Projet de statistique appliquée
Sujet 2
10 mai 2016

Collaborative Filtering

Biwei Cui
Claudia Delgado
Mehdi Miah
Ulrich Mpeli Mpeli

Tuteurs : Vincent Cottet, Mehdi Sebbar

Encadrant : Pierre Alquier

Table des matières

Rappel des notations utilisées

Pour un problème donné, nous avons les variables suivantes :

- N_U le nombre d'utilisateurs ;
- N_M le nombre de films ;
- N le nombre de notes ;
- $U = \{u_1, u_2, ..., u_{N_U}\}$ l'ensemble des utilisateurs ;
- $M = \{m_1, m_2, ..., m_{N_M}\}$ l'ensemble des films ;
- $Y = (y_{u,m})_{(u,m)\in U\times M}$ telle que : $y_{u,m}$ = note donnée par l'utilisateur u pour le film m ;
- $R = (r_{u,m})_{(u,m)\in U\times M}$ telles que : $r_{u,m} = \begin{cases} 1 & \text{si l'utilisateur } u \text{ a noté le film } m \\ 0 & \text{sinon} \end{cases}$
- $\Omega_0 = \{(u, m) \in U \times M | r_{u,m} = 0\}$, ensemble des couples (u, m) où l'utilisateur u n'a pas noté le film m ;
- $\Omega_1 = \{(u, m) \in U \times M | r_{u,m} = 1\}$, ensemble des couples (u, m) où l'utilisateur u a noté le film m ;
- $\forall u_0 \in U, \mathcal{M}(u_0) = \{m | r_{u_0,m} = 1\}$, l'ensemble des films qu'a vu l'utilisateur u_0 ;
- $\forall m_0 \in M, \mathcal{U}(m_0) = \{u | r_{u,m_0} = 1\}$, l'ensemble des utilisateurs qui ont vu le film m_0 ;
- $\forall u_0 \in U, \overline{\mathcal{M}(u_0)} = M \setminus \mathcal{M}(u_0)$, l'ensemble des films que n'a pas vu l'utilisateur u_0

Note des auteurs : Ce rapport va de pair avec le code implémenté sous R qui est disponible sur `https://github.com/kkmm001/StatApp-Collaborative-Filtering`.

1 Introduction

1.1 Contexte et justification

En facilitant l'accès à l'information, l'avènement du numérique met en avant un autre problème : celui du choix. En effet, avec l'évolution de la quantité d'information disponible sur Internet, le nombre de choix qui nous sont offerts augmente sans cesse. Livres, jeux, musiques, images et films sont autant d'éléments pour lesquels nous sommes amenés à opérer une sélection sans avoir ni le temps, ni la possibilité de considérer toutes les informations nécessaires.

Pour résoudre ce problème, il est de plus en plus fréquent de proposer aux internautes d'attribuer une note à un article (sur une échelle de 1 à 10, ou 1 à 5) en fonction de leur goût. Ces notes constituent un ensemble d'information qui pourront orienter d'autres utilisateurs du web dans leurs décisions quotidiennes. A titre d'illustration, si un internaute a apprécié un article, il est fort probable qu'un autre internaute ayant des goûts similaires à ce dernier apprécie également ledit article. Il est donc possible de tirer profit des informations disponibles sur les goûts de certains internautes pour induire des préférences sur les choix d'autres internautes. La formalisation et l'exploitation de cette intuition sont l'objet des travaux en filtrage collaboratif (collaborative filtering) qui est un cas particulier de filtrage, qui a pour principe d'exploiter les évaluations faites sur certains documents par un utilisateur, afin de recommander ces mêmes documents à d'autres utilisateurs proches de lui, et sans qu'il soit nécessaire d'analyser le contenu des documents [5].

Dans le cadre de cette étude, nous nous intéressons à la recommandation de films pour des internautes grâce aux données issues de la plateforme MovieLens.

1.2 Position du problème

Les données mises à notre disposition concernent un ensemble N_U d'utilisateurs qui sont amenés à noter (en donnant une note discrète comprise entre 1 et 5) un échantillon de N_M films. Les données sont donc représentées par une matrice Y de taille $(N_U \times N_M)$ dont l'élément y_{um} est la note donnée par l'individu u au film m quand elle existe.
La difficulté réside dans le fait qu'un individu ne peut regarder qu'un nombre fini de films et donc toutes les réalisations (u, m, y_{um}) ne sont pas observées ; en général, pour des grandes bases de données, on a moins de 5 % de réalisations. Le but est donc de prédire sur la base d'une quantité d'information restreinte les notes que chaque utilisateur attribuera aux films qu'il n'a pas encore regardés.

1.3 Résultats attendus

Pour atteindre notre but, nous commencerons par mettre en oeuvre un certain nombre de méthodes dites naïves ; nous utiliserons ensuite les méthodes des plus proches voisins et nous terminerons par les méthodes de décomposition de matrice en faible rang. Pour chaque méthode, l'erreur quadratique moyenne (Root Mean Square Error : RMSE) et le temps d'exécution seront mesurés. Nous nous attendons à ce que les méthodes naïves soient les plus rapides, mais aussi les moins précises (RMSE > 1). Pour les deux autres familles de méthodes, nous nous attendons à ce qu'elles soient plus lentes à l'éxécution, mais beaucoup plus précises (RMSE < 1).

1.4 Données disponibles

Dans ce projet, nous disposons des bases issues du site web `http://grouplens.org/datasets/movielens/`. Ainsi, quatre bases de taille variable sont disponibles :
◇ ml-100k avec près de 100 000 notes attribuées par 900 utilisateurs à 1600 films ;
◇ ml-1m avec près de 1 000 000 notes attribuées par 6 000 utilisateurs à 4 000 films ;
◇ ml-10m avec près de 10 000 000 notes attribuées par 71 000 utilisateurs à 10 000 films ;
◇ ml-20m avec 20 000 000 notes attribuées par 138 000 utilisateurs à 27 000 films.

Ces quatre problèmes disposent de fichiers contenant :
◇ les notes attribuées par les utilisateurs aux films, noté data.Ratings ;
◇ les descriptions des films, noté data.Movies ;
◇ les descriptions des utilisateurs, noté data.Users quand le fichier existe.

1.5 Présentation de la base ml-100k

Nous considérons les bases de données suivantes : data.Ratings, data.Movies et data.Users. Nous noterons dans la suite :

- N_U le nombre d'utilisateurs ;
- N_M le nombre de films ;
- N le nombre de notes.

Pour la base ml-100k, après suppression des doublons : $N_U = 943$, $N_M = 1663$ et $N = 99737$.

La base des notes : data.Ratings Cette base comprend pour chaque couple (utilisateur, film) un entier comprise en 1 et 5 représentant la note attribuée par l'utilsateur au film.

La base des films : data.Movies Cette base comprend toutes les données pour caractériser un film :
◇ l'identifiant du film ;
◇ le titre du film ;
◇ l'année de sortie ;
◇ la variable `IMDbURL` qui indique le lien url du film sur le site `http://imdb.com` ;
◇ 19 variables booléennes qui caractérisent le genre cinématographique : `unknown`, `action`, `adventure`, `animation`, `children.s`, `comedy`, `crime`, `documentary`, `drama`, `fantasy`, `film.noir`, `horror`, `musical`, `mystery`, `romance`, `sci.fi`, `thriller`, `war`, `western`.
Ainsi, chaque film est caractérisé par 23 variables, dont 19 booléennes.

La base des utilisateurs : data.Users Cette base comprend toutes les données pour caractériser un utilisateur :
◇ l'identifiant de l'utilisateur ;
◇ l'âge ;
◇ le sexe ;
◇ l'activité professionnelle ;
◇ le code postal.

Un extrait de chaque base du problème ml-100k est disponible en annexe (tableaux 10, 11 et 12).

1.6 Notation pour la prédiction

Soit $U = \{u_1, u_2, ..., u_{N_U}\}$ l'ensemble des utilisateurs et $M = \{m_1, m_2, ..., m_{N_M}\}$ l'ensemble des films.
Considérons la matrice $Y = (y_{u,m})_{(u,m)\in U\times M}$ telle que :

$$y_{u,m} = \text{note donnée par l'utilisateur } u \text{ pour le film } m \, (y_{u,m} \in [\![1, 5]\!])$$

Y est donc une matrice $N_U \times N_M$. A quoi ressemble une telle matrice ?
Pour le cas du probleme ml-100k, Y est une matrice de taille 943×1663 comprenant 1 568 209 éléments dont exactement 99 737 valeurs non nulles ; donc le taux de complétion est de 6.4%. C'est donc une matrice creuse.
Pour recommander des films, nous allons tenter de prédire les notes de tous les films non notés de

la base par l'utilisateur. Introduisons pour cela les variables booléennes $r_{u,m}$ pour $(u,m) \in U \times M$ telles que :

$$r_{u,m} = \begin{cases} 1 & \text{si l'utilisateur } u \text{ a noté le film } m \\ 0 & \text{sinon} \end{cases}$$

Ainsi $y_{u,m}$ (la note) n'a de sens que si $r_{u,m} = 1$. Le but est donc de déterminer les éléments de $Y = (y_{u,m})$ tels que $r_{u,m} = 0$.

Remarque : les prédictions utiliseront uniquement les données présentes dans la base des notes data.Ratings. Aucune information issue de la base des utilisateurs ou de la base des films ne sera exploitée par le programme.

Nous noterons les ensembles suivants :
- $\Omega_0 = \{(u,m) \in U \times M | r_{u,m} = 0\}$;
- $\Omega_1 = \{(u,m) \in U \times M | r_{u,m} = 1\}$;
- $\forall u_0 \in U, \mathcal{M}(u_0) = \{m | r_{u_0,m} = 1\}$, l'ensemble des films qu'a vu l'utilisateur u_0 ;
- $\forall m_0 \in M, \mathcal{U}(m_0) = \{u | r_{u,m_0} = 1\}$, l'ensemble des utilisateurs qui ont vu le film m_0

2 Leçons de la littérature

Les publications sur le filtrage collaboratif abondent sur Internet. Ainsi, avant de mener des analyses plus approfondies sur le sujet de notre étude, il importe de présenter quelques résultats des études précédentes en rapport avec celui-ci.

