Doria Hammadi
Ortholand Juliette

Rapport TME 4

Filtrage collaboratif

I. Introduction

Dans ce TME nous allons étudier les méthodes de filtrages collaboratifs basés sur la factorisation matricielle. Nous fonderons notre étude sur une base de données contenant des notes de films liées à des utilisateurs. Le but sera d'utiliser la similarité entre les utilisateurs pour réaliser des prédictions. L'étude comparera différentes méthodes.

Dans un premier temps nous présenterons la base de données et les méthodes utilisées. Puis nous étudierons des méthodes de prédiction basées sur la réduction de dimension (SVD et NMF). Enfin nous étudierons différentes version de prédiction basée sur une descente de gradient stochastique.

II. Méthode

a. Base de donnée

Nous avons utilisés le jeu de données MovieLens qui comporte différents jeux de données avec des tailles différentes. Celui que nous avons utilisé ci-dessous comprend 100 000 notes de 943 utilisateurs sur une échelle de 1 à 5, pour 1682 films.

Dans cet ensemble de données, chaque utilisateur a attribué une note à au moins 20 films, ce qui signifie que 6,3% des évaluations film-utilisateur ont une valeur.

Pour ne pas représenter les valeurs manquantes comme étant 0 (valeurs nulles) nous avons utilisé des matrices sparses qui représentent juste les valeurs non vides.

b. Découpage train test

Afin de nous permettre d'évaluer les différentes méthodes de prédiction nous avons découpé le set de données en deux : un training set et un testing set.

Pour construire le testing set nous avons dans un premier temps parcouru les utilisateurs et sélectionné un film au hasard parmi les films notés. Nous avons ensuite sélectionné pour chaque film une note donnée par un utilisateur au hasard. Toutes les informations extraites de la base de données afin de constituer le testing set ont été supprimées du training set.

Une fois constitué le testing set contenait 2 606 notes ce qui correspond à 2.6% de la base de données initiale.

c. Méthode d'évaluation

Pour nous permettre de mieux interpréter les évaluations des résultats obtenus nous avons réalisé des tests avec des algorithmes naïfs. Il est important de noter que les résultats de prédiction des algorithmes seront arrondit à l'entier le plus proche pour respecter le format de la base de données.

Nous avons dans un premier temps prédit les notes avec un entier aléatoire entre 1 et 5.

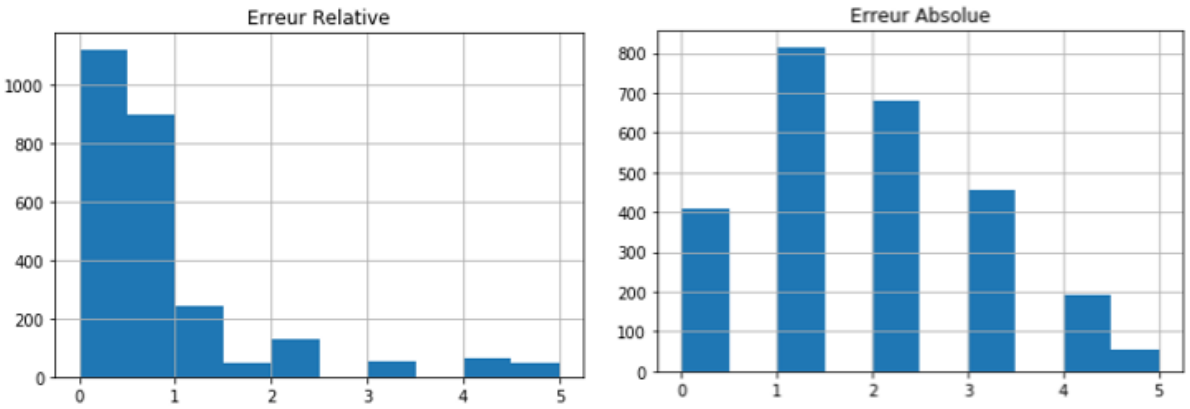


Figure 1: Répartition de l'erreur pour le tirage aléatoire

	Erreur Relative	Erreur Absolue
count	2606	2606
mean	0.761	1.755
std	0.972	1.239
min	0	0
25%	0.25	1
50%	0.5	2
75%	0.8	3
max	5	5

Figure 2: Statistiques de répartition de l'erreur en tirage aléatoire

On obtient bien une répartition normale de l'erreur absolue liée au tirage aléatoire.

Nous avons ensuite testé en remplaçant la note prédite par la moyenne de toutes les notes. Dans notre cas la moyenne était de 3.6 nous avons donc remplacé par 4.

