Filtrage Collaboratif pour les Systèmes de Recommandation

Dioula DOUCOURE

Cassiopée, Télécom SudParis. Encadré par Yohan Petetin

Abstract. A recommender system aims at helping users find compelling contents that match their preferences in a large corpus. From e-commerce to online advertisement, recommendation systems are today part of our daily online lives. On YouTube for instance, more than 70 % of watch time comes from recommendation. This shows how important recommendation is to online businesses since being able to recommend the right items to users significantly increases customer retention and it also enhances user satisfaction. In this paper, we will focus on matrix completion in collaborative filtering, one of the most popular approach to build a recommender system. We will on the one hand implement a matrix factorization model applied to MovieLens dataset. On the other hand, we will study matrix completion via convex optimization.

Keywords: Collaborative filtering · Matrix factorization · Matrix completion · Alterning least square (ALS) · Stochastic gradient descent (SGD) · Nuclear norm · Convex Optimization

1 Introduction

Avec le développement des nouvelles technologies de l'information, nous nous retrouvons souvent submergés par une surabondance informationnelle. Se pose alors la question du filtrage de toutes ces informations afin d'en éliminer celles qui ne sembleraient pertinentes pour les utilisateurs. En effet, un utilisateur qui n'est équipé que d'outils de recherche ne peut faire face à tout ce flux d'informations. Les moteurs de recherche à l'instar de Google en sont une parfaite illustration. Lorsqu'un utilisateur formule son besoin sur google avec des mots clés, lui sont retournés des résultats plus ou moins pertinents et généralement la personne ne va pas au delà de la première page de navigation du moteur de recherche à cause d'une surchage d'informations. Ce problème va faire émerger de nouvelles techniques permettant non seulement de réduire le temps de recherche de l'utilisateur mais aussi lui proposer des ressources (articles, vidéos, produits e-commerce) en adéquation avec ses centres d'intérêt. Dès lors, il reçoit des suggestions auxquelles il n'aurait pas spontanément prêtées attention. On parle de recommandation.

Un **système de recommandation** a pour objectif de prédire l'intérêt que peut avoir un utilisateur pour un item (produit) afin de le lui recommander. Contrairement à la recherche d'informations, la spécificité d'un système de recommandation est de proposer des ressouces *pertinentes* aux utilisateurs. Ces systèmes

font partie de notre quotidien. Nous les retrouvons sur les commerces en ligne, les plateformes de streaming musical, sur Youtube, sur Netflix etc. Prenons l'exemple du géant américain Amazon. Ce dernier, grâce à son système de recommandation, suggère à ses clients des articles pouvant les intéresser, des produits similaires à ceux qu'ils ont déjà achetés ou mis dans leurs paniers. Plus de 35% des articles achetés sur amazon et plus 80% des contenus visionnés sur Netflix sont des recommandations [11]. Spotify est un autre exemple illustrant la présence des systèmes de recommandation dans notre quotidien. La plateforme propose à ses utilisateurs des playlists personnalisées de 30 chansons toutes les semaines. Des millions d'utilisateurs découvrent chaque lundi une playlist conçue par des algorithmes de recommandation.
La recommandation revêt donc d'une importance capitale pour les entreprises. Elle leur permet d'améliorer l'expérience utilisateur de leurs clients et constitue un moyen de fidélisation. Elle est par ailleurs un moyen très efficace pour augmenter le chiffre d'affaires. C'est ce qui poussera sans doute Netflix a lancé un challenge recommandation en 2007 en mettant un million de dollars en jeu [1].

2 La recommandation

Recommander c'est avant tout être capable de prédire si l'utilsateur sera susceptible d'aimer ou pas la ressource que l'on souhaiterait lui proposer. Ce problème de prédiction des préférences d'un utilisateur peut être considéré comme un problème d'optimisation[5]. Considérons U un ensemble d'utilisateurs et I l'ensemble constitué de documents que l'on souhaiterait recommander. Notons f la fonction qui va mesurer l'intérêt qu'un utilisateur u peut avoir pour un document i. En trouvant tous les documents maximisant la fonction f, on pourra alors déterminer ceux que u juge utiles et intéressants et les lui recommander. On peut tout simplement réecrire ce problème comme suit:

$$f : U \text{ x } I \longmapsto \text{R}$$
$$\forall\ u \in U,\ d_u = \arg\max_{i \in I} f(u, i)$$

L'objectif est donc d'estimer la fonction d'utilité d'un document i pour un utilisateur u [13].

3 Les différentes approches dans la recommandation

Dans son livre intitulé *Recommender Systems*[2], Charul Aggarwal résume un bon système de recommandation en quatre points:

- la *pertinence* des articles recommandés

- la *nouveauté*: la recommandation aura plus de sens si l'utilisateur n'a jamais interagit avec la ressource qu'on lui suggère
- la *sérendipité* qui consiste à recommander des articles inattendus. Aggarwal donne l'exemple suivant pour illustrer ce point. "Si un nouveau restaurant indien ouvre dans un quartier, alors la recommandation de ce restaurant à un utilisateur qui mange normalement de la nourriture indienne est nouvelle mais pas nécessairement fortuite. Lorsque le même utilisateur se voit recommander de la nourriture éthiopienne, et que l'utilisateur ne savait pas qu'un tel aliment pourrait lui plaire, alors la recommandation est fortuite"
- la *diversité* dans les articles recommandés

Il existe différentes approches des systèmes de recommandation. Elles se fondent sur des facteurs tels que le profil de l'utilisateur, la connaissance des ressources à recommander, les similarités entre profils de différents utilisateurs etc.
On peut classer les approches en trois grandes catégories: les recommandations basées sur le contenu, le filtrage collaboratif et enfin les approches hybrides.