L'équipe *BellKor's Pragmatic Chaos* a remporté en 2009 le grand prix NetFlix, en améliorant de 10,06 % le propre algorithme de NetFlix pour prédire les évaluations. En effet, le prix Netflix était un concours organisé sur plusieurs années entre 2006 et 2009, afin de créer le meilleur algorithme de filtrage collaboratif c'est-à-dire celui qui prédise au mieux les notes des utilisateurs dans leurs choix de films. Pour ce faire, l'équipe a utilisé une base de 100 480 507 de notes concernant 17 770 films et 480 189 utilisateurs anonymes. Deux grands groupes de modèles ont été utilisées :
- les méthodes de voisinage fondés sur des indices de similarité (corrélation de pearson, corrélation des rangs de Spearman, ...) entre utilisateurs ou (exclusif) entre films ;
- les modèles à facteurs latents basés sur une décomposition de faible rang de la matrice des notes utilisateurs×films qui est très creuse avec une éventuelle contrainte de régularisation ;

La validation croisée a été utilisée pour estimer la pertinence de chaque modèle de prédiction et obtenir les paramètres optimaux. Le critère de sélection de la meilleure méthode était la minimisation de l'erreur quadratique moyenne. Le meilleur résultat auquel ils ont abouti (erreur de prédiction) est de 0.88 avec les modèles à facteurs latents.
Notons néanmoins que, bien que l'équipe *BellKor's Pragmatic Chaos* ait eu des résultats intéressants, leurs algorithmes n'ont jamais été utilisés par NetFlix. En effet, l'algorithme vainqueur du concours n'était pas adapté à la base de données de NetFlix. Ainsi, un bon algorithme doit d'une part engendrer des prédictions précises et d'autre part avoir un temps d'exécution minimal.

Abondant dans le même sens, Somnath Banerjee et Krishnan Ramanathan ont implémenté en 2008 un système de recommandations sur un ensemble de données biaisées. Ces derniers ont utilisé deux jeux de données :
- les données provenant d'une agence indienne de publicité en ligne et concernant 5 391 436 enregistrements (soient 869 478 utilisateurs distincts et 232 annonces publicitaires) ;

— l'ensemble de données MovieLens contenant 1 million d'enregistrements, 6400 utilisateurs et 3900 films.

Les évaluations sont faites sur une échelle de 5 étoiles.
Les deux grandes méthodes de filtrage collaboratif utilisées étaient :

— le modèle sémantique latent, qui suppose l'existence de variables de classes latentes dans la génération du triplet (utilisateur-item-note). Les paramètres du modèle étaient estimés en utilisant l'algorithme EM, en maximisant la probabilité d'observer les données. ;
— la méthode des plus proches voisins.

Les paramètres étaient obtenus par validation croisée : chaque jeu de données était divisé de façon aléatoire en 70 % pour l'apprentissage et 30 % pour le test.

3 Méthodologie

3.1 Quantification de l'erreur

A partir de l'étude de la littérature, une métrique semble dominer : la RMSE.
Soient $y_1 = (y_1^1, y_1^2, ..., y_1^n), y_2 = (y_2^1, y_2^2, ..., y_2^n) \in \mathbb{R}^n$, la métrique RMSE est définie par :

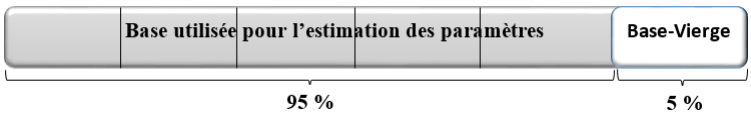
$$\text{RMSE}(y_1, y_2) = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(y_1^k - y_2^k)^2}$$

D'autres métriques apparaissent également dans la littérature comme MAE et NMAE mais elles ne seront pas utilisées dans ce projet.

3.2 Validation croisée

Afin de pouvoir comparer les scores des différents algorithmes sans problème de sur-apprentissage, on met de côté 5% de l'échantillon d'origine qu'on appelle base vierge et qui servira de base test par la suite.

FIGURE 1 – Découpage de la base originelle



Pour estimer la pertinence de chaque modèle et trouver le ou les paramètres optimaux, nous avons utilisé une méthode fondée sur une technique d'échantillonnage : la méthode de validation-croisée appelée « k-fold cross-validation » avec k = 5.
On divise de manière aléatoire l'échantillon d'origine privée de la base Vierge en k sous-échantillons, puis on sélectionne un des k sous-échantillons comme ensemble de validation et les (k-1) autres sous-échantillons constituent l'ensemble d'apprentissage. Le modèle est bâti sur l'échantillon d'apprentissage et validé sur l'échantillon de validation. L'erreur est estimée en calculant un score de performance du modèle sur l'échantillon de validation grâce à l'erreur quadratique moyenne. Lorsqu'un utilisateur ou un film est présent dans la base de test mais absent de la base d'apprentissage (ce qui arrive lorsqu'un film n'est noté qu'une seule fois), la note prédite est NA, ce qui n'impacte pas l'évaluation de l'erreur. Enfin on répète l'opération en sélectionnant un autre échantillon de validation parmi les (k-1) échantillons qui n'ont pas encore été utilisés pour la validation du modèle. L'opération se répète donc k fois pour que chaque sous-échantillon ait été utilisé exactement une fois comme ensemble de validation. La moyenne des k erreurs quadratiques moyennes est enfin calculée pour estimer l'erreur de prédiction de chaque algorithme.

4 Prédiction

4.1 Algorithmes naïfs

Considérons $u_0 \in U, m_0 \in M$ tels que $(u_0, m_0) \in \Omega_0$.

Aléatoire (`random-unif`) On affecte de manière aléatoire suivant une distribution uniforme sur $[\![1, 5]\!]$ une note à l'élément y_{u_0,m_0}.

Aléatoire (`random-samp`) On affecte de manière aléatoire suivant la distribution observée des notes, une note à l'élément y_{u_0,m_0}.

Note unique : la moyenne de toutes les notes (`mean`) Cette seconde approche donne à tous les éléments la valeur :

$$y_{u_0,m_0} := \overline{y} \triangleq \frac{1}{N} \sum_{(u,m)\in\Omega_1} y_{u,m}$$

Note unique : la moyenne des moyennes des films (`meanOfMovies`) Ici, ce sera une autre valeur qui sera attribuée à l'ensemble Ω_0 : on affecte à y_{u_0,m_0} la moyenne des moyennes des films.

Note unique : la moyenne des moyennes par utilisateur (`meanOfUsers`) Par analogie, on affecte ici la valeur la moyenne des moyennes des utilisateurs.

Prédiction par la moyenne des notes du film (`meanByMovie`) On affecte dans ce cas la même note à l'ensemble $\{y_{u,m_0} | (u, m_0) \in \Omega_0\}$ la valeur :

$$y_{u_0,m_0} := \overline{y_{\cdot,m_0}} \triangleq \frac{1}{|\mathcal{U}(m_0)|} \sum_{u\in\mathcal{U}(m_0)} y_{u,m_0}$$

$\overline{y_{\cdot,m_0}}$ représente la note moyenne donnée au film m_0 par les utilisateurs.

Prédiction par la moyenne des notes de l'utilisateur (`meanByUser`) De manière duale, on affecte ici la même note à l'ensemble $\{y_{u_0,m} | (u_0, m) \in \Omega_0\}$ la valeur :

$$y_{u_0,m_0} := \overline{y_{u_0,\cdot}} \triangleq \frac{1}{|\mathcal{M}(u_0)|} \sum_{m\in\mathcal{M}(u_0)} y_{u_0,m}$$

$\overline{y_{u_0,\cdot}}$ représente la note moyenne donnée par l'utilisateur u_0.

Certaines de ces méthodes sont totalement inefficaces pour du filtrage collaboratif (toutes hormis la prédiction par la moyenne des notes des films) car certaines prédictions ne permettent pas d'établir un classement de films, empéchant de fait la recommandation ...

4.1.1 Résultats des algorithmes naïfs

Pour les algorithmes naïfs, une fois établie la méthode de prédiction, nous nous sommes confrontés à une nouvelle difficulté. En effet, à travers la prédiction de notes comme étant la moyenne des films, il arrivait que celle-ci vaille 5 (la note maximale, donc très recommandable) car le film n'avait été noté qu'une seule fois. Pour remédier à ce soucis d'objectivité, nous avons dû introduire un seuil de visionnage. Il suffit alors de recommander parmi les films non-visionnés par l'utilisateur qui ont dépassés ce seuil de visionnage ceux qui ont récolté les meilleures moyennes. Il ressort du tableau 13 en annexe que la méthode de prédiction par la moyenne des notes des

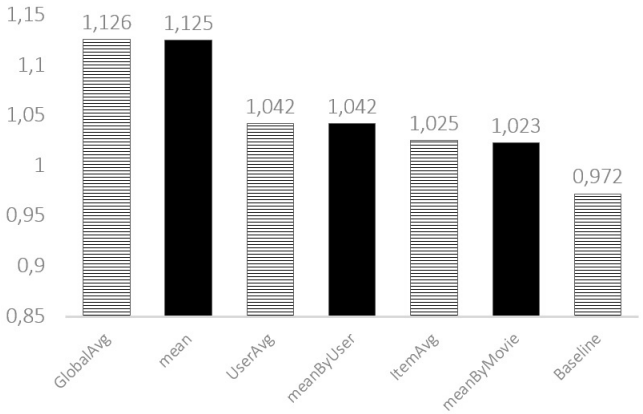
films est la meilleure parmi les méthodes naïves car elle minimise l'erreur RMSE comme le montre le tableau récapitulatif suivant :

TABLE 1 – Moyenne des erreurs des méthodes naïves

	Moyenne des erreurs RMSE
`random-unif`	**1.699**
`random-samp`	**1.594**
`meanOfMovies`	**1.207**
`meanOfUsers`	**1.127**
`mean`	**1.125**
`meanByUser`	**1.042**
`meanByMovie`	**1.023**

La figure ci-dessous compare les erreurs RMSE de nos résultats (en noir) avec ceux des benchmarks (en motifs rayés).

FIGURE 2 – Benchmark pour les méthodes naïves



Détails des méthodes

- mean, meanByUser et meanByMovie sont les méthodes implémentées dans ce rapport ;
- GlobalAbg, UserAvg et ItemAvg sont les mêmes méthodes que celles vues dans ce rapport, les résultats sont disponibles sur le site `http://www.librec.net` ;
- Baseline est une méthode développée dans [2] qui utilise à la fois la moyenne de l'utilisateur et la moyenne du film.

4.2 Méthode des plus proches voisins

Dans cette partie, nous développerons la méthode User-User Collaborative Filtering. Cette technique consiste, pour un individu, à rechercher un échantillon d'utilisateurs qui lui sont le plus proches, au sens d'une certaine similarité. Ainsi, toujours pour cet individu, et pour un film donné, il devient possible de prédire la note qu'il aurait donné à ce film à partir des notes de ses plus proches voisins.

Cette méthode repose sur deux hypothèses :

— les goûts cinématographiques ne changent pas au cours du temps ;

— les individus qui ont eu les mêmes goûts cinématographiques dans le passé sont plus susceptibles de partager les mêmes goûts dans le futur.