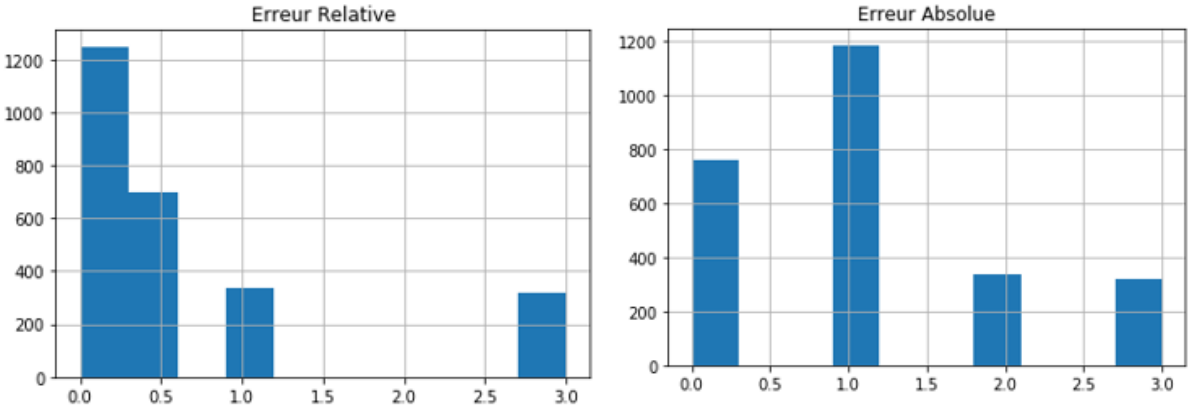


Figure 3: Répartition de l'erreur pour une prédiction constante

	Erreur Relative	Erreur Absolue
count	2606	2606
mean	0.626	1.085
std	0.940	0.953
min	0	0
25%	0	0
50%	0.333	1
75%	1	2
max	3	3

Figure 4: Statistiques de répartition de l'erreur pour une prédiction constante

On observe qu'en moyenne les résultats sont meilleurs avec une prédiction constante à la valeur moyenne qu'avec un tirage aléatoire.

III. Méthodes pré-implémentées

d. Prédiction SVD

i. Construction de l'algorithme

La décomposition en valeurs singulières SVD est une méthode de factorisation matricielle elle permet de décomposer une matrice M en trois autre matrice :

$$M = Q\Lambda P.$$

Cette méthode nous permet de rechercher les matrices Q et P reconstruisant au mieux la matrice des notes.

Pour construire l'algorithme nous avons utilisés la fonction **TruncatedSVD** de la bibliothèque sklearn avec 25 dimensions.

ii. Résultats

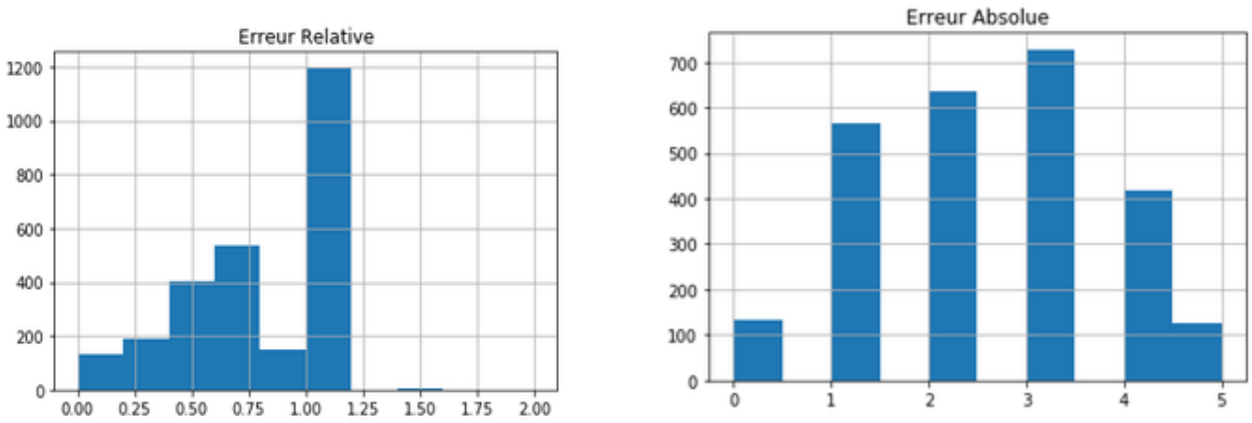


Figure 5 Répartition de l'erreur pour une prédiction SVD

	Erreur Relative	Erreur Absolue
count	2606	2606
mean	0,752	2,425
std	0,289	1,263
min	0,000	0,000
25%	0,600	1,000
50%	0,800	2,000
75%	1,000	3,000
max	2,000	5,000

Figure 6 : Statistiques de répartition de l'erreur pour une prédiction SVD

On peut remarquer que l'erreur relative et absolue sont moins bien que l'aléatoire, on a donc décidé de refaire les tests en remplacent cette fois les valeurs manquantes par la moyenne des notes, les résultats sont représenté ci-dessous :