3.1 La recommandation basée sur le contenu (content-based)

Dans les approches basées sur le contenu, on recommande à un utilisateur des ressources qui sont similaires à celles qu'il a déjà appréciées dans le passé. Ces approches requièrent la connaissance du profil de l'utilisateur. On utilise le plus souvent les évaluations (une note sur une échelle de 1-5) que l'utilisateur a fait de certains items pour lui en recommander d'autres similaires. Une fois le profil mis en place, on caractérise les documents par leurs attributs. Si nous prenons l'exemple des films sur Netflix, les attributs peuvent être: le genre du film, le réalisateur, l'acteur etc. A partir de la connaissance de ces deux facteurs, on définit un score de similarité.

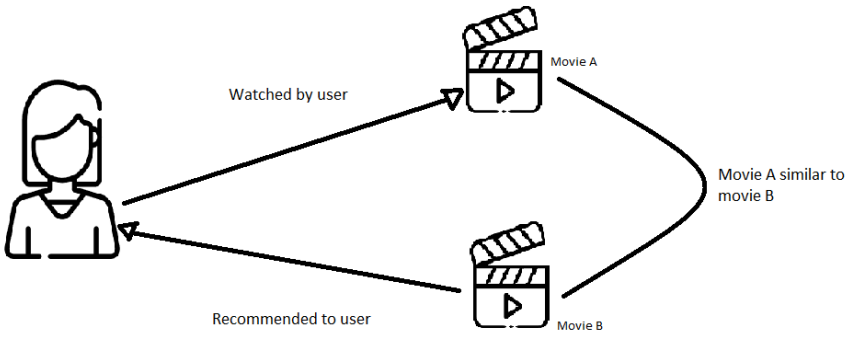


Fig. 1: Recommandation basée sur le contenu

La recommandation basée sur le contenu a l'avantage de ne nécessiter que la connaissance de l'utilisateur. Son fonctionnement ne requiert pas une grande communauté d'utilisateurs. Cependant, elle souffre d'un certain nombre de limites. En effet, étant donné que la recommandation ne se base que sur le profil, on a très souvent une redondance thématique dans les documents recommandés car cette approche est incapable de proposer à l'utilisateur des ressources différentes de celles qu'il a déjà évaluées. Aussi, pour que les propositions soient pertinentes, il faut que l'utilisateur ait évalué un certain nombre de documents, ce qui n'est pas le cas des nouveaux utilisateurs.

3.2 Recommandation basée sur le filtrage collaboratif

Contrairement aux approches basées sur le contenu, dans le filtrage collaboratif, on recommande à un utilisateur des items que d'autres utilisateurs similaires ont appréciés dans le passé. Cette méthode ne se base que sur la proximité des profils des différents utilisateurs pour effectuer une recommandation. Ces profils sont représentés par l'historique des avis des utilisateurs sur des ressources. Cela peut prendre la forme d'un achat, d'une note, d'un simple partage (d'un article) etc. Par exemple, s'il s'avère que Hatem et Atman ont tous deux acheté les produits A et B. Si Atman achète en plus des deux précédents, un nouveau produit C. Etant donné que Hatem et Atman ont acheté 2 produits identitiques, il est donc très probable qu'ils partagent certaines préférences. Le système de recommandation estime alors que Hatem acheterait probablement le produit C s'il l'avait vu. Il le lui recommande.

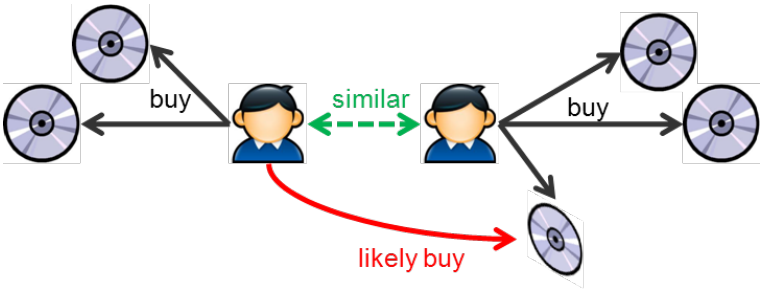


Fig. 2: Source: https://dzone.com/articles/recommendation-engine-models

L'un des avantages du filtrage collaboratif est la diversité thématique dans la recommandation contraitement à la méthode basée sur le contenu. Ici, l'utilisateur se voit proposer des ressources aussi différentes les unes des autres. Le filtrage peut être appliqué à tout type de documents. Par ailleurs, cette méthode est très flexible. Elle permet de prendre rapidement en compte les changements de goût des utilisateurs.

En revanche, un des inconvénients du filtrage collaboratif est le problème très connu du *démarrage à froid*. En effet, un nouvel utilisateur n'ayant évalué aucun document ne peut recevoir de recommandations avec cette méthode. De même, les nouveaux documents qui n'ont pas été évalués sont difficillement recommandés aux utilisateurs car les documents ne sont décrits que par les notes qui leur ont été attribuées. Pour résoudre ce problème de démarrage à froid se manifestant donc par une absence d'informations sur les nouveaux documents/utilisateurs, on combine la recommandation basée sur le contenu au filtrage collaboratif. On parle alors de système de recommandation hybride.