Techniquement, cette méthode fera intervenir différents paramètres, ce qui multipliera le nombre d'approches possibles. Une prédiction dépendra en effet :

— de la notion de similarité utilisée ;
— du nombre de plus proches voisins considéré ;
— du prédicteur (la fonction d'évalution de la note)

4.2.1 Notion de similarité

Pour calculer la similarité entre deux utilisateurs, on se restreint aux seules composantes qu'ils ont en commun. Ainsi pour définir les plus proches voisins, il faut se donner une mesure de similarité entre utilisateurs. Comme le coefficient de corrélation de Pearson est l'une des similarités les plus présentes dans la littérature, comme le précise l'article [11] d'une part. Et que d'autre part la corrélation de Pearson est la seule métrique recommandée dans l'article [6] de Herlocker et al, qui ont essayé différentes approches pour la méthode des plus proches voisins, c'est pourquoi nous l'avons donc choisi. Notons toutefois que d'autres métriques apparaissent également dans la littérature comme la corrélation de Spearman ou la similarité cosinus.

La corrélation de Pearson Considérons y_{u_1} et y_{u_2} les vecteurs des notes attribuées aux films visionnés par les utilisateur u_1 et u_2. Le coefficient de corrélation de Pearson entre l'individu u_1 et u_2 est défini par :

$$s_{u_1,u_2} = \frac{\sum_{m \in \mathcal{M}(u_1) \cap \mathcal{M}(u_2)} (y_{u_1,m} - \overline{y_{u_1,\cdot}})(y_{u_2,m} - \overline{y_{u_2,\cdot}})}{\sqrt{\sum_{m \in \mathcal{M}(u_1) \cap \mathcal{M}(u_2)} (y_{u_1,m} - \overline{y_{u_1,\cdot}})^2 \sum_{m \in \mathcal{M}(u_1) \cap \mathcal{M}(u_2)} (y_{u_2,m} - \overline{y_{u_2,\cdot}})^2}}$$

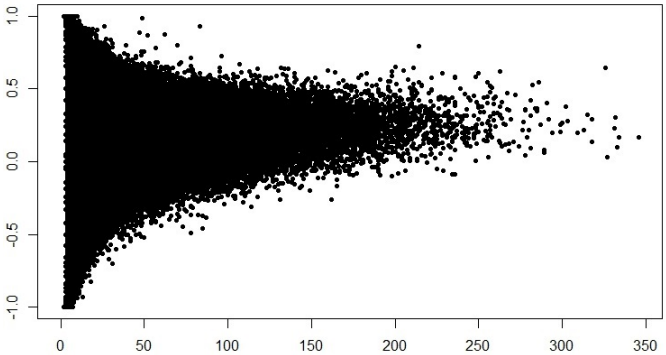
Ainsi seules les notes des films vu à la fois par l'utilisateur u_1 et l'utilisateur u_2 sont prises en compte pour le calcul du coefficient de corrélation de Pearson.

La valeur du coefficient de corrélation est comprise entre – 1 et 1 : une valeur proche de 1 signifie que les deux individus notent les films de la même manière (ils ont donc des goûts cinématographiques similaires), tandis qu'une corrélation proche de -1 signifie que leur notation est opposée (leurs goûts sont antagonistes).

Comme révélé par l'article [6] et les travaux de [4], un défaut de corrélation apparaît lorsque deux individus ont peu de films notés en commun : *"In our experience with collaborative filtering systems, we have found that it was common for the active user to have highly correlated neighbors that were based on a very small number of co-rated items. These neighbors that were based on tiny samples (often three to five co-rated items) frequently proved to be terrible predictors for the active user."* (Herlocker and al, 2002).

En effet, d'après la figure 3, il y a lien entre une similarité élevée et le nombre de films en commun, qui a lui-même un lien avec le nombre de film noté par individu. Il apparaît que les individus ayant une similarité forte partagent peu de films en commun. Compte tenu du fait que la prédiction sera obtenue à partir de ces "voisins", les résultats seront potentiellement médiocres.

FIGURE 3 – Lien entre corrélation de Pearson et nombre de films en commun



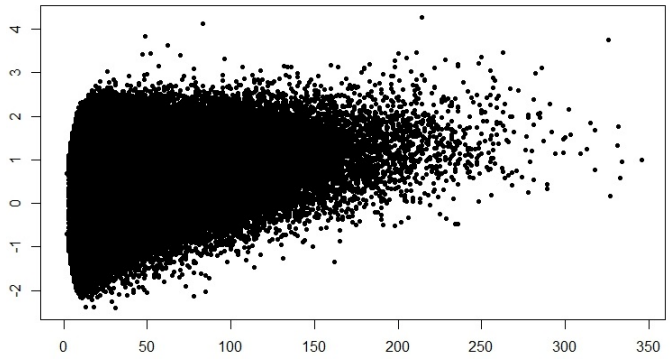
Pour résoudre ce problème, nous avions dans un premier temps implémenté une version basée sur "un seuil de voisinage" : seuls les voisins ayant un nombre de films en commun supérieur à un seuil défini pouvait se prétendre voisin de l'utilisateur actif. Cependant, les résultats ne semblent pas être significatifs pour les prédicteurs considérés dans l'étude (voir les sous-parties 4.2.2 et 4.2.4). L'implémentation sous R comprend toujours cette nuance mais aucun résultat ne sera abordé dans le rapport.

La corrélation RFP Pour palier ce problème, nous avons innové dans un second temps avec une nouvelle similarité : RFP (pour ratings-frequency Pearson), définie par le produit de la corrélation de Pearson et du logarithme du nombre de films vus en commun :

$$s_{u_1,u_2} = \frac{\sum_{m\in\mathcal{M}(u_1)\cap\mathcal{M}(u_2)} (y_{u_1,m} - \overline{y_{u_1,\cdot}})(y_{u_2,m} - \overline{y_{u_2,\cdot}})}{\sqrt{\sum_{m\in\mathcal{M}(u_1)\cap\mathcal{M}(u_2)} (y_{u_1,m} - \overline{y_{u_1,\cdot}})^2 \sum_{m\in\mathcal{M}(u_1)\cap\mathcal{M}(u_2)} (y_{u_2,m} - \overline{y_{u_2,\cdot}})^2}} \times \ln(|\mathcal{M}(u_1)\cap\mathcal{M}(u_2)|)$$

Ainsi, le nombre de films en commun intervient directement dans cette formule : pour une même corrélation de Pearson, un voisin ayant plus de films en commun sera privilégié dans le cas où la similarité de Pearson était positive. Cette nouvelle définition nous est venue grâce à la méthode TF-IDF utilisée dans l'extraction de mots-clés. Cette fois-ci, les voisins avec une similarité élevée (et donc choisi par l'algorithme lors du calcul de la prédiction) seront de meilleurs qualités (c'est-à-dire qu'ils partageront davantage de films en commun avec l'utilisateur actif), comme le suggère la figure 4.

FIGURE 4 – Lien entre corrélation RFP et nombre de films en commun



Remarque : lors de l'implémentation du programme calculant les similarités en individus, la similarité avec soi-même n'a pas été calculée : la valeur NA était retournée. En effet, un utilisateur ne sera jamais considéré comme son propre voisin.
Deux autres similarités basées sur des notions de distance ont été implémentées dans le programme mais ne seront pas abordées dans le rapport.

4.2.2 Prédicteurs

Un prédicteur est une fonction qui retourne une note à partir de la matrice de similarité, du nombre de plus proches voisins et des notes des utilisateurs. Nous en avons explicité et implémenté cinq (liste non exhaustive).

Soient un utilisateur u_0 et un film m_0 tels que $(u_0, m_0) \in \Omega_0$. On notera dans la suite s_{u_1,u_2} la similarité entre les utilisateurs u_1 et u_2.
L'ensemble $\mathcal{N}_{K,s}(u_0, m_0) = \{u \in U | (u, m_0) \in \Omega_1$ et $s_{u_0,u}$ parmi les K plus grandes valeurs de $s_{u_0,\cdot}\}$ désigne les voisins considérés pour u_0 dans la prédiction de la note du film m_0 avec la similarité

s, où $s_{u_0,\cdot}$ désigne l'ensemble des similarités de u_0 avec les autres utilisateurs. Nous pouvons remarquer que le cardinal de $\mathcal{N}_{K,s}(u_0, m_0)$ n'est pas fixe : pour un film m_0 vu moins de K, le cardinal vaut $|\mathcal{U}(m_0)|$, et K sinon.

Prédicteur mean Le premier consiste, pour un nombre de voisins K fixé à prédire :

$$\text{prédicteur } \texttt{mean} \qquad y_{u_0,m_0} := \frac{1}{|\mathcal{N}_{K,s}(u_0,m_0)|} \sum_{u \in \mathcal{N}_{K,s}(u_0,m_0)} y_{u,m_0},$$

où $|\mathcal{U}(m_0)|$ désigne le nombre d'utilisateurs ayant vu le film m_0. Dans cette première approche, la somme est effectuée sur les voisins de u_0 ayant vu le film considéré : la somme contient donc au maximum K termes.

Prédicteurs weighted et weighted&a Les deux suivants consistent à pondérer la note par la similarité : plus un voisin est proche de l'utilisateur considéré, plus sa note aura de l'importance dans la prédiction.
Ainsi les deux prédicteurs sont `weighted` et `weighted&a` (`&a` désigne la valeur absolue au dénominateur) :

$$\text{prédicteur } \texttt{weighted} \qquad y_{u_0,m_0} := \frac{\sum_{u \in \mathcal{N}_{K,s}(u_0,m_0)} y_{u,m_0} \times s_{u,u_0}}{\sum_{u \in \mathcal{N}_{K,s}(u_0,m_0)} s_{u,u_0}},$$

et

$$\text{prédicteur } \texttt{weighted\&a} \qquad y_{u_0,m_0} := \frac{\sum_{u \in \mathcal{N}_{K,s}(u_0,m_0)} y_{u,m_0} \times s_{u,u_0}}{\sum_{u \in \mathcal{N}_{K,s}(u_0,m_0)} |s_{u,u_0}|}$$

L'utilisation de la valeur absolue nous a été suggérée par l'article [12].

Prédicteurs weighted-centered et weighted-centered&a Les deux derniers prédicteurs reposent sur la pondération-centrée des notes : la note prédite dépend directement des moyennes des utilisateurs.
Ainsi, ces deux prédicteurs sont `weighted-centered` et `weighted-centered&a` :

$$\text{prédicteur } \texttt{weighted-centered} \qquad y_{u_0,m_0} := \overline{y_{u_0,\cdot}} + \frac{\sum_{u \in \mathcal{N}_{K,s}(u_0,m_0)} (y_{u,m_0} - \overline{y_{u,\cdot}}) \times s_{u,u_0}}{\sum_{u \in \mathcal{N}_{K,s}(u_0,m_0)} s_{u,u_0}},$$

et

$$\text{prédicteur } \texttt{weighted-centered\&a} \qquad y_{u_0,m_0} := \overline{y_{u_0,\cdot}} + \frac{\sum_{u \in \mathcal{N}_{K,s}(u_0,m_0)} (y_{u,m_0} - \overline{y_{u,\cdot}}) \times s_{u,u_0}}{\sum_{u \in \mathcal{N}_{K,s}(u_0,m_0)} |s_{u,u_0}|},$$

où $\overline{y_{u_0,\cdot}}$ désigne la note moyenne de l'utilisateur u_0.