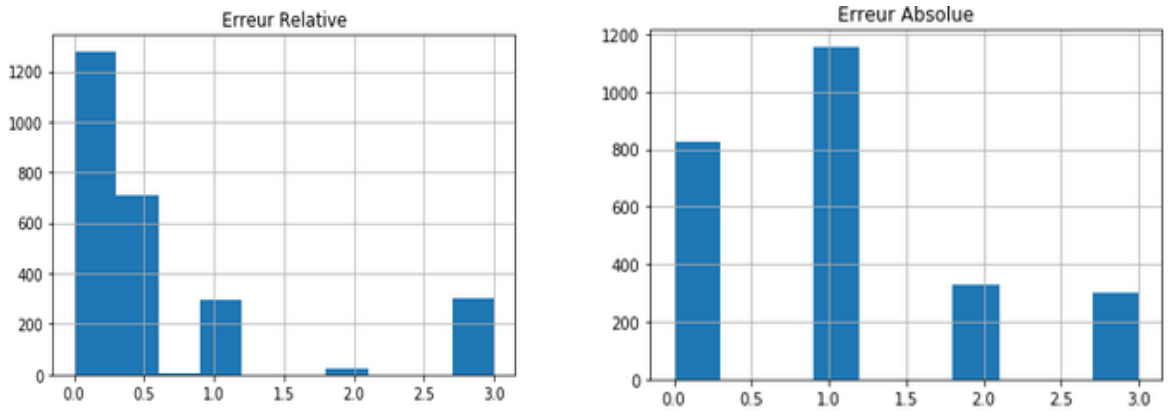


Figure 7 Répartition de l'erreur pour une prédiction SVD avec remplacements des valeurs manquantes

	Erreur Relative	Erreur Absolue
count	2606	2606
mean	0,603	1,035
std	0,930	0,951
min	0,000	0,000
25%	0,000	0,000
50%	0,333	1,000
75%	0,400	1,000
max	4,000	4,000

Figure 8 : Statistiques de répartition de l'erreur pour une prédiction SVD avec remplacement des valeurs manquantes

Après avoir remplacé les valeurs manquantes on remarque que les résultats obtenus sont meilleurs, ce remplacement nous a permis de diminuer l'erreur relative et absolue.

e. Prédiction NMF

iii. Construction de l'algorithme

La factorisation de matrice non négative (NMF) permet de décomposer la matrice d'origine M en deux matrices non négatives Q et P tel que : **M ≈ Q P.**

Pour construire l'algorithme nous avons utilisés la fonction **NMF** de la bibliothèque sklearn avec un espace de 25 dimensions.

iv. Résultats

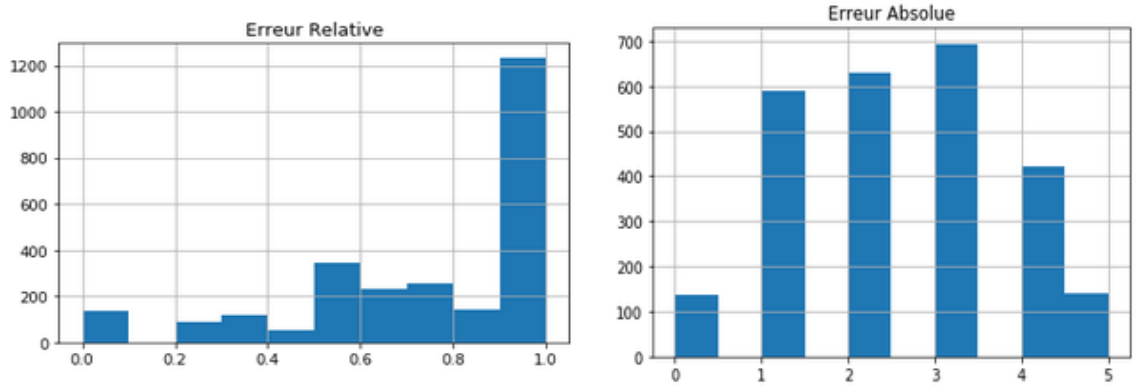


Figure 9 : Répartition de l'erreur pour une prédiction NMF

	Erreur Relative	Erreur Absolue
count	2606	2606
mean	0,751	2,418
std	0,294	1,285
min	0,000	0,000
25%	0,500	1,000
50%	0,800	2,000
75%	1,000	3,000
max	1,000	5,000

Figure 10: Statistiques de répartition de l'erreur pour une prédiction NMF

Apres remplacement des valeurs manquantes par la moyenne :