3.3 Méthode hybride

Les sytsèmes de recommandation hybrides ont pour but de palier les limites des deux méthodes précédentes en les combinant. C'est ce système de recommandation que Netflix utilise pour les suggestions de films. Il prend en compte les élémens suivants:

- l'historique des contenus visionnés et des évaluations des différents films
- les profils des utilisateurs et leurs similarités
- les attributs des films

4 Notre approche

Dans cette étude, nous nous intéresserons à la recommandation basée sur le filtrage collaboratif. Nous étudierons deux méthodes très utilisées que sont la factorisation de matricielle (décomposition en deux matrices de facteurs latents) et la complétion matricielle par optimisation convexe.

5 Factorisation matricielle

5.1 Les matrices dans la recommandation

Nous avons vu que les systèmes de recommandation se décomposent en deux principales tâches: une prédiction des notes qu'un utilisateur est susceptible de donner à un item (film, livres, article etc) et la suggestion des éléments qui pourraient l'intéresser. Dans les approches de type filtrage collaboratif, ces évaluations sont regroupées dans des matrices de type utilisateur/produit où chaque ligne représente un utilisateur et chaque colonne un item (produit). On les appelle *rating matrix* (matrices d'évaluation). Elles sont non seulement de grandes tailles (des milliers voire des millions d'utilisateurs/produits) mais elles ont aussi la particularité d'être *sparse* [10] (creuses). En effet, seul un nombre assez faible d'items se retrouvent notées ce qui fait qu'une grande partie des entrées de la matrice sont inconnues. Pour remédier à ce problème et réduire la dimensionnalité des matrices d'évaluations, on les décompose en produits de matrices. On parle de factorisation matricielle.

Fig. 3: Exemple de matrice d'évaluations

5.2 Notation et formulation générale

Supposons que nous disposons d'une matrice R de $\mathcal{M}_{m,n}(\mathbb{R})$ dont les entrées r_{ui} sont les évaluations des films sur Netflix et où chaque ligne représente un utilisateur et chaque colonne un film. L'objectif de la factorisation matricielle est de trouver deux matrices de facteurs latents dont le produit est égal à la matrice R. On apprend donc un modéle latent d'utilisateurs $X \in \mathrm{M}_{m,d}(\mathbb{R})$ et d'items $Y \in \mathrm{M}_{d,n}(\mathbb{R})$ afin que le produit $\hat{r}_{ui} = x_u^T \cdot y_i$ estime l'entrée r_{ui} de la matrice R [8]. Le nombre de variables latentes est égal à d. Ces facteurs latents sont en général les attributs des items. Dans le cas des films, il est évident que les utilisateurs préferent certains genres, acteurs ou encore réalisateurs. Ces catégories représentent les facteurs latents. Deux utilisateurs fans d'un même réalisateur ou qui apprécient un genre en particulier aiment vraisemblement des films similaires. Pour un utilisateur, un attribut peut être par exemple un nombre indiquant à quel point il aime les films de comédie et pour un film, un nombre indiquant son caractère humoristique (par exemple 1 s'il s'agit d'une comédie et 0 sinon). Par conséquent, les entrées de X traduiront la force de préférence des utilisateurs pour les attributs choisis (genre du film dans notre exemple). Ces informations nous permettront de prédire les notes manquantes dans la matrice et donc de recommander les films non évalués.

5.3 Un peu de maths

Pour déterminer les variables latentes X et Y, on minimise la fonction objective suivante:

$$\mathcal{L} = \sum_{(u,i)\in\Gamma} (r_{ui} - x_u^T \cdot y_i)^2 \tag{1}$$

Le terme $\hat{r}_{ui} = x_u^T \cdot y_i$ représente la prédiction de la note de l'item i par l'utilisateur u. Γ est l'ensemble des (u,i) telle que r_{ui} est connue.

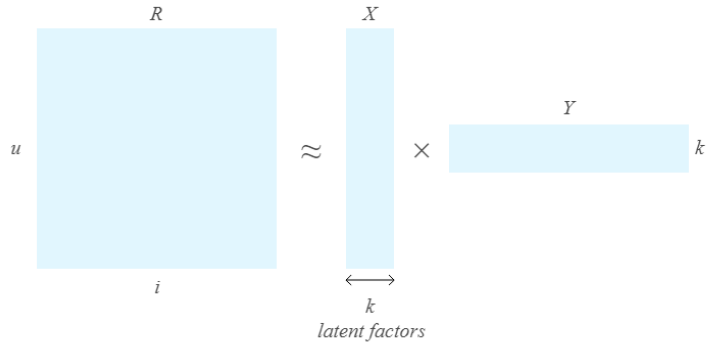


Fig. 4: Factorisation matricielle

Afin d'éviter le surapprentissage, on ajoute à la fonction objective précédente des termes de régularisation. L'égalité précédente se réécrit de la manière suivante:

$$\mathcal{L} = \sum_{(u,i)\in\Gamma} (r_{ui} - x_u^T \cdot y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2) \tag{2}$$

La constante de régularisation λ est un hyperparamètre. On peut la déterminer avec une validation croisée.

Pour minimiser la fonction $\mathcal{L}$, nous allons utiliser deux algorithmes: la méthode des moindres carrés alternés (Alterning Least Square, ALS) et la descente de gradient stochastique (SGD).