D'autres prédicteurs font intervenir la variance des utilisateurs et des films, comme dans l'article de Herlocker and al (1999) [7]. Toutefois, cette approche ne semble pas donner de meilleurs résultats d'après les auteurs de cet article.

Remarque : le fait de considérer les valeurs absolues au dénominateur évite "l'explosion" des prédictions dans certains cas. En effet, si l'on considérons un K proche du nombre de visionnage

du film en question, toutes les personnes ayant visionnées interviendront dans la prédiction. Nous considérerons donc dans la formule de prédiction de la note des individus tantôt proches (coefficient de corrélation positive) et tantôt éloignés (négative). La somme des similarités peut ainsi être proche de la valeur nulle, entraînant de fait l'"explosion" des prédictions. Pour pallier ce problème, nous avons considéré la somme des valeurs absolues (d'où le rajout de `&a`). De plus, lors de chacune des prédictions, une majoration à 5 et une minoration à 1 sera faite.

4.2.3 Nombre de plus proches voisins

Pour déterminer le nombre de plus proches voisins optimal, il faudra faire varier ce nombre tout en conservant les autres paramètres et évaluer en conséquence l'erreur RMSE. Ainsi, nous pouvons définir un modèle par :

$$\text{modele} = \text{similarité} + \text{prédicteur}$$

Dans la suite, on notera `RFP_weighted-centered` pour désigner le modèle de plus proches voisins basé sur la similarité RFP et le prédicteur `weighted-centered`.

Pour chacun des modèles étudiés, nous évaluerons tout d'abord, avec un pas de 10, les erreurs RMSE pour K allant de 10 à 100. Puis, une fois la zone de K optimale identifiée, une approche plus fine sera abordée afin d'estimer $K_{optimal}$.

4.2.4 Résultats de la méthode des plus proches voisins

Algorithme de prédiction Ci-dessous est décrit l'algorithme de prédiction basé sur la méthode des plus proches voisins.

Soit un utilisateur u_0 et un film m_0 tels que $(u_0, m_0) \in \Omega_0$.
On considère un modèle avec une similarité s, un nombre de plus proches voisins K et un prédicteur $p : (u_0, m_0, Y, K, s) \mapsto y_{u_0,m_0}$ où Y est la base des notes.

1. Calcul des similarités : pour tout utilisateur différent de u_0, on calcule la similarité au sens de s entre les deux individus
2. Détermination des voisins de u_0 pour le film m_0 : à partir du calcul précédent, on extrait les (au plus) K les utilisateurs ayant les similarités les plus fortes avec u_0 et ayant vus le film m_0.
3. Calcul de la note : à partir de la fonction de prédiction p, on détermine la note y_{u_0,m_0}.

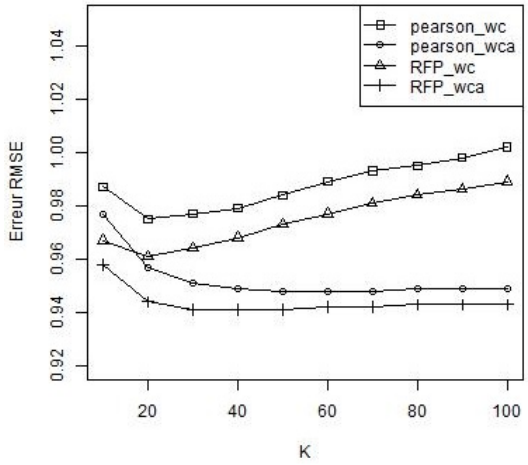
Dans un objectif de clarté, nous avons étudié les deux corrélations (Pearson et RFP) et deux des cinq prédicteurs présentés : `weighted-centered` et `weighted-centered&a`. En effet, une étude préliminaire a montré que les meilleurs résultats étaient obtenus pour ces deux similarités.

Nous avons donc étudié les quatre modèles suivants :
- `pearson_weighted-centered` (noté `pearson_wc`) ;
- `pearson_weighted-centered&a` (noté `pearson_wca`) ;
- `RFP_weighted-centered` (noté `RFP_wc`) ;
- `RFP_weighted-centered&a` (noté `RFP_wca`)

Etude des modèles pour déterminer la zone de K_{opt} Pour chacun des modèles étudiés, nous avons évalué les erreurs RMSE de prédictions pour des K appartenant à {10, 20, ..., 100 } afin d'obtenir une première idée du $K_{optimal}$.

Les résultats obtenus sont agrégés dans la figure 5 et dans les tables 14, 15, 16 et 17 en annexe.

FIGURE 5 – Erreur RMSE en fonction de K pour les 4 modèles



Nous constatons que les modifications des similarités et des prédicteurs peuvent entraîner des améliorations de l'ordre de 3.5% dans notre cas.

Etude des modèles pour K_{optimal} Pour déterminer K_{optimal} de manière plus précise, nous mesurons les erreurs dans les intervalles où K donnent les meilleurs résultats. Les résultats sont agrégés dans les figures de la table 2 et dans la table 3.

TABLE 2 – Erreur RMSE en fonction de K pour chacun des modèles

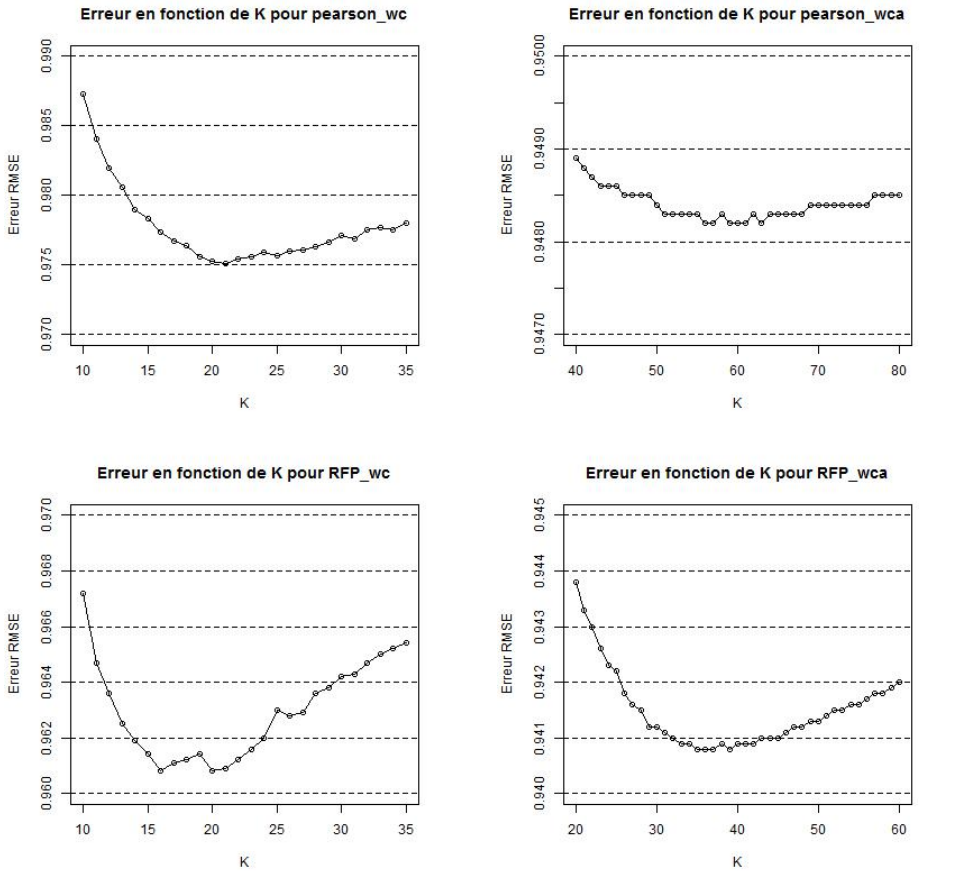
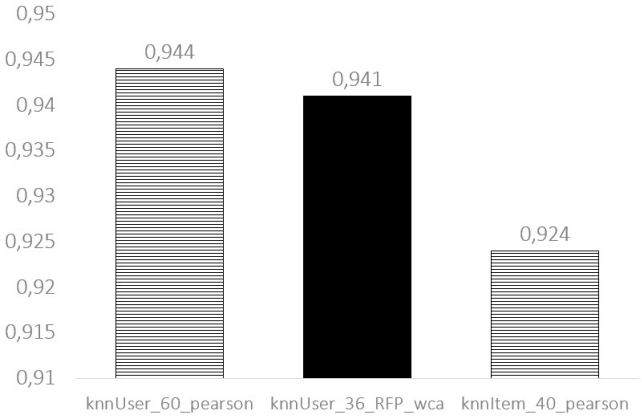


TABLE 3 – K optimaux pour les 4 méthodes

Méthode	K optimaux	**Moyenne des erreurs RMSE**
`pearson_wc`	entre 20 et 22	**0.975**
`pearson_wca`	entre 56 et 63	**0.948**
`RFP_wc`	entre 15 et 22	**0.961**
`RFP_wca`	entre 32 et 42	**0.941**

Au final, parmi les méthodes de plus proches voisins, `RFP_weighted-centered&a` est le meilleur au sens de l'erreur RMSE : celui-ci atteint 0.941 ; c'est la méthode que nous retenons dans la famille des méthodes des plus proches voisins.
La figure ci-dessous compare les erreurs RMSE de notre résultat (en noir) avec ceux des benchmarks (en motifs rayés).

FIGURE 6 – Benchmark pour les méthodes des plus proches voisins



Détails des méthodes
- knnUser_36_RFP_wca est la méthode développée dans ce rapport ;
- knnUser_60_pearson et knnItem_40_pearson sont deux méthodes utilisant la similarité de Pearson. La première considère les 60 plus proches utilisateurs tandis que la seconde repose sur les 40 plus proches films (résultats disponibles sur `http://www.librec.net`).

4.3 Méthode de décomposition en faible rang

Afin de pallier le problème de rapidité des algorithmes des K plus proches voisins, nous avons décidé d'explorer d'autres algorithmes de recommandation. Après une brève revue de la littérature, nous avons décidé d'implémenter la méthode de décomposition en faible rang. Cette méthode s'appuie sur la décomposition en valeurs singulières, abrégée en SVD (de l'anglais : Singular Value Decomposition). Le procédé d'algèbre linéaire de décomposition en valeurs singulières est un outil important de factorisation des matrices rectangulaires permettant la diagonalisation en une base orthonormée de valeurs singulières.
Le reste de cette section présente l'approche naïve de la complétion de la matrice des notes puis celle basée sur la méthode de descente de gradient ainsi que les algorithmes de recommandation basés sur ces méthodes. Enfin la dernière section présente nos résultats.