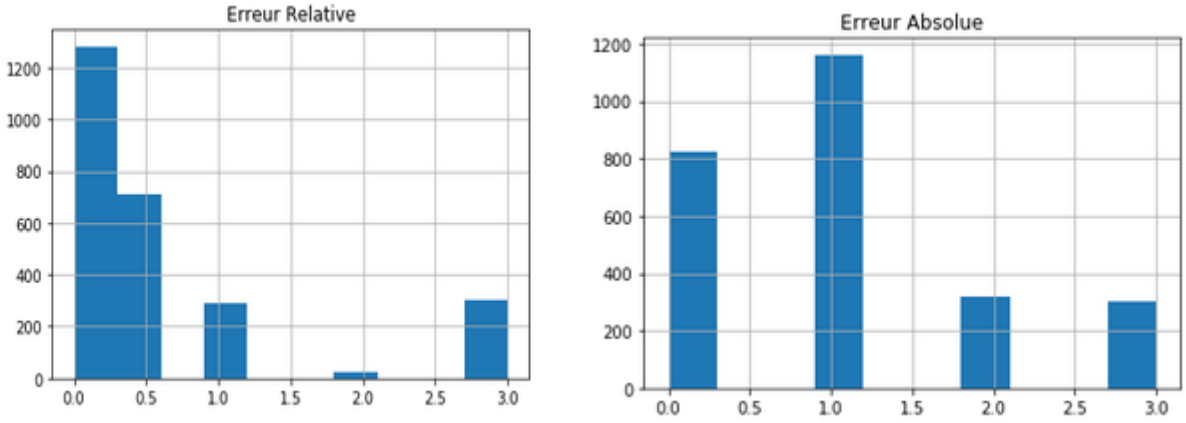


Figure 11 : Répartition de l'erreur pour une prédiction NMF avec remplacements des valeurs manquantes

	Erreur Relative	Erreur Absolue
count	2607	2607
mean	0,604	1,039
std	0,928	0,949
min	0,000	0,000
25%	0,000	0,000
50%	0,333	1,000
75%	0,400	1,000
max	3,000	3,000

Figure 12 : Statistiques de répartition de l'erreur pour une prédiction NMF avec remplacement des valeurs manquantes

On remarque que les résultats obtenues sont très proches des résultats obtenue par la méthode SVD.

f. Conclusion

Que ce soit la méthode SVD ou NMF les scores obtenus pour la base de données sont similaires au score de prédiction aléatoire : environs 0.75 d'erreur relative. Les deux méthodes ont des scores très similaires autant en répartition qu'en moyenne pour l'erreur relative et absolue. Cependant, si l'on remplace les valeurs manquantes par la moyenne des valeurs de la base de données, on réussit à faire légèrement mieux qu'avec la prédiction à valeur constante : 0.60 contre 0.63 d'erreur relative.

IV. Prédiction gradient

g. Algorithme avec pénalité L2

v. Construction de l'algorithme

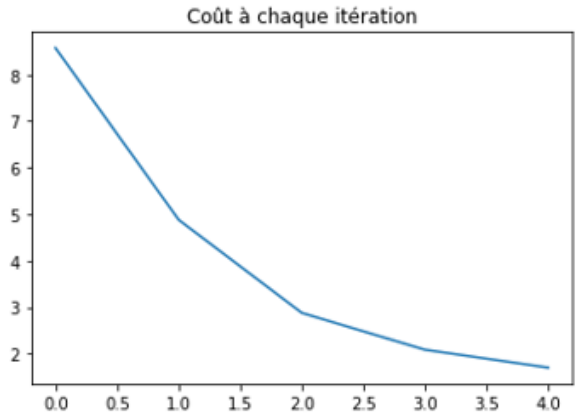
L'algorithme mis en place correspond à une descente de gradient stochastique avec pénalisation L2. Pour cela nous avons bouclé sur les notes non nulles contenues dans le training set et avons cherché à minimiser le coût :

$$J(y,q,p) = \left(y_{i,j} - (\langle q_i; p_j\rangle + \lambda\langle q_i; q_i\rangle + \lambda\langle p_j; p_j\rangle)\right)^2$$

vi. Résultats

Nous avons dans un premier temps cherché à tester sans prendre en compte la pénalisation L2. Les paramètres de tests étaient les suivants :

- Dimension : 2
- λ : 0
- ϵ : 10^{-3}
- Maximum d'itération : 5



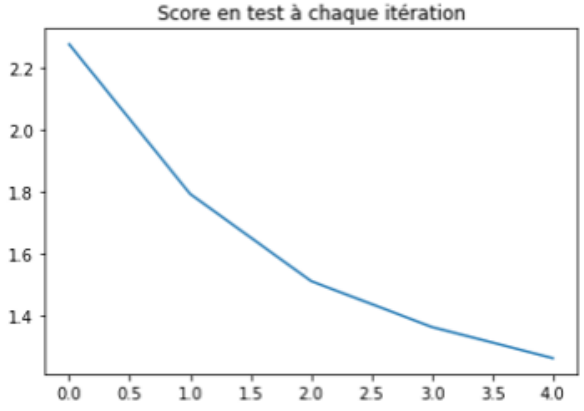


Figure 13: Validation du bon fonctionnement de l'algorithme sans pénalité L2

On obtient des courbes descendantes autant pour le coût que pour le test ce qui nous

permet de vérifier que l'algorithme a été codé correctement. On peut ensuite s'intéresser au résultats.