6 Algorithmes d'optimisation

6.1 Moindres carrés alternés

La méthode des moindres carrés alternés consiste à fixer un des facteurs latents (X ou Y) et à calculer le gradient par rapport l'autre facteur. En annulant le gradient, on trouve la règle de mise à jour d'un des facteurs. En refaisant la même opération, on détermine aisément l'autre facteur. Essayons de dériver la fonction objective. Fixons y_i:

$$\frac{\partial \mathcal{L}}{\partial x_u} = -2 \sum_i (r_{ui} - x_u^T \cdot y_i) y_i^T + 2\ \lambda\ x_u^T$$

$$\lambda x_u^T - (r_u - x_u^T Y^T) Y = 0$$

$$r_u Y = x_u^T (Y^T Y + \lambda I)$$

On en déduit que:

$$\boxed{x_u^T = r_u Y (Y^T Y + \lambda I)^{-1}} \tag{3}$$

En fixant x_i, le même raisonnement nous permet d'écrire:

$$\boxed{y_i^T = r_i X (X^T X + \lambda I)^{-1}} \tag{4}$$

Pour chaque itération de l'algorithme, on fixe un des facteurs et on calcule l'autre. On définit ainsi une règle de mise à jour des facteurs en répétant ces itérations jusqu'à ce qu'il y ait convergence.

Algorithm 1 Moindres carrés alternés

Initialiser X et Y

1: **repeat**
2: **for** $u = 1 \dots \mathrm{m}$ **do**

$$x_u^T = r_u Y (Y^T Y + \lambda I)^{-1}$$

3: **end for**
4: **for** $i = 1 \dots \mathrm{n}$ **do**

$$y_i^T = r_i X (X^T X + \lambda I)^{-1}$$

5: **end for**
6: **until** convergence

6.2 Descente de gradient stochastique (SGD)

La minimisation de la fonction objective (2) par la descente de gradient stochastique a été popularisée par Simon Funk [7]. Comme dans l'algorithme précédent, on commence tout d'abord par initialiser les matrices de facteurs latents X et Y. On définit un ensemble d'entraînement sur lequel on passe en revue les évaluations de la matrice de départ. À chaque itération, on évalue l'erreur de prédiction e_{ui} définie par:

$$e_{ui} = r_{ui} - \hat{r}_{ui}$$

On met à jour les variables latentes x_u et y_i dans la direction opposée au gradient avec un pas γ.

- $x_u \longleftarrow x_u + \gamma \, (e_{ui} y_i - \lambda x_u)$
- $y_i \longleftarrow y_i + \gamma \, (e_{ui} x_u - \lambda y_i)$

Les biais dans la factorisation matricielle

Certains films ont tendance à recevoir plus de bonnes notes que d'autres en raison de leur popularité. On constate aussi que certains utilisateurs sont plus généreux que d'autres dans la notation des films. Prenons l'exemple de deux utilisateurs A et B. Admettons que la moyenne des notes de A soit égale à 1.5 et celle de B à 4.7. Une note de 3 attribuée à un nouveau film par l'utilisateur A

n'a pas la même signification qu'un 3 de l'utilisateur B. A a sûrement apprécié le nouveau film contrairement à B. C'est la raison pour laquelle on introduit des biais qu'on associe aux utilisateurs et aux films. La prise en compte de ces biais augmente considérablement la pertinence des items recommandés. Les vainqueurs du challenge netflix ont d'ailleurs pris en compte ces biais dans leur algorithme [9].

Le biais b_{ui} associé à la note r_{ui} est défini par:

$$b_{ui} = \mu + b_u + b_i$$

- μ est la moyenne des évaluations
- b_u biais associé aux utilisateurs
- b_i biais associé aux films

Dès lors, la prédiction de la note de l'item i par l'utilisateur u se réecrit:

$$\hat{r}_{ui} = x_u^T y_i + b_u + b_i + \mu$$

La fonction objective à minimiser est donc:

$$\mathcal{L} = \sum_{(u,i)\in\Gamma} (r_{ui} - x_u^T \cdot y_i - b_u - b_i - \mu)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2 + b_u^2 + b_i^2) \qquad (5)$$

En prenant en compte les biais, les mises à jour à effectuer dans le cas de la descente de gradient stochastique sont les suivantes:

- $x_u \longleftarrow x_u + \gamma\ (e_{ui} y_i - \lambda x_u)$
- $y_i \longleftarrow y_i + \gamma\ (e_{ui} x_u - \lambda y_i)$
- $b_u \longleftarrow b_u + \gamma\ (e_{ui} - \lambda b_u)$
- $b_i \longleftarrow b_i + \gamma\ (e_{ui} - \lambda b_i)$

7 Complétion matricielle par optimisation convexe

7.1 Préliminaires

Avant de continuer, résumons dans les quelques lignes qui vont suivre les notations utilisées dans cette partie.

- $\|\|\|_{op}$ la norme opérateur d'une matrice définie par $\|M\|_{op} = \sup_{\|x\|_2 \leq 1} \|Mx\|_2$
- $rang(M)$ le rang de la matrice M
- $\Gamma = \{(i,j), \quad m_{ij} \quad connue\}$ de cardinal m
- $M = U\Sigma V^T = \sum_{k=1}^{r} u_k \sigma_k v_k^T$ la décomposition en valeurs singulières de la matrice M de rang r où les u_k et v_k sont les vecteurs singuliers de M
- $\|\|\|_*$ la norme nucléaire définie par: $\|M\|_* = \sum_{k=1}^{r} \sigma_i$ où les σ_i sont les valeurs singulières de la matrice.