4.3.1 Méthode SVD

Rappelons brièvement le principe de la décomposition en valeurs singulières :
Soit Y une matrice $m \times n$ dont les coefficients appartiennent au corps $\mathbb{K}$ (où $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$).
Alors il existe une factorisation de la forme : $Y = U\Sigma V^T$, avec :

— U et V deux matrices orthogonales sur $\mathbb{K}$ respectivement de taille $m \times r$ et $n \times r$, où r est le rang de la matrice Y ;
— Σ une matrice diagonale $r \times r$ dont les coefficients diagonaux sont des réels positifs ou nuls.

Les matrices obtenues grâce à la méthode SVD nous fournissent la meilleure approximation du rang de la matrice d'origine Y (meilleur dans le sens où le rang est le plus petit possible). Si on réduit la matrice Σ afin d'obtenir une matrice Σ_k ne contenant que les k plus grandes valeurs singulières ($k < r$) et si les matrices U et V sont réduites de la même manière, alors la matrice reconstruite $Y_k = U_k \ \Sigma_k \ V_k^T$ est considérée comme la plus proche matrice de Y ayant pour rang k.
En d'autres termes, Y_k = argmin || Y - N || sur toutes les matrices N de rang k, où || . || est la norme Frobenius [1].

Notons Ω_1 l'ensemble des informations disponibles dans la base d'apprentissage. On crée à partir de cette base d'apprentissage la matrice des notes Users-Movies noté $Y = (Y_{um})$ définie par :

$$\forall (u,m) \in U \times M, Y_{um} = \begin{cases} y_{u,m} & \text{si } (u,m) \in \Omega_1 \\ 0 & \text{sinon} \end{cases}$$

Cette matrice Y est à l'origine très peu remplie (on dit que c'est une matrice creuse). La première étape de l'algorithme est de « remplir » cette matrice Y afin de supprimer le phénomène de parcimonie. Pour cela deux méthodes ont été choisies : nous avons dans un premier temps utilisé une approche naïve pour la complétion de cette matrice puis nous avons choisi une approche se servant de la méthode de descente de gradient.

4.3.2 Approche de la complétion de matrice par méthode naïve

Pour remplir notre matrice Users-Movies deux approches triviales ont été implémentées : l'une utilise les moyennes des notes par utilisateurs et l'autre les moyennes des notes par film. Ainsi :

$$\forall (u,m) \in U \times M, Y_{um} = \begin{cases} y_{u,m} & \text{si } (u,m) \in \Omega_1 \\ \overline{y_{\cdot,m}} & \text{si approche = moyenne par film} \\ \overline{y_{u,\cdot}} & \text{si approche = moyenne par utilisateur} \end{cases}$$

Voici les étapes de l'algorithme :

1. Création de la matrice de notes Y
2. Complétion de la matrice Y en remplaçant les « zéro » de la matrice par la moyenne du film/la moyenne de l'utilisateur en question
3. Normalisation de la matrice Y en soustrayant de chaque note la moyenne de l'utilisateur en question (à cette étape on obtient une matrice remplie et normalisée)
4. Factorisation de la matrice Y grâce à la méthode SVD : $Y = U \ \Sigma \ V^T$
5. Réduction de la matrice Σ en une matrice Σ_k de dimension k
6. Calcul des matrices résultantes : $U_k \ \Sigma_k$ et $\Sigma_k \ V_k^T$
7. Calcul de la prédiction grâce à ces matrices résultantes
   Pour tout utilisateur u_i et tout film m_j la prédiction est obtenue en calculant le

1. Pour $A \in \mathbb{R}^{n \times m}$, $\| A \|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} |a_{ij}|^2}$

produit de la $u_i{}^{\text{ème}}$ rangée de la matrice $(U_k\ \Sigma_k)$ et de la $m_j{}^{\text{ème}}$ colonne de la matrice $(\Sigma_k\ V_k^T)$ auquel on ajoute la moyenne des notes de l'utilisateur en question ; la formule finale est donc la suivante :
$y_{u_i,m_j} = (U_k\Sigma_k) * (\Sigma_k V_k^T)[u_i, m_j] + \overline{y_{u_i,\cdot}}$

Les étapes 1 à 4 (respectivement 5 à 6) sont présentées sous forme algorithmique dans le pseudo code 2 (respectivement dans le pseudo code 3).

Choix du rang k Plutôt que de nous intéresser directement au nombre k de valeurs singulières à garder nous avons trouvé plus judicieux de transposer le problème en choisissant la quantité $\tau = \dfrac{\sum_{i=1}^{k} \Sigma_{ii}}{\sum_{i=1}^{r} \Sigma_{ii}}$ d'inertie à conserver, qui est liée à un k donné. Nous avons choisi d'évaluer l'algorithme pour des τ compris entre 0.05 et 0.95 afin de trouver le $\tau_{optimal}$ c'est-à-dire celui permettant d'éviter les erreurs d'«overfitting». Les résultats sont présentés dans les tableaux en annexe.

Comme l'avait prédit l'article [1], la méthode de complétion de la matrice en ajoutant la moyenne des notes par film produit un meilleur résultat que celle en ajoutant la moyenne des notes par utilsateur comme le montrent les tableaux 18, 19, 20 et 21 en annexe, et nous trouvons expérimentalement que le τ optimal est $\tau_{optimal} = 0.245$.

4.3.3 Approche de la complétion de matrice par la méthode de descente de gradient

On considère la matrice incomplète formée à partir des notes de la base d'entraînement. L'objectif est de prédire une partie des notes manquantes.

On définit l'opérateur $P_\Omega : \mathbb{R}^{(N_U \times N_M)} \to \mathbb{R}^{(N_U \times N_M)}$ tel que :

$$[P_\Omega(X)]_{um} = \begin{cases} y_{um} & \text{si } (u,m) \in \Omega_1 \\ 0 & \text{sinon} \end{cases}$$

Il s'agit de résoudre le problème d'optimisation sous contrainte suivant :

$$\begin{aligned} &\text{Minimiser Rang(X)} \\ &\text{s.c. } P_\Omega(X) = P_\Omega(Y) \end{aligned} \tag{1}$$

Comme la fonction rang(.) n'est pas convexe, on se retrouve face à un problème NP-hard. Afin de pallier le problème (1), l'article [3] nous propose d'introduire le problème convexe suivant :

$$\begin{aligned} &\text{Minimiser } \| X \|_* \\ &\text{s.c. } P_\Omega(X) = P_\Omega(Y) \end{aligned} \tag{2}$$

En notant $r = \text{rang}(X)$ et $\forall i \in [\![1, r]\!]$, $\sigma_i(X)$ la i^{eme} valeur singulière, alors la norme nucléaire $\| X \|_*$ s'écrit de la manière suivante :

$$\| X \|_* = \sum_{i=1}^{r} \sigma_i(X)$$

Le problème de minimisation (2) se basant sur la norme nucléaire est la meilleure alternative convexe du problème de minimisation du rang de la matrice (1). Pour simplifier la résolution numérique, on reformule le problème comme ci-dessous :

$$\begin{aligned} X &= \arg\min_X f(X) \\ f(X) &= \frac{1}{2} \parallel P_\Omega(X) - P_\Omega(Y) \parallel_F^2 + \lambda \parallel X \parallel_*, \end{aligned} \tag{3}$$

où $\| \cdot \|_F$ est la norme de Frobenius et λ un réel à optimiser.

Algorithme du gradient classique Tout d'abord, nous choisissons l'algorithme du gradient classique (la descente de gradient) pour résoudre le problème de minimisation,

On se donne un point initial aléatoire et un seuil de tolérance $\varepsilon \geq 0$. L'algorithme du gradient définit une suite d'itérés R^{n1*n2}, jusqu'à ce qu'un test d'arrêt soit satisfait.

Il passe de X_k à X_{k+1}, grâce aux étapes suivantes :

1. Simulation : calcul de $\nabla f(X) = \nabla \frac{1}{2} \parallel P_\Omega(X) - P_\Omega(M) \parallel_F^2 + \nabla\lambda \parallel X \parallel_*$

$$\nabla \frac{1}{2} \parallel P_\Omega(X) - P_\Omega(Y) \parallel_F^2 = \begin{cases} \nabla \frac{1}{2} \sum (X_{ij} - Y_{ij})^2 = \sum (X_{ij} - Y_{ij}) & \text{si (i,j)} \in \Omega_1 \\ 0 & \text{sinon} \end{cases}$$

2. Test d'arrêt : si $\nabla \parallel f(X) \parallel \leq \varepsilon$
3. Calcul du pas $\mu_n = \frac{1}{n}$ (n est l'indice de l'itéré)
4. Nouvel itéré : $X_{k+1} = X_k - \frac{\mu_n}{2} \nabla f(X)$

Algorithme du gradient proximal Au lieu d'utiliser la méthode du gradient descente, l'article [10] propose la méthode du gradient proximal pour ce problème. On définit la fonction proximale comme :

$$\text{prox}(M) = \arg\min_X \frac{1}{2\mu} \parallel Y - X \parallel_F^2 + \lambda \parallel X \parallel_*$$

Le problème de minimisation(3) peut se calculer par récurrence telle que :

$$X_{k+1} = \text{prox}(X_k - \frac{\mu_k}{2} \times \nabla \parallel P_\Omega(X_k) - P_\Omega(Y) \parallel_F^2) = \text{prox}(X_k + \mu_k(P_\Omega(Y) - P_\Omega(X_k)))$$

Selon l'article[10], on peut simplifier le calcul de la fonction proximale par

$$\text{prox}(M) = U\Sigma_\lambda V^t,$$

où U et V sont obtenues grâce à la décomposition SVD de Y : $Y = U\Sigma V^t$.
Σ_λ est la matrice diagonale définie par $(\Sigma_\lambda)_{ii} = \max\{\Sigma_{ii} - \lambda, 0\}$

1. Simulation : calcul de $P_\Omega(Y) - P_\Omega(X_k) = \sum_{(i,j)\in\Omega_1} (X_{ij} - Y_{ij})$
2. Test d'arrêt : si $\nabla \parallel f(X) \parallel \leq \varepsilon$
3. Calcul du pas $\mu_k = \frac{1}{k}$ (k est l'indice de l'itéré)
4. Décomposition SVD : $X_k + \mu_k(P_\Omega(Y) - P_\Omega(X_k)) = U\Sigma V^t$
5. Calcul $(\Sigma_\lambda)_{ii} = \max\{\Sigma_{ii} - \lambda, 0\}$
6. Nouvel itéré : $X_{k+1} = U\Sigma_\lambda V^t$

Choix du λ Le λ est le coefficient du terme de régularisation qui pénalise le rang du X. Nous trouvons expérimentalement que $\lambda_{optimal} = 10$ selon la table 23 et 24 en annexe.

