97.392	35.54	82.69	42.98	99.122	12.69	0.214273	11
tr45	10	8261	690	96.603	5.22	7.25	72.0	98.28	38.44	0.3239622	28

---------- acc-C-CD ----------

NG20	20	43586	19949	99.819	10.23	75.89	13.48	99.989	5208	0.0142912	14
ng3sim	3	15810	2998	99.291	17.38	48.0	36.21	99.964	49.91	0.0563387	18
classic	4	41681	7094	99.924	15.99	28.31	56.47	99.999	65.97	0.0068529	4
ohscal	10	11465	11162	99.473	23.08	89.34	25.83	99.979	109.03	0.0329319	5
k1b	6	21819	2340	99.407	68.5	96.37	71.09	99.947	43.3	0.0498644	9
hitech	6	10080	2301	98.571	37.59	74.32	50.58	99.922	24.29	0.1024805	9
reviews	5	18483	4069	98.991	58.76	74.2	79.2	99.918	44.77	0.0699244	6
sports	7	14870	8580	99.144	33.47	63.18	52.98	99.877	152.53	0.046874	10
tr11	9	6424	414	95.693	4.59	9.9	46.34	97.139	19.63	0.4301992	18
tr23	6	5831	204	93.409	8.82	11.76	75.0	92.83	22.0	1.0853116	18
tr41	10	7453	878	97.392	33.94	75.17	45.15	99.367	13.72	0.2088956	10
tr45	10	8261	690	96.603	5.07	7.25	70.0	98.117	15.94	0.3251649	7

Lambda = sqrt(1-Sx)/2

dataset	r	m	n	Sx	lambda	Acc	H0	Acc0	WtX0	Swh	Time	LastError	Maxiter
						S-CD							
NG20	20	43586	19949	99.819	0.0212	16.34	84.02	19.45	99.98	95.444	971.47	0.0117832	16
ng3sim	3	15810	2998	99.291	0.0420	23.08	59.67	38.68	99.93	93.883	14.68	0.0493692	29
classic	4	41681	7094	99.924	0.0137	41.51	53.33	77.85	95.74	98.714	11.05	0.0054	12
ohscal	10	11465	11162	99.473	0.0362	31.66	96.98	32.65	100.0	95.125	59.3	0.0285594	23
k1b	6	21819	2340	99.407	0.0385	67.48	91.37	73.85	100.0	95.276	14.24	0.040664	16
hitech	6	10080	2301	98.571	0.0597	39.29	88.96	44.16	100.0	90.786	22.71	0.0874634	26
reviews	5	18483	4069	98.991	0.0502	56.03	81.54	68.72	100.0	93.424	60.83	0.0583139	38
sports	7	14870	8580	99.144	0.0462	42.67	92.73	46.02	100.0	94.057	69.69	0.0380982	23
tr11	9	6424	414	95.693	0.1037	40.1	92.27	43.46	100.0	41.807	4.2	0.3823167	8
tr23	6	5831	204	93.409	0.1283	23.04	67.16	34.31	100.0	48.107	12.68	0.9994549	37
tr41	10	7453	878	97.392	0.0807	32.23	96.81	33.29	100.0	72.252	16.15	0.1781941	21
tr45	10	8261	690	96.603	0.0921	32.17	91.3	35.24	100.0	44.732	67.6	0.2887577	66

Lambda Decrising

dataset	r	m	n	Sx	lambda	Acc	H0	W0	Acc0	WtX0	Swh	Time	LastError	Maxiter
							S-CD							
NG20	20	43586	19949	99.819	0.014	19.08	91.53	100.0	20.84	99.99	91.714	1596	0.0110415	25
ng3sim	3	15810	2998	99.291	0.0084	35.02	92.03	100.0	38.06	99.97	72.257	149.75	0.0394162	50
classic	4	41681	7094	99.924	0.0011	51.13	91.51	100.0	55.87	99.89	83.772	115.96	0.0035865	14
ohscal	10	11465	11162	99.473	0.5	24.4	92.24	100.0	26.45	97.89	99.936	8.93	0.0328528	6
k1b	6	21819	2340	99.407	0.5	66.41	90.77	100.0	73.16	100.0	99.874	6.21	0.0494147	10
hitech	6	10080	2301	98.571	0.039	39.03	92.05	100.0	42.4	100.0	86.316	127.28	0.0817223	38
reviews	5	18483	4069	98.991	0.023	60.7	90.83	100.0	66.83	100.0	86.508	189.86	0.0512866	37
sports	7	14870	8580	99.144	0.039	44.3	94.98	100.0	46.64	100.0	93.143	295.73	0.037118	22
tr11	9	6424	414	95.693	0.108	39.13	95.17	100.0	41.12	100.0	43.722	47.89	0.3795012	33
tr23	6	5831	204	93.409	0.039	36.27	97.55	100.0	37.19	100.0	13.561	44.25	0.9324865	33
tr41	10	7453	878	97.392	0.108	38.27	95.44	100.0	40.1	100.0	73.245	104.9	0.1877006	98
tr45	10	8261	690	96.603	0.065	34.78	98.12	100.0	35.45	100.0	34.219	162.83	0.2802145	84