7.2 Complétion des matrices de rang faible

Nous avons vu que dans les bases de données de recommandation, le nombre d'observations est largement inférieur au nombre total d'entrées (des matrices d'évaluations). Reconstruire directement ces matrices sans réduire la dimension du problème est une tâche difficile. Dans la méthode de factorisation matricielle, on les décomposait en produits de deux matrices de facteurs latents afin de reduire cette dimension. Dans la complétion matricielle en générale, on fait l'hypothèse de rang faible, ce qui diminue considérablement la dimension du problème et permet de reconstruire notre matrice inconnue. Considérons par exemple la matrice M de $\mathcal{M}_n(\mathbb{R})$ de rang r. Nous avons $2(n-r)r$ dégrés de liberté. Lorsque M est de rang faible, le nombre de degrés de liberté vaut $2nr$ et la dimensionnalité est réduite d'un facteur $\frac{n}{2r}$.

La formulation mathématique du problème de complétion matricielle sous l'hypothèse du rang faible est la suivante:

$$\begin{cases} \min\limits_{X} & rang(X) \\ s.c & X_{i,j} = M_{i,j} \quad \forall (i,j) \in \Gamma \end{cases} \tag{6}$$

Une manière naïve de résoudre le problème (5) est de recourir à une stratégie de recherche combinatoire [12]. On suppose d'abord que M est de rang 1. On a alors pour toutes colonnes m_i et m_j de M la relation: $m_i = \lambda_{i,j} m_j$ (les colonnes sont linéairement dépendantes) avec $\lambda_{i,j} \in \mathbb{R}$. Si le système n'a pas de solution pour l'hypothèse de rang 1, on suppose alors que M est de rang 2. Le nouveau système à résoudre est donc: $m_i = \lambda_{i,j} m_j + \lambda_{i,k} m_k$. On répète ainsi la procédure jusqu'à ce qu'on trouve la solution. Cette méthode de résolution a une compléxité de l'ordre de l'exponentiel de la dimension de la matrice M [4]. Par ailleurs, le problème (5) n'est pas convexe et est NP-difficile [6].

Candès et Recht [3] ont montré qu'avec un certain nombre *d'hypothèses* que nous énoncerons un peu plus loin, le problème (5) admet une unique solution X. De plus, ils ont proposé une relaxation convexe de (5) [14]. Il s'agit de la minimisation de la norme nucléaire.

$$\begin{cases} \min\limits_{X} & \|X\|_* \\ s.c & X_{i,j} = M_{i,j} \quad \forall (i,j) \in \Gamma \end{cases} \tag{7}$$

Candès et Recht ont par ailleurs introduit la notion de *cohérence* comme garantie de reconstruction des matrices de rang faible. En effet, toutes les matrices de rang faible ne peuvent être reconstruites. Considérons la matrice M de rang 1 ci-dessous,

$$M = \begin{bmatrix} 0\,0\ldots 0\,1 \\ 0\,0\ldots 0\,0 \\ \vdots\,\vdots\ \vdots\ \vdots\,\vdots \\ 0\,0\ldots 0\,0 \end{bmatrix}$$

toutes les entrées de M sont nulles (inconnues) à l'exception d'une, valant 1. Il nous est donc impossible de prédire les valeurs nulles à moins de les avoir toutes observées. On s'aperçoit alors que la connaissance de la seule entrée non nulle ne donne pas d'informations sur les autres entrées. C'est ce à quoi répond la notion de cohérence.

Cohérence

Soient U une famille de vecteurs orthonormés de $\mathbb{R}^n$ de dimension r et P_U la projection orthogonale sur U. La cohérence de U est définie par:

$$\mu(U) = \frac{n}{r} \max_{1 \leq i \leq n} \|P_U e_i\|^2$$

où $(e_i)_{1 \leq i \leq n}$ est la base canonique de $\mathbb{R}^n$.

Lorsque les vecteurs lignes et colonnes d'une matrice ont une cohérence faible, chaque entrée renseigne sur la même quantité dinformations ce qui nous permet de reconstruire la matrice de départ.

Résultats sur la Cohérence

R_1: Les vecteurs lignes et colonnes U et V de la matrice $M = U\Sigma V^T$ vérifient: $\max(\mu(U), \mu(V)) \leq \mu_0$ avec $\mu_0 > 0$

R_2: La matrice $M = \sum_{k=1}^{r} u_k v_k^T$ de $\mathcal{M}_{n_1,n_2}(\mathbb{R})$ vérifie: $\|UV^T\|_{op} \leq \mu_1 \sqrt{\frac{r}{n_1 n_2}}$ où μ_1 est un réel positif.

Avec ces résultats, on peut énoncer le théorème principal de Candès qui permet de reconstruire notre matrice d'évaluations lorsque ses colonnes et ses lignes sont faiblement cohérentes.

Theorem 1. *Soit M une matrice de $\mathcal{M}_{n_1,n_2}(\mathbb{R})$ de rang r. On pose $n = \max(n_1, n_2)$. Supposons que M vérifie R_1 et R_2 et que l'échantillonnage de m observations de ses entrées est uniforme et est identiquement distribué. Il existe alors des constantes C et c telles que*

$$m \geq C \max(\mu_1, \mu_0^{\frac{1}{2}} \mu_1, \mu_0 n^{\frac{1}{4}}) nr(\beta \log n)$$

Pour un certain $\beta > 2$, le problème (6) admet une unique solution égale à M avec une probabilité $p \geq 1 - cn^{-\beta}$

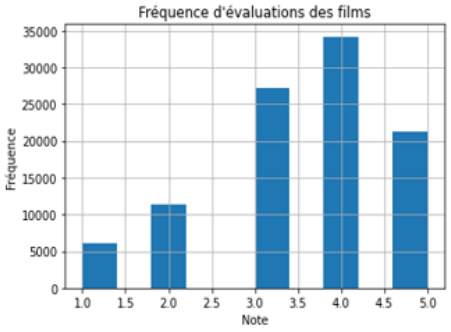
8 Application à la base de données de films MovieLens