4.3.4 Résultats des méthodes de décomposition en faible rang

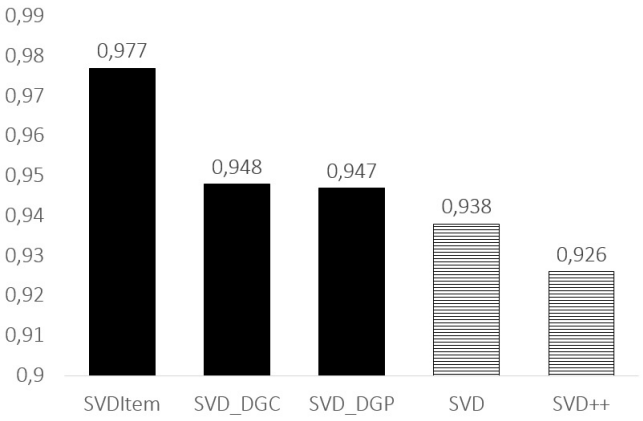
Au final, la méthode SVD naïve qui est la meilleure au sens de l'erreur RMSE est obtenue avec $\tau_{optimal} = 0.245$ et elle atteint 0.977. Quand on effectue 30 itérations dans les méthodes SVD avec descente de gradient classique et descente de gradient proximal les meilleurs scores sont obtenus avec $\lambda_{optimal} = 10$ et valent respectivement 0,948 et 0,947.

TABLE 4 – Synthèse des résultats SVD

Méthode	Paramètre optimal	**Moyenne des erreurs RMSE**
Item	$\tau_{optimal} = 0.245$	**0.977**
Descente de gradient classique	$\lambda_{optimal} = 10$	**0.948**
Descente de gradient proximal	$\lambda_{optimal} = 10$	**0.947**

La figure ci-dessous compare les erreurs RMSE de nos résultats (en noir) avec ceux des benchmarks (en motifs rayés).

FIGURE 7 – Benchmark pour les méthodes de décomposition en faible rang



Détails des méthodes

- SVDItem, SVD_DGC (SVD avec la descente de gradient classique) et SVD_DGP (SVD avec la descente de gradient proximal) sont les méthodes développées dans le rapport ;
- SVD est une méthode développée dans [2] basée sur un prédicteur de type baseline et une descente de gradient stochastique ;
- SVD++ est une méthode développée dans [2] qui améliore les résultats de SVD en rajoutant dans le modèle une composante sur les films.

4.4 Comparaison des performances des différents algorithmes

Comme nous l'avons dit plus haut, nous avons implémenté les différentes méthodes sur la base vierge (5 % de la base initiale) afin de les comparer leurs performances. les résultats obtenus sont contenus dans le tableau ci-dessous :

Table 5 – Résultats sur la base vierge

	Paramètres optimaux	**RMSE**
Méthode naive (meanByMovie)		**1.033**
Plus proches voisins (RFP_wca)	k = 36	**0.953**
SVD approche naive (item)	$\tau = 0.245$	**0.977**
SVD descente de gradient classique	$\lambda = 10$	**0.938**
SVD descente de gradient proximal	$\lambda = 10$	**0.934**

On remarque que les erreurs RMSE ont légèrement augmenté ; ce qui est normal puisque nous n'avons utilisé que 5 % de la base pour la comparaison des performances des différents algorithmes. Ainsi donc, la méthode SVD par descente de gradient proximal semble être la meilleure au sens de la minimisation du RMSE (toutes choses égales par ailleurs).

5 Problème de dimension

La qualité d'un système de recommandation ne se résume pas uniquement à l'erreur commise dans la prédiction. En effet, le temps de calcul, que ce soit lors de l'estimation des paramètres (comme dans les méthodes de knn et SVD) ou lors de la recommandation en elle-même pour l'utilisateur final, est un paramètre à considérer.
En effet, tous les algorithmes ne sont pas utilisables pour de très larges données. Par exemple, bien que l'algorithme basé sur la méthode des plus proches voisins donne de meilleurs résultats que les méthodes naïves, il aura certainement des difficultés a être utilisé sur une base de données plus volumineuse.

Rappelons les dimensions des problèmes de ml-100k et ml-1m :

Table 6 – Dimension des problèmes

Problème	Nombre de notes	Nombre d'utilisateurs	Nombre de films	Taux de complétion
ml-100k	99 684	943	1 663	6.36%
ml-1m	1 000 209	6 040	3 706	4.47%

Tout d'abord, voyons les temps d'exécution des différents algorithmes sur le problème ml-100k. Nous distinguons les trois étapes suivants dans le processus de recommandation :

— la préparation des données : génération des statistiques, listes et matrices nécessaires à la prédiction des 5 bases de test, de la base vierge et de la base complète ;
— prédiction des notes sur les 5 bases de test et estimation des paramètres ;
— recommandation de films pour un utilisateur final

Table 7 – Ordre de grandeur du temps d'exécution pour le problème ml-100k (en secondes)

Méthode	Préparation des données	Prédictions et estimations	Recommandation
naïve (×7)	10	15	1
knn (RFP_wca)	9 000	18 000	120
svd-naïf (×2)	200	450	1
svd-gradient	10	12500	190
svd-proximal	10	13000	205

Nous constatons que l'algorithme basé sur la méthode des plus proches voisins s'exécute lentement sur le problème ml-100k. Ainsi, il est évident que cet algorithme n'est pas utilisable pour les problèmes à grande dimension. Nous ne traiterons dans la suite que des méthodes naïves et basées sur la réduction de matrice, qui semblent pouvoir s'adapter au problème de scabilité.

Ci-dessous se trouvent les résultats portant sur le problème ml-1m :

TABLE 8 – Ordre de grandeur du temps d'exécution pour le problème ml-1m (en secondes)

Méthode	Préparation des données	Prédictions et estimations	Recommandation
naïve (×7)	380	430	1
svd-naïf(×2)	8 000	18 000	2

Finalement, nos résultats montrent que les méthodes basées sur les plus proches voisins et sur la descente de gradient s'adaptent mal au problème de dimension. Recommander sur ml-1m n'est donc faisable qu'avec les deux méthodes reposant sur une logique naïve.

6 Tests sur les recommandations

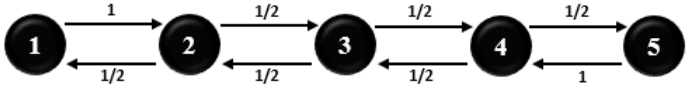
Nous nous sommes proposés comme objectif secondaire d'effectuer différents tests sur les recommandations produites par nos algorithmes.

6.1 Robustesse des recommandations

La méthode des plus proches voisins et la décomposition en faible rang reposent directement sur la matrice des notes. Comment sont affectées les recommandations lorsque cette matrice est légèrement modifiée ?

Pour tester la robustesse des recommandations, nous avons modifié légèrement les notes de l'individu concerné puis nous avons comparé les recommandations avant et après modification à l'aide du coefficient de rang de Spearman. En ce qui concerne la modification, si l'individu a donné la note 5 à un film, on change cette note en 4 ; si la note est plutôt de 1, on la change en 2, et si la note est comprise entre 2 et 4, elle peut évoluer de ± 1 avec probabilité $\frac{1}{2}$.

FIGURE 8 – Illustration de la méthode de robustesse



Nous avons procédé de la manière suivante :

— on choisit la méthode de recommandation ;
— on choisit un seuil de visionnage : c'est le nombre minimal de fois que le film a été noté ;
— pour un utilisateur donné, on choisit le pourcentage de notes à modifier : il est compris entre 0 et 100 % ;
— A la fin, l'algorithme nous renvoit les 2 groupes de recommandation et le coefficient de corrélation.

Notons qu'avec la méthode des plus proches voisins, on a la possibilité de choisir le nombre de plus proches voisins et le prédicteur.

Par ailleurs, le coefficient de corrélation avec la méthode de prédiction naive sera toujours de 1, parce que les notes prédites ne sont pas fonction des individus ; par contre, ce coefficient varie en fonction des différents paramètres en ce qui concerne les autres méthodes, mais reste supérieur à 0,7.

6.2 Biais de popularité

La plupart des systèmes de recommandation présente généralement un biais de popularité communément appelé "effet Harry Potter" (dû au fait que Harry Potter est un film que beaucoup de personnes aiment). Ceci vient du fait que les utilisateurs apprécient généralement les films populaires ainsi en pareille circonstance un système de recommandation n'est pas très intéressant. Parallèlement, il existe souvent des films qui ne sont jamais recommandés.
Afin de voir si les films recommandés sont toujours les mêmes ou s'il y a des films qui ne sont jamais recommandés, nous avons procédé de la manière suivante :

- on choisit la méthode de recommandation ;
- on choisit le nombre de recommandations : c'est le nombre de films que l'on veut recommander à tous les utilisateurs ;
- on choisit un seuil de visionnage : pour qu'un film soit recommandé, il faut qu'il ait été vu un nombre minimal de fois ;
- A la fin, l'algorithme nous renvoit la liste des films toujours recommandés et le nombre de films qui ne sont jamais recommandés

Les principaux résultats auxquels nous aboutissons sont les suivants :

- pour les méthodes naïves, si l'on souhaite recommander au moins 10 films à tous les utilisateurs de la base, il suffit d'imprimer la liste des 50 meilleurs films (du point de vue de leur note moyenne) ;
- en ce qui concerne les deux autres méthodes, on constate que toutes choses égales par ailleurs, il y a en général près d'un millier de films qui ne sont jamais recommandés ;

Ce résultat est très intéressant dans la mesure où il nous permettrait de ne prendre en compte que les films qui sont recommandables lors de la recommandation en elle-même et donc de réduire le temps d'exécution lorsqu'un utilisateur fait appel à l'algorithme (cf algorithme de recommandation par la méthode des plus proches voisins, algorithm 1).

6.3 Corrélation entre les méthodes

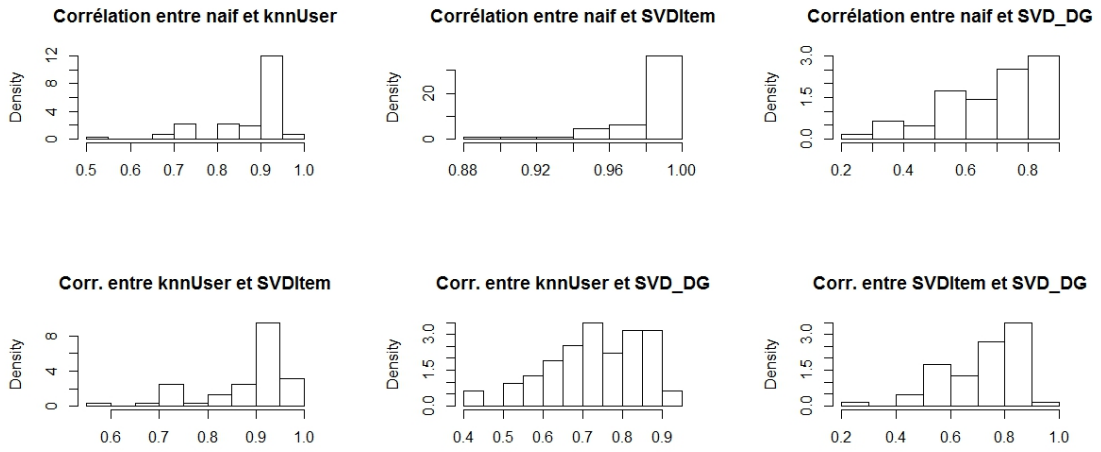
Les systèmes de recommandations vus dans ce rapport semblent réellement différentes. Certaines dépendent de paramètres à optimiser, d'autres non, certaines s'exécutent rapidement tandis que d'autres nécessitent un peu plus de patience ... Nous souhaitons dans cette partie tester la corrélation entre les différentes méthodes : en effet, comme elles ont pour but de recommander des films à un utilisateur elles devraient être fortement corrélées.