Lambda Col

ALPHA = 0.25

dataset	r	m	n	Sx	Acc	H0	W0	Acc0	WtX0	Swh	Time	LastError	Maxiter
							column_l1_nmf						
NG20	20	43586	19949	99.819	10.99	87.03	100.0	12.63	99.64	99.942	53.88	0.0143363	6
ng3sim	3	15810	2998	99.291	29.29	84.42	100.0	34.69	99.43	99.844	2.14	0.0565197	6
classic	4	41681	7094	99.924	18.79	30.98	75.0	60.65	48.65	99.996	2.61	0.006871	4
ohscal	10	11465	11162	99.473	27.25	94.33	100.0	28.89	99.57	99.74	27.43	0.0326357	15
k1b	6	21819	2340	99.407	64.83	93.29	100.0	69.49	100.0	99.624	11.08	0.0486964	15
hitech	6	10080	2301	98.571	48.59	96.09	100.0	50.57	100.0	98.969	36.53	0.0999799	58
reviews	5	18483	4069	98.991	73.29	86.63	100.0	84.6	100.0	99.391	13.94	0.0679874	14
sports	7	14870	8580	99.144	44.11	93.6	100.0	47.13	99.99	99.187	27.2	0.0453621	14
tr11	9	6424	414	95.693	39.86	99.52	100.0	40.05	100.0	73.584	7.82	0.4264687	15
tr23	6	5831	204	93.409	42.65	95.1	100.0	44.85	100.0	50.194	5.43	1.1134564	18
tr41	10	7453	878	97.392	47.15	99.77	100.0	47.26	100.0	95.001	12.24	0.2071058	20
tr45	10	8261	690	96.603	30.58	77.1	100.0	39.66	100.0	80.532	10.75	0.3250274	13

ALPHA = 0.1

dataset	r	m	n	Sx	Acc	H0	W0	Acc0	WtX0	Swh	Time	LastError	Maxiter
							column_l1_nmf						
NG20	20	43586	19949	99.819	17.27	96.43	100.0	17.91	99.83	99.68	689.34	0.0140814	13
ng3sim	3	15810	2998	99.291	32.79	90.29	100.0	36.31	99.67	98.974	5.52	0.0554739	13
classic	4	41681	7094	99.924	32.62	66.45	100.0	49.09	77.39	99.945	4.89	0.0067539	6
ohscal	10	11465	11162	99.473	27.57	98.58	100.0	27.97	100.0	98.593	49.53	0.0315021	22
k1b	6	21819	2340	99.407	69.83	99.36	100.0	70.28	100.0	98.273	19.61	0.045902	22
hitech	6	10080	2301	98.571	46.76	99.78	100.0	46.86	100.0	95.167	32.42	0.094157	41
reviews	5	18483	4069	98.991	74.15	99.71	100.0	74.37	100.0	97.192	31.28	0.0639603	23
sports	7	14870	8580	99.144	56.85	98.95	100.0	57.46	99.99	97.196	86.0	0.0423597	33
tr11	9	6424	414	95.693	28.99	100.0	100.0	28.99	100.0	51.585	18.77	0.3902837	31
tr23	6	5831	204	93.409	41.18	100.0	100.0	41.18	100.0	21.041	2.17	1.0508458	7
tr41	10	7453	878	97.392	56.15	100.0	100.0	56.15	100.0	81.468	72.33	0.1897164	94
tr45	10	8261	690	96.603	33.19	100.0	100.0	33.19	100.0	54.141	7.86	0.3024126	8