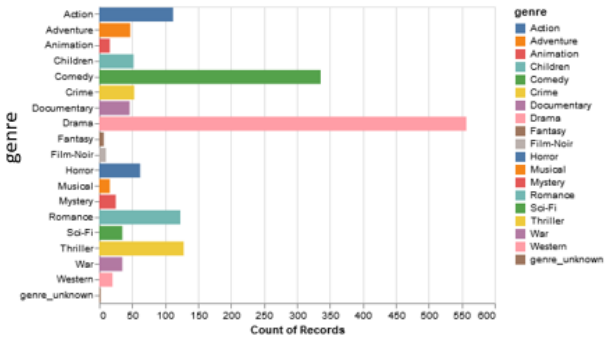
Nous allons appliquer la méthode de factorisation matricielle sur la base de données de films de MovieLens. Cette base de données contient 100000 notes attribuées par 943 utilisateurs à 1682 films. Chaque utilisateur a évalué au moins à une vingtaine de films.

	title	rating count	rating mean
49	Star Wars (1977)	583	4.358
257	Contact (1997)	509	3.804
99	Fargo (1996)	508	4.156
180	Return of the Jedi (1983)	507	4.008
293	Liar Liar (1997)	485	3.157
285	English Patient, The (1996)	481	3.657
287	Scream (1996)	478	3.441
0	Toy Story (1995)	452	3.878

	user_id	movie_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596
5	298	474	4	884182806

Chaque film a des attributs qui lui sont propres. Comme nous l'avons déjà évoqué, ces attributs sont très utiles pour effectuer une recommandation. Les films et les utilisateurs sont identifiés par leurs identifiants respectifs comme dans la figure précédente. Nous utiliserons ces identifiants pour créer notre matrice d'évaluations R de $\mathcal{M}_{943,1682}(\mathbb{R})$.





Seul 6,3% des entrées de la matrice sont connues. Cette sparsity se calcule en divisant simplement le nombre d'entrées observées (100000 pour la base de données 100k MovieLens) par le nombre total d'entrées de la matrice ($943 * 1682$)

```
array([[ 5.,  3.,  4., ..., nan, nan, nan],
       [ 4., nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       ...,
       [ 5., nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan,  5., nan, ..., nan, nan, nan]])
```

Fig. 5: Matrice d'évaluation

Pour stocker les données creuses, nous avons utilisé les matrices du module *Sparse* de *Scipy*. Elles sont plus adaptées que les matrices numpy car permettant de manipuler plus efficacement les matrices sparses. Pour la complétion matricielle par optimisation convexe, nous avons utilisé la bibliothèque **fancyimpute**. Elle est téléchargeable à l'adresse suivante:`https://pypi.org/project/fancyimpute/`

9 Résultats

9.1 Descente de gradient stochastique

Comme nous nous y attendions, avec l'algorithme de descente de gradient stochastique, les paramètres de notre modèle s'affinent au fil du temps. Nous avons choisi au hasard au début ces paramètres. L'une des premières observations que nous avons faites est l'influence du nombre de facteurs latents sur notre modèle. Avec des facteurs latents de tailles différentes, on a constaté que l'erreur de prédiction variait considérablement.

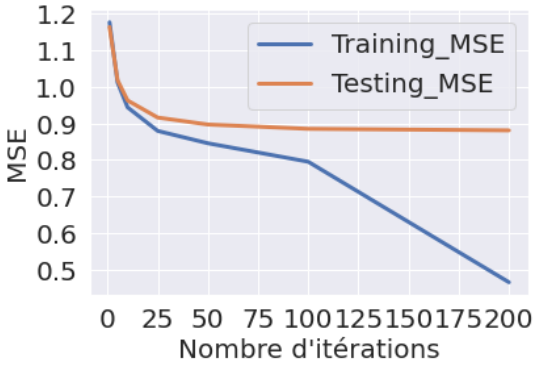


Fig. 6: Factorisation matricielle avec la descente de gradient stochastique

Nous avons observé qu'un très grand nombre de facteurs latents conduit à un phénomène de sur-apprentissage (Fig.7). Plus on augmente les facteurs latents plus notre modèle se spécialise sur les données de notre ensemble d'entrainement et il le fait si bien qu'il se généralise très mal. En réduisant le nombre de variables latentes, le sur-apprentissage diminue et disparait pour certaines valeurs.
Pour déterminer les meilleurs paramètres de notre modèle, nous avons effectué une validation croisée sur les listes suivantes:

Variables latentes	5	10	20	40	50	80	100	150	200
Régularisation (λ)	0.001	0.002	0.01	0.02	0.1	0.2	0.25	0.5	1

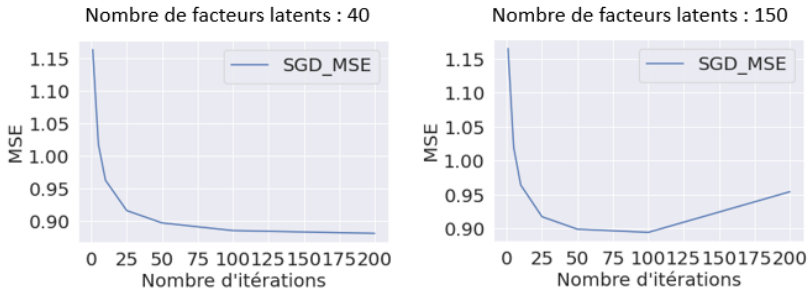


Fig. 7: Evolution MSE en fonction du nombre d'itératons

Nous avons trouvé que le nombre de facteurs latents qui optimise le mieux notre modèle est égal à 40. Le paramètre de régularisation trouvé lors de la validation croisée vaut 0.01 avec un pas de gradient $\gamma = 0.001$.
Le nombre de variables latentes que nous avons trouvé dans la descente de gradient stochastique est cohérent avec les résultats obtenus notamment lors du challenge Netflix de 2009. En effet, lorsque la base de données n'est pas trop large, il est plus judicieux de choisir un nombre de variables latentes pas trop grand ("Lower is better"). Un grand nombre de facteurs n'est meilleur que lorsque les données d'apprentissage deviennent grandes.