Pour cela, nous allons évaluer le coefficient de corrélation de Spearman (corrélation de Pearson appliquée au rang des films lors de la recommandation) entre les méthodes sur l'ensemble des films recommandés par individu.

Nous fixons le seuil de visionnage à 25 (nombre minimal de fois qu'un film doit être noté pour être recommandable), et présentons les corrélations entre les 4 méthodes sur un échantillon de 60 individus.

Les résultats sont résumés dans les figures de la table 9. Ainsi, la corrélation entre deux méthodes pour un individu donné est presque toujours supérieur à 0.5 : cela signifie qu'il y a une cohérence entre les méthodes. Par ailleurs, une corrélation de 0.98 est observée entre l'algorithme naïf et l'algorithme de décomposition de faible rang par approche naïve. Ce résultat est cohérent dans la mesure où l'algorithme naïf est le cas particulier de l'approche par décomposition lorsque le taux d'inertie conservé est de 100%.

TABLE 9 – Corrélation dans les recommandations entre nos quatre méthodes



7 Conclusion

Notre étude avait comme objectif principal de réaliser un système de recommandations de films à partir de la base de données de movielens, contenant les scores que des utilisateurs avaient donnés aux films qu'ils avaient déjà vus. Pour atteindre cet objectif, nous avons implémenté différentes méthodes, des méthodes dites naïves aux méthodes de décomposition de matrices en passant par la méthode des plus proches voisins.

Il ressort que :

- la méthode SVD descente de gradient proximal est la plus précise car elle minimise le RMSE (RMSE = 0.934) ; cette méthode est aussi plus rapide que la méthode des plus proches voisins, ce d'autant plus qu'elle ne demande pas une préparation des données au préalable ;
- le nombre optimal de voisins qui minimise l'erreur RMSE (pour la méthode des plus proches voisins) est de l'ordre de la quarantaine ;
- les méthodes naïves sont les moins précises (RMSE = 1.033), mais les plus rapides ; de plus si l'on souhaite recommander 10 films à chacun des individus via les méthodes naïves, il suffit de remettre à chacun d'eux la liste des 50 films les mieux classés ;
- les recommandations à l'endroit d'un individu varient très peu si l'on modifie légèrement les notes de ce dernier ;
- les différentes méthodes sont plutôt cohérentes en terme de recommandations.

Un prolongement intéressant de ce projet serait d'implémenter d'autres méthodes à l'instar de la modélisation économétrique sur variables catégorielles (qui prendra notamment en compte les goûts et les caractéristiques des utilisateurs) et arriver à exploiter les variables latentes estimées par la factorisation de matrices pour prédire facilement et rapidement une note fiable pour chaque nouvel utilisateur.

Références

[1] Nick Asendorf, Madison McGaffin, Matt Prelee, and Ben Schwartz. Algorithms for completing a user ratings matrix.

[2] Zhouxiao Bao and Haiying Xia. Movie rating estimation and recommendation. *CS229*, 2012.

[3] Emmanuel J Candes and Yaniv Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6) :925–936, 2010.

[4] Leah E. Steinberg Daniel Lew, Ben Sowell and Amrit S. Tuladhar. Recommender Systems. `http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/about.html/`, 2007. [Online ; accessed 02-May-2016].

[5] Jean-Pierre GIRAUDIN. *COCoFil2 : Un nouveau système de filtrage collaboratif basé sur le modèle des espaces de communautés*. PhD thesis, Université Joseph Fourier–Grenoble, 2009.

[6] Jon Herlocker, Joseph A Konstan, and John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information retrieval*, 5(4) :287–310, 2002.

[7] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[8] Anand Rajaraman Jure Leskovec and Jeff Ullman. Mining massive datasets. `https://fr.coursera.org/course/mmds/`, 2015.

[9] Joseph Konstan and Michael D Ekstrand. Introduction to Recommender Systems. `https://fr.coursera.org/learn/recommender-systems/`, 2015.

[10] Mustrum Ridcully. Proximal gradient descent and acceleration. `www.stat.cmu.edu/~ryantibs/convexopt/lectures/08-prox-grad.pdf`.

[11] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[12] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

Nous avons également consulté des cours disponibles sur Coursera, notamment les MOOC [9] et [8].

8 Annexe

Données de ml-100k

Table 10 – data.Ratings

userID	movieID	rating
196	242	3
186	302	3
22	377	1
244	51	2
166	346	1
298	474	4
115	265	2
253	465	5
305	451	3
6	86	3

Table 11 – data.Movies (3 des 23 variables)

movieID	title	date
1	Toy Story (1995)	01-Jan-1995
2	GoldenEye (1995)	01-Jan-1995
3	Four Rooms (1995)	01-Jan-1995
4	Get Shorty (1995)	01-Jan-1995
5	Copycat (1995)	01-Jan-1995

Table 12 – data.Users

userID	age	gender	occupation	zip.code
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213
6	42	M	executive	98101
7	57	M	administrator	91344
8	36	M	administrator	05201
9	29	M	student	01002
10	53	M	lawyer	90703

Résultats pour la partie validation-croisée sur ml-100k

Résultats pour la partie validation-croisée des méthodes naïves sur ml-100k

Table 13 – Erreur RMSE pour les méthodes naïves

méthode	Test 1	Test 2	Test 3	Test 4	Test 5	Moyenne
random_unif	1.705	1.700	1.693	1.702	1.696	1.699
random_samp	1.599	1.587	1.596	1.593	1.593	1.594
meanOfMovies	1.213	1.199	1.207	1.200	1.216	1.207
meanOfUsers	1.129	1.120	1.128	1.128	1.130	1.127
mean	1.128	1.118	1.127	1.126	1.129	1.125
meanByUser	1.038	1.034	1.042	1.046	1.049	1.042
meanByMovie	1.022	1.024	1.025	1.020	1.022	1.023

Résultats pour la partie validation-croisée des méthodes knn sur ml-100k

Table 14 – Erreur RMSE pour pearson_weighted-centered en fonction de K

K	Test 1	Test 2	Test 3	Test 4	Test 5	Moyenne
10	0.981	0.986	0.989	0.988	0.992	0.987
20	0.970	0.974	0.977	0.978	0.978	0.975
30	0.972	0.977	0.979	0.978	0.979	0.977
40	0.974	0.979	0.982	0.981	0.982	0.979
50	0.979	0.984	0.987	0.986	0.985	0.984
60	0.984	0.987	0.992	0.991	0.989	0.989
70	0.988	0.991	0.997	0.995	0.991	0.993
80	0.992	0.993	1.000	0.997	0.993	0.995
90	0.997	0.996	1.000	1.000	0.996	0.998
100	1.000	0.999	1.004	1.004	1.001	1.002

Table 15 – Erreur RMSE pour pearson_weighted-centered&a en fonction de K

K	Test 1	Test 2	Test 3	Test 4	Test 5	Moyenne
10	0.972	0.978	0.976	0.979	0.983	0.977
20	0.951	0.957	0.955	0.959	0.963	0.957
30	0.946	0.950	0.950	0.954	0.956	0.951
40	0.945	0.947	0.948	0.951	0.954	0.949
50	0.944	0.947	0.947	0.950	0.953	0.948
60	0.944	0.947	0.947	0.950	0.953	0.948
70	0.945	0.947	0.947	0.951	0.953	0.948
80	0.945	0.947	0.947	0.951	0.953	0.949
90	0.945	0.947	0.947	0.951	0.953	0.949
100	0.945	0.947	0.947	0.951	0.954	0.949

TABLE 16 – Erreur RMSE pour RFP_weighted-centered en fonction de K

K	Test 1	Test 2	Test 3	Test 4	Test 5	Moyenne
10	0.963	0.968	0.966	0.968	0.972	0.967
20	0.959	0.960	0.960	0.964	0.962	0.961
30	0.962	0.964	0.964	0.968	0.964	0.964
40	0.965	0.967	0.966	0.972	0.968	0.968
50	0.970	0.971	0.973	0.976	0.974	0.973
60	0.974	0.976	0.976	0.981	0.977	0.977
70	0.978	0.980	0.981	0.984	0.980	0.981
80	0.982	0.983	0.985	0.988	0.981	0.984
90	0.986	0.985	0.985	0.991	0.983	0.986
100	0.990	0.988	0.990	0.993	0.987	0.989

TABLE 17 – Erreur RMSE pour RFP_weighted-centered&a en fonction de K

K	Test 1	Test 2	Test 3	Test 4	Test 5	Moyenne
10	0.954	0.960	0.953	0.959	0.962	0.958
20	0.940	0.944	0.940	0.946	0.948	0.944
30	0.937	0.941	0.939	0.944	0.945	0.941
40	0.938	0.940	0.939	0.943	0.945	0.941
50	0.938	0.940	0.940	0.944	0.945	0.941
60	0.939	0.941	0.940	0.945	0.946	0.942
70	0.939	0.941	0.940	0.945	0.946	0.942
80	0.939	0.942	0.941	0.946	0.947	0.943
90	0.939	0.942	0.941	0.946	0.947	0.943
100	0.939	0.942	0.941	0.946	0.948	0.943

Résultats pour la partie validation-croisée des méthodes de décomposition en faible rang sur ml-100k

TABLE 18 – Erreur RMSE pour la méthode SVD-Item en fonction de τ

τ	Test 1	Test 2	Test 3	Test 4	Test 5	Moyenne
0.05	0.975	0.977	0.979	0.985	0.981	0.980
0.15	0.975	0.977	0.979	0.985	0.981	0.980
0.25	0.975	0.979	0.980	0.976	0.978	0.978
0.35	0.990	0.993	0.994	0.988	0.991	0.991
0.45	1.000	1.003	1.004	0.999	1.001	1.001
0.55	1.009	1.012	1.013	1.008	1.011	1.011
0.65	1.015	1.019	1.019	1.014	1.016	1.017
0.75	1.019	1.021	1.023	1.018	1.020	1.020
0.85	1.021	1.024	1.025	1.020	1.021	1.022
0.95	1.022	1.024	1.025	1.020	1.022	1.023