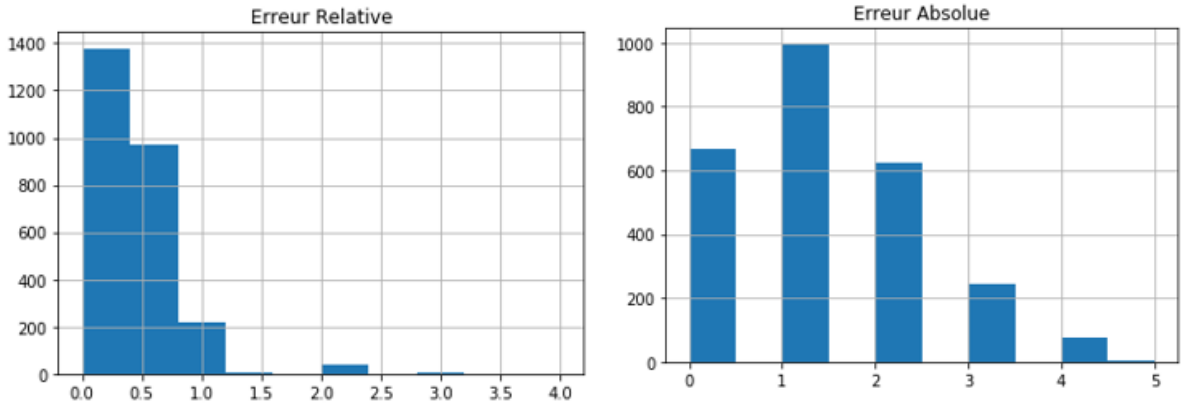


Figure 14: Répartition de l'erreur sans pénalité L2

	Erreur Relative	Erreur Absolue
count	2606	2606
mean	0.393	1.261
std	0.362	1.040
min	0	0
25%	0	0
50%	0.333	1
75%	0.5	2
max	4	5

Figure 15: Statistiques de répartition de l'erreur sans pénalité L2

On obtient en moyenne de bien meilleur résultats qu'avec les algorithmes utilisés précédemment. L'erreur absolue semble être répartie selon une loi normale autour de 1.

Dans un deuxième temps nous avons pris en compte la pénalité L2. Les paramètres de tests étaient les suivants :

- Dimension : 2
- λ : 0.001
- ϵ : 10^{-3}
- Maximum d'itération : 5

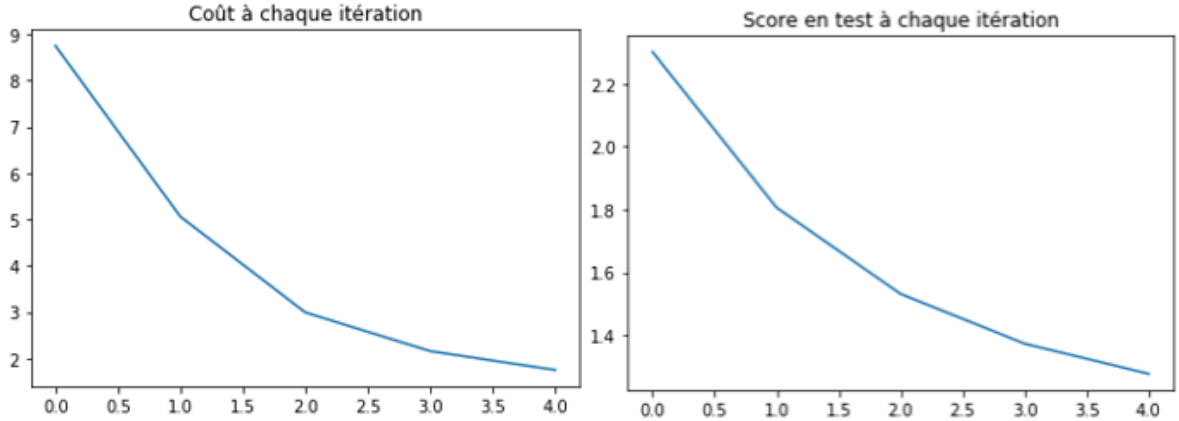


Figure 16 : Validation du bon fonctionnement de l'algorithme avec pénalité L2

Les courbes descendantes nous permettent de valider le bon fonctionnement de l'algorithme.