Nous avons trouvé un score RMSE égal à **0.88**. À titre indicatif, celui de Netflix lors du challenge de 2007 était de l'ordre de 0.95.

```
[[ 5.24681419  3.44999746  2.93827376 ...,  4.93948659  4.20981051
   5.90208862]
 [ 3.45890691  2.88852161  1.94166001 ...,  3.45883726  2.87506586
   3.92243295]
 [ 3.28104943  2.11192302  2.53980984 ...,  3.09046697  2.66709054
   3.67397603]
 ...,
 [ 2.79261176  2.11598226  2.5189713  ...,  2.36535702  1.94035837
   2.67286888]
 [ 2.86420341  1.8404591   2.10479273 ...,  2.83349656  2.49764757
   3.71285494]
 [ 2.82295066  2.46283972  2.08517971 ...,  2.90593869  2.04198094
   3.28897811]]
```

Fig. 8: Matrice d'évaluations après complétion

9.2 Moindres carrés alternés

Comme pour la descente de gradient stochastique, nous avons essayé de jouer sur certains paramètres clés de notre problème afin de déterminer leur influence sur le modèle. Nous arrivons à une conclusion similaire avec la méthode des moindres carrés alternés. Nous retenons que plus le nombre de facteurs latents est important, plus le modèle va être complexe et plus le temps de calcul (dérivation des facteurs latents pour les utlisateurs et les films) augmentera. Sur la Fig.9 par exemple, avec un nombre de facteurs latents égal à 40, on constate que le MSE de notre ensemble d'entrainement est deux fois plus petit que celui de l'ensemble de validation. Pour réduire cet écart, nous avons fait varier le terme de régularisation et avons aussi modifié le nombre de facteurs latents avec une validation croisée comme dans la descente de gradient stochastique.

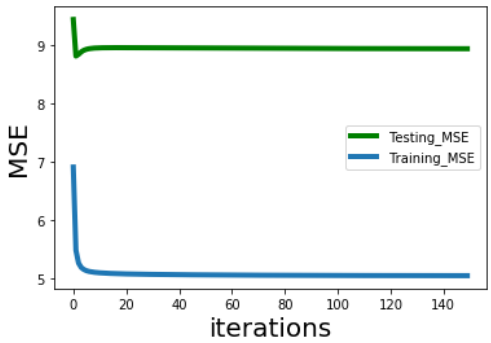


Fig. 9: Moindres carrés alternés: 40 facteurs latents, $\lambda = 0.1$

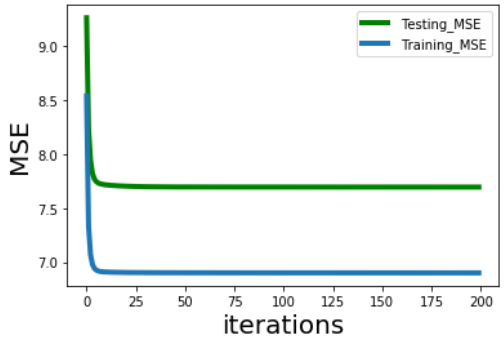


Fig. 10: Moindes carrés alternés: 10 facteurs latents, $\lambda = 0.01$

Nous trouvons que les paramètres qui opitmisent notre modèle sont: 10 facteurs latents et $\lambda = 0.01$ avec un score MSE de **8.011**.

Outre le nombre de facteurs latents et le terme de régularisation, un autre paramètre qui influe sur la performance de notre modèle aussi bien avec la descente de gradient stochastique qu'avec la méthode des moindres carrés alternés est le nombre d'itérations. Les observations faites en sus de la validation croisée

ont montré que nous obtenons de meilleurs résultats avec des itérations comprises entre 100 et 200.

ALS vs SGD

Nos résultats nous ont amenés à la conclusion suivante: la factorisation matricielle par la descente de gradient stochastique (SGD) est plus performante sur la base de données 100k de MovieLens que celle avec la méthode des moindres carrés alternés (ALS), ce qui pourrait s'expliquer par le fait qu'on ait pris en compte les différents biais dans la descente de gradient stochastique. La recommandation avec la SGD est donc beaucoup plus pertinente et les notes prédites par celle-ci pour un utilisateur sont cohérentes avec le profil de ce dernier comme on peut le voir sur la figure ci-dessous.