TABLE 19 – Erreur RMSE pour la méthode SVD-User en fonction de τ

τ	Test 1	Test 2	Test 3	Test 4	Test 5	Moyenne
0.05	0.985	0.981	0.989	0.991	0.995	0.988
0.15	0.998	0.994	1.003	1.004	1.008	1.001
0.25	1.012	1.006	1.015	1.015	1.020	1.014
0.35	1.020	1.017	1.025	1.026	1.031	1.024
0.45	1.026	1.023	1.032	1.033	1.037	1.030
0.55	1.033	1.028	1.036	1.039	1.042	1.036
0.65	1.036	1.032	1.039	1.043	1.046	1.039
0.75	1.037	1.033	1.041	1.045	1.049	1.041
0.85	1.038	1.033	1.041	1.046	1.049	1.041
0.95	1.038	1.034	1.042	1.046	1.049	1.042

TABLE 20 – Erreur RMSE pour la méthode SVD-Item en fonction de τ

τ	Test 1	Test 2	Test 3	Test 4	Test 5	Moyenne
0.05	0.9752	0.9774	0.9789	0.9852	0.9811	0.9796
0.07	0.9752	0.9774	0.9789	0.9852	0.9811	0.9796
0.09	0.9752	0.9774	0.9789	0.9852	0.9811	0.9796
0.11	0.9752	0.9774	0.9789	0.9852	0.9811	0.9796
0.13	0.9752	0.9774	0.9789	0.9852	0.9811	0.9796
0.15	0.9752	0.9774	0.9789	0.9852	0.9811	0.9796
0.17	0.9984	1.0015	1.0016	0.9985	0.9998	1.0000
0.19	0.9984	1.0015	1.0016	0.9985	0.9998	1.0000
0.21	0.9984	1.0015	1.0016	0.9985	0.9998	1.0000
0.23	0.9776	0.9844	0.9829	0.9778	0.9813	0.9808
0.25	0.9752	0.9794	0.9799	0.9759	0.9779	0.9777
0.27	0.9771	0.9816	0.9814	0.9783	0.9804	0.9797
0.29	0.9794	0.9842	0.9838	0.9808	0.9820	0.9820
0.31	0.9829	0.9880	0.9875	0.9834	0.9858	0.9855
0.33	0.9862	0.9904	0.9909	0.9865	0.9879	0.9884
0.35	0.9896	0.9932	0.9938	0.9885	0.9907	0.9911

TABLE 21 – Erreur RMSE pour la méthode SVD-User en fonction de τ

τ	Test 1	Test 2	Test 3	BTest 4	Test 5	Moyenne
0.05	0.9850	0.9815	0.9888	0.9914	0.9948	0.9883
0.07	0.9866	0.9837	0.9914	0.9945	0.9962	0.9905
0.09	0.9892	0.9851	0.9930	0.9959	0.9994	0.9925
0.11	0.9917	0.9881	0.9972	0.9989	1.0016	0.9955
0.13	0.9953	0.9911	0.9993	1.0013	1.0042	0.9982
0.15	0.9975	0.9945	1.0032	1.0038	1.0077	1.0013
0.17	1.0012	0.9979	1.0047	1.0058	1.0097	1.0039
0.19	1.0051	1.0003	1.0083	1.0089	1.0117	1.0069
0.21	1.0064	1.0023	1.0104	1.0109	1.0147	1.0089
0.23	1.0083	1.0051	1.0122	1.0135	1.0173	1.0113
0.25	1.0116	1.0063	1.0153	1.0152	1.0196	1.0136
0.27	1.0136	1.0088	1.0180	1.0178	1.0215	1.0159
0.29	1.0148	1.0102	1.0197	1.0202	1.0246	1.0179
0.31	1.0158	1.0110	1.0217	1.0221	1.0276	1.0196
0.33	1.0174	1.0142	1.0238	1.0237	1.0293	1.0217
0.35	1.0200	1.0166	1.0254	1.0258	1.0312	1.0238

TABLE 22 – Erreur RMSE pour la méthode SVD-Item en fonction de τ

τ	Base Test 1	Base Test 2	Base Test 3	Base Test 4	Base Test 5	Moyenne
0.23	0.9776	0.9844	0.9829	0.9778	0.9813	0.9808
0.235	0.9761	0.9808	0.9808	0.9755	0.9789	0.9784
0.24	0.9752	0.9805	0.9795	0.9757	0.9791	0.9780
0.245	0.9742	0.9793	0.9803	0.9751	0.9766	0.9771
0.25	0.9752	0.9794	0.9799	0.9759	0.9779	0.9777
0.255	0.9747	0.9799	0.9807	0.9765	0.9789	0.9781
0.26	0.9752	0.9802	0.9814	0.9773	0.9785	0.9785
0.265	0.9761	0.9817	0.9810	0.9781	0.9792	0.9792
0.27	0.9771	0.9816	0.9814	0.9783	0.9804	0.9797
0.275	0.9778	0.9824	0.9828	0.9794	0.9807	0.9806
0.28	0.9782	0.9831	0.9836	0.9804	0.9810	0.9813
0.285	0.9788	0.9837	0.9837	0.9806	0.9818	0.9817
0.29	0.9794	0.9842	0.9838	0.9808	0.9820	0.9820

TABLE 23 – Erreur RMSE pour la méthode SVD par la descente de gradient classique en fonction de λ

λ	Test 1	Test 2	Test 3	Test 4	Test 5	Moyenne
5	1.0227	1.0151	1.0276	1.0154	1.0212	1.0204
10	0.9464	0.9436	0.9532	0.9458	0.9524	0.9483
15	0.9557	0.9537	0.9624	0.9562	0.9636	0.9583
20	0.9842	0.9798	0.9909	0.9819	0.9908	0.9855
25	1.0121	1.0064	1.0176	1.0104	1.0179	1.0129
30	1.0371	1.0309	1.0429	1.0362	1.0425	1.0379
35	1.0615	1.0552	1.0670	1.0599	1.0656	1.0618
40	1.0819	1.0786	1.0923	1.0807	1.0884	1.0844

TABLE 24 – Erreur RMSE pour la méthode SVD par la descente de gradient proximal en fonction de λ

λ	Test 1	Test 2	Test 3	Test 4	Test 5	Moyenne
5	1.0144	1.0028	1.0186	1.0087	1.0205	1.0130
10	0.9453	0.9423	0.9509	0.9445	0.9527	0.9471
15	0.9539	0.9509	0.9591	0.9539	0.9607	0.9557
20	0.9758	0.9712	0.9809	0.9752	0.9822	0.9771
25	0.9966	0.9921	1.0016	0.9958	1.0022	0.9976
30	1.0140	1.0090	1.0184	1.0138	1.0194	1.0149
35	1.0298	1.0246	1.0348	1.0288	1.0349	1.0306
40	1.0425	1.0372	1.0477	1.0415	1.0469	1.0432

Algorithmes

Data: base des notes, utilisateur, film, K, similarité, prédicteur
Result: La liste des films recommandés
Voisins := couples(voisins, similarité) trié par ordre décroissant de similarité ;
for *film non noté par l'utilisateur* **do**
    Knn := () [les K plus proches voisins de utilisateur pour ce film] ;
    for *voisin dans Voisins ayant noté le film* **do**
        ajouter voisin à Knn ;
    end
    prediction du film := prédicteur(base des notes, Knn, Voisins) ;
end
retourner la liste des films ayant reçu les meilleures prédictions ;

Algorithm 1: Recommandation par User-User Collaborative Filtering

Data: mode,data.Ratings
Result: La décomposition SVD de la matrice de notes Y
nb.Users := nombre d'individus différents dans la data frame data.Ratings ;
nb.Movies := nombre de films différents dans la data frame data.Ratings ;
Y := création d'une matrice de "NA" de taille nb.Users×nb.Movies.
On remplit la matrice Y de la manière suivante :
if *mode = utilisateur* **then**
  for *u dans 1 : nb.Users* **do**
    Y[u,]= moyenne des notes de l'utisateur d'indice u
  end
else
  for *m dans 1 : nb.Movies* **do**
    Y[,m]= moyenne des notes obtenu par le film dindice m
  end
end
for *(utilisateur,film) dans data.Ratings* **do**
  Y[utilisateur,film] = la note donnée par l'utilisateur au film
end
for *u dans 1 : nb.Users* **do**
  Y[u,] = Y[u,] - moyenne des notes de l'utisateur d'indice u
end
Retourner la decomposition SVD de Y

Algorithm 2: svd_filledMatrix

Data: mat.SVD, X
mat.SVD := list(U, Σ, V) tel que $R = U\ \Sigma\ V^T$ i.e. la liste des trois matrices de la décomposition de la matrice des notes R ;
τ := taux d'inertie qu'on souhaite garder ;
Result: Les deux matrices résultantes de la décomposition SVD servant à la prédiction
Inertie.totale := inertie totale de la matrice Σ ;
k := nombres de valeurs singulières nécessaires pour avoir τ% de l'inertie totale
Σ_k := la matrice diagonale composé des k premières valeurs singulières de Σ
U_k := les k premières colonnes de U
V_k := les k premières colonnes de V
retourner les matrices $U_k\ \Sigma_k^{1/2}$ et $\Sigma_k^{1/2}\ V_k^T$

Algorithm 3: matUS-matSV

Erreur RMSE sur la base vierge de ml-100k

Table 25 – Erreurs RMSE pour les méthodes naïves avec la base vierge

méthode	RMSE
random_unif	1.696
random_samp	1.575
meanOfMovies	1.216
meanOfUsers	1.128
mean	1.127
meanByUser	1.049
meanByMovie	1.033

TABLE 26 – Erreurs RMSE pour la méthode `RFP_weighted-centered&a` (knn) avec la base vierge

K	32	33	34	35	36	37	38	39	40	41	42
RMSE	0.952	0.952	0.952	0.952	0.953	0.953	0.953	0.952	0.952	0.952	0.952

TABLE 27 – Erreur RMSE pour la méthode SVD-Item avec la base vierge

τ	0.23	0.235	0.24	0.245	0.25	0.255	0.26
RMSE	0,97792	0,97947	0,97799	0,97665	0,97753	0,97815	0,98057

TABLE 28 – Erreur RMSE pour la méthode SVD par la descente de gradient classique avec la base vierge boucler 30 fois

λ	6.00	8.00	10.00	12.00	14.00	16.00
RMSE	0.9735	0.9468	0.9384	0.9383	0.9416	0.9503

TABLE 29 – Erreur RMSE pour la méthode SVD par la descente de gradient proximal avec la base vierge boucler 30 fois

λ	6.00	8.00	10.00	12.00	14.00	16.00
RMSE	0.9400	0.9324	0.9336	0.9420	0.9536	0.9663