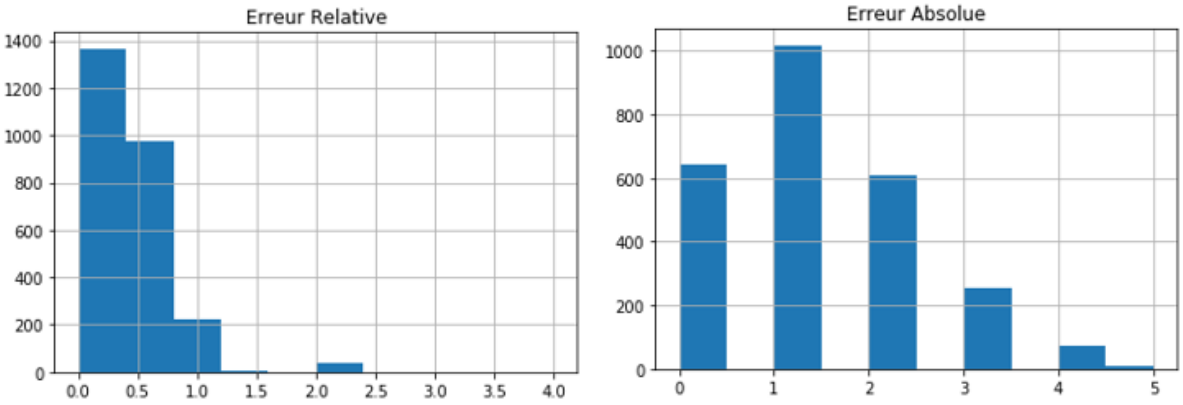


Figure 17: Répartition de l'erreur avec pénalité L2

	Erreur Relative	Erreur Absolue
count	2606	2606
mean	0.397	1.277
std	0.356	1.044
min	0	0
25%	0.2	1
50%	0.333	1
75%	0.6	2
max	4	5

Figure 18: Statistiques de répartition de l'erreur avec pénalité L2

Ajouter la pénalisation L2 permet de faire baisser l'erreur moyenne qu'elle soit absolue ou relative. La répartition de l'erreur est sensiblement la même avec et sans pénalisation L2.

vii. Conclusion

Ajouter la pénalisation L2 à l'algorithme de descente de gradient stochastique permet d'améliorer sensiblement les résultats mais ne change pas la répartition de l'erreur absolue. Une recherche du meilleur paramètre λ pourrait être conduite afin de minimiser l'erreur moyenne pour cette méthode.

h. Algorithme avec pénalité L2 et biais

viii. Construction de l'algorithme

Afin de tenir compte du biais dans l'algorithme de descente de gradient stochastique avec pénalité L2, nous avons modifié la fonction coût vue précédemment :

$$J(y,q,p) = \left(y_{i,j} - \left(\langle q_i; p_j\rangle + \mu + b_j^u + b_i^f + \lambda\left[\langle q_i; q_i\rangle + \langle p_j; p_j\rangle + \mu + b_j^u + b_i^f\right]\right)\right)^2$$

$$avec\ b^u\ le\ biais\ utilisateur\ et\ b^f\ le\ biais\ associé\ au\ films$$

ix. Résultats

Nous avons testé l'algorithme avec biais en utilisant aussi la pénalisation L2. Des tests ont aussi été réalisés sans pénalité L2 et sont présents en annexe. Les paramètres de tests étaient les suivants :

- Dimension : 2
- λ : 0.001
- ϵ : 10^{-3}
- Maximum d'itération : 5

Les résultats obtenus :

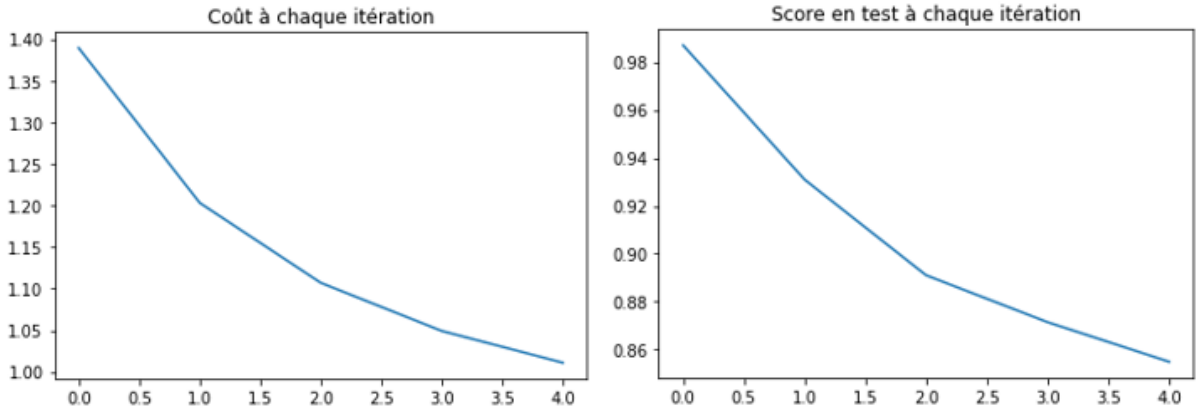


Figure 19: Validation du bon fonctionnement de l'algorithme avec biais et pénalisation L2

Les courbes descendantes nous permettent de valider le bon fonctionnement de l'algorithme.

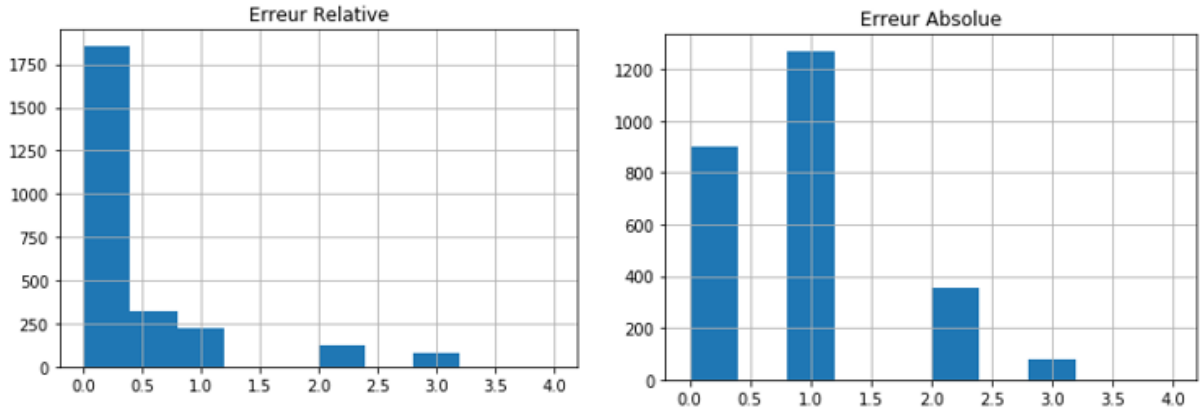


Figure 20: Répartition de l'erreur pour une prédiction avec biais et pénalité L2

	Erreur Relative	Erreur Absolue
count	2606	2606
mean	0.424	0.855
std	0.647	0.767
min	0	0
25%	0	0
50%	0.25	1
75%	0.4	1
max	4	4

Figure 21: Statistiques de répartition de l'erreur pour une prédiction avec biais et pénalité L2

La valeur moyenne de l'erreur relative est légèrement plus élevée pour l'algorithme avec biais par rapport à celui sans biais. Cependant l'erreur absolue passe de 1.277 à 0.885 avec la prise en compte des biais. De plus lorsque l'on analyse la répartition de l'erreur relative on observe que pour

la méthode avec biais le nombre d'erreurs comprises entre 0 et 0.25 est plus élevé que pour la méthode sans biais. On pourrait mener des analyses en augmentant le nombre d'itération afin de voir si l'erreur relative de la méthode avec biais devient inférieure à celle de la méthode sans biais. Là encore la répartition de l'erreur n'est pas modifiée.

i. Conclusion

On a pu observer que l'ajout de pénalité et/ou de biais permet d'améliorer les scores de la méthode de descente de gradient stochastique, que ce soit en erreur relative ou en erreur absolue. Quelle que soit la méthode utilisée la répartition de l'erreur reste sensiblement la même est suit une loi normal centrée en 1.

Pour ce qui est de l'ajout de biais dans la méthode avec pénalité L2, on obtient une erreur absolue plus faible 0.885 contre 1.246. Cependant l'erreur relative augmente sensiblement et passe de 0.397 à 0.424. Cette différence s'explique par la répartition de l'erreur relative. Bien qu'en nombre plus élevée entre 0 et 0.25, la quantité d'erreur pour des valeurs supérieures à 2 est plus importante.

V. Conclusion

Au cours de ce rapport nous avons pu étudier différentes méthodes de prédiction. Les méthodes utilisant des algorithmes de réduction de dimension, SVD et NMF, sans traitement on donnés des scores similaires à des résultats de prédiction aléatoire (0.75 d'erreur relative). Remplacé les valeurs manquantes par la moyenne de la base de données a permis d'améliorer très légèrement les scores (0.60 contre 0.62 d'erreur relative) par rapport à une prédiction constante.