Films regardés par l'utilisateur 50

	Notes	Prédiction	Titre
171	5.000	3.893	Empire Strikes Back, The (1980)
180	5.000	3.814	Return of the Jedi (1983)
172	5.000	3.714	Princess Bride, The (1987)
49	5.000	3.834	Star Wars (1977)
143	5.000	3.884	Die Hard (1988)
82	5.000	3.823	Much Ado About Nothing (1993)

Films susceptibles de l'interesser

	Notes	Prédiction	Titre
1270	0.000	4.255	North (1994)
1057	0.000	4.253	War, The (1994)
1497	0.000	4.252	Farmer & Chase (1995)
850	0.000	4.235	Two or Three Things I Know About Her (1966)
1658	0.000	4.220	Getting Away With Murder (1996)
1480	0.000	4.215	S.F.W. (1994)

Fig. 11: SGD Recommendation

Une autre différence entre ALS et la SGD est que le premier converge plus rapidement. La descente de gradient stochastique est très lente surtout lorsque nous avons affaire à de grands jeux de données. Les moindres carrés alternés ont une complexité cubique, par conséquent plus grande que celle de la SGD. On ne peut cependant pas conclure sur l'efficacité des recommandations basées sur la descente de gradient stochastique sur celles basées sur la méthode des moindres carrés alternés d'une manière générale. En effet, si nous prenons l'exemple des données implicites (clics sur un article etc), il a été démontré que l'ALS est plus performant que la SGD.

9.3 Optimisation convexe

La bibliothèque fancyimpute que nous avons évoquée dans la partie sur la complétion par optimisation convexe utilise d'une part la programmation sémi-définie (SDP) pour résoudre le problème (7) notamment à l'aide du solver SDPT3 que l'on peut retrouver ici: `https://github.com/SQLP/SDPT3`. D'autre part, fancyimpute utilise le module CVXPY de python pour la minimisation de la

norme nucléaire. On peut retrouver ce package à l'adresse suivante: `https://www.cvxpy.org`. Cette dernière méthode est très lente, beaucoup plus lente que la SGD. Nous en avons fait les frais.

Le score MSE trouvé avec cette méthode avec notre base de données est de **0.86**. Nous trouvons une valeur inférieure à celle obtenue avec la descente de gradient stochastique. Sur le jeu de données 100k de MovieLens, la conclusion que nous pouvons faire à ce stade est que la factorisation par la descente de gradient stochastique est plus efficace que les moindres carrés alternés et la complétion matricielle par optimisation convexe.

10 Conclusion

Nous avons étudié dans ce projet les différentes approches des systèmes de recommandation et avons souligné leur importance pour les entreprises. Nous avons par ailleurs vu que la recommandation c'est avant tout la prédiction de l'intérêt qu'un utilisateur peut avoir pour un item (article, films, vidéos, livres etc). L'approche sur laquelle nous avons travaillé est la complétion matricielle pour les recommandations basées sur le filtrage collaboratif. Les algorithmes que nous avons implémentés pour construire notre système de recommandation nous ont fournis des résultats assez concluants. Une perspective intéressante pourrait être une étude comparative entre la factorisation matricielle et les modèles probablistes Restricted Boltzmann Machine (RBM) pour la prédiction des données manquantes dans les matrices d'évaluations. Les contraintes liées au temps ne nous ont malheureuement pas permis de faire cette étude.

11 Remerciements

Le travail effectué dans ce projet n'aurait pas été possible sans mon encadrant, M. Petetin Yohan. Je tiens à le remercier pour la confiance qu'il m'a accordée en me proposant ce projet et en acceptant de m'encadrer et pour tous ses conseils. Ce projet a été source abondante de découvertes et d'apprentissage et a été très formateur.

12 Avec des images c'est encore mieux

Fig. 12: Recommandation par le filtrage collaboratif

References

1. Netflix challenge. online: `https://www.netflixprize.com/` (2007)
2. Aggarwall, C.: Recommender Systems, The Textbook. Springer (2016)
3. Candès, E, J., Recht, B.: Exact Matrix Completion Via Convex Optimisation. Found.Comput. Math, 9(6):717–772 (2009)
4. Chistov, A, L., Grigoriev, D, Y.: Complexity of quantifier elimination in the theory of algebraically closed fields. In Proceedings of the 11th Symposium on Mathematical Foundations of Computer Science,, volume 176 of Lecture Notes in Computer Science, pages 17-31. Springer Verlag (1984)
5. Delporte, J.: Factorisation Matricielle, Application à la Recommandation Personnalisée de Préférences. Machine Learning [stat.ML], INSA de Rouen (2014)
6. Fazel, M.: Matrix rank minimization with applications. Ph.D. dissertation, Elec. Eng. Dept., Standford Univ., Stanford, CA (2002)
7. Funk, S.: Netflix update: Try this at home (2006), `http://sifter.org/simon/journal/20061211.html`
8. Koren, Y., Bell, R., Volinskiy, C.: Matrix factorization techniques for recommender systems. IEEE Computer Society, (0018-9162) :42–49 (2009)
9. Koren, Y.: The bellkor solution to the netflix grand prize
10. Koren, Y.: Factorization meets the neighborhood : A multifaceted collaborative filtering model. Proc. 14th ACM SIGKDD Int'l Conf, Knowledge Discovery and Data Mining, ACM (2008)
11. MacKenzie, I., Meyer, C., Noble, S.: How retailers can keep up with consumers (2013), `http://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers`
12. Nguyen, L, T., Kim, J., Shim, B.: Low-Rank Matrix Completion: A Contemporary Surve. Information System Laboratory, Department of Electrical and Computer Engineering, Seoul National University (2019)
13. Oufaida, H., Nouali, O.: Le filtrage collaboratif et le web 2.0, `https://www.cairn.info/revue-document-numerique-2008-1-page-13.htm?contenu=article`
14. Recht, B.: A simpler approach to matrix completion. Journal of Machine Learning Research, 12:3413–3430, December (2011)