UNIVERSITÉ DU QUÉBEC À MONTRÉAL

PRÉVISIONS BASÉES SUR LES RANGS ET SYSTÈMES DE RECOMMANDATION

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN MATHÉMATIQUES

PAR

CHARLES NOËL

FÉVRIER 2019

REMERCIEMENTS

Merci à mes parents, Jocelyne et Jean-Claude, mes soeurs, Bérénice et Camille, et mon frère Philippe pour leurs encouragements incessants.

Merci à mes amis et coéquipier des Citadins, Adèle, Agnès, Caroline, Catherine, Éléonore, Max, Vincent et les autres avec qui j'ai découvert ma passion pour la course à pied. Sans oublier les longues soirées d'études, les brunchs sportifs et les courses sous la pluie.

Merci à mes collègues de maîtrise, Jean-Mathieu, Nourredine, Michael et bien sûr Dominic. Merci également à Olivier et Roland pour les lectures et les remarques constructives.

Je remercie mes professeurs Alain Desgagné et Glenn Shorrock, qui par leurs enseignements m'ont inspiré à continuer mon parcours académique.

Finalement, j'aimerais remercier mon directeur Simon Guillotte pour sa patience, ses conseils, son enseignement et sa disponibilité.