Les algorithmes de descente de gradient, qu'en a eux, ont produit des résultats bien meilleurs (0.42 d'erreur relative en moyenne). L'ajout de pénalisation L2 à l'algorithme de descente stochastique a permis d'améliorer sensiblement les scores tout en gardant une distribution très similaire. La prise en compte du biais a quant à elle modifié la répartition de l'erreur (augmentation des erreurs extrêmes) et réduit l'erreur absolue (0.855 contre 1.277). L'erreur relative a cependant légèrement augmenté et passe de 0.397 à 0.424.

On peut conclure que les méthodes à descentes de gradient sont à privilégier, en choisissant suivant la sensibilité aux erreurs souhaité une descente de gradient stochastique avec pénalité L2 avec ou sans biais.

VI. Annexe 1 – Descente de gradient stochastique avec biais uniquement

Nous avons testé l'algorithme avec biais sans la pénalisation L2. Les paramètres de tests étaient les suivants :

- Dimension : 2
- λ : 0
- ϵ : 10^{-3}
- Maximum d'itération : 5

Les résultats obtenus :

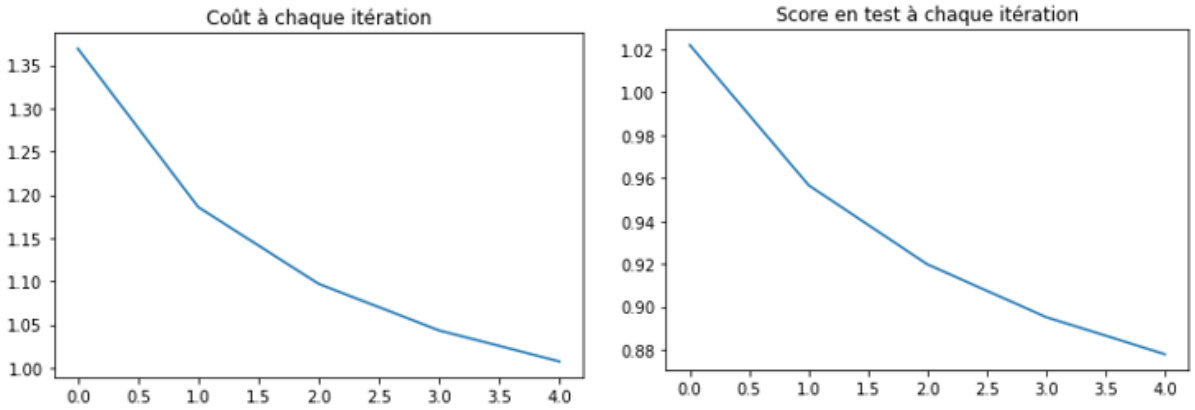


Figure 22: Validation du bon fonctionnement de l'algorithme avec biais et sans pénalisation L2

Les courbes descendantes nous permettent de valider le bon fonctionnement de l'algorithme.

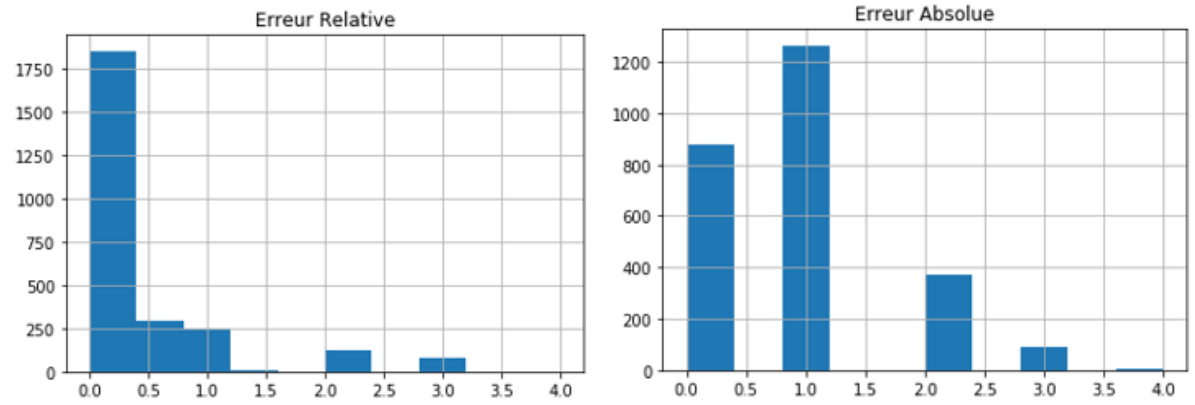


Figure 23: Répartition de l'erreur pour une prédiction avec biais et sans pénalité L2

	Erreur Relative	Erreur Absolue
count	2606	2606
mean	0.442	0.878
std	0.664	0.781
min	0	0
25%	0	0
50%	0.25	1
75%	0.5	1
max	4	4

Figure 24: Statistiques de répartition de l'erreur pour une prédiction avec biais et sans pénalité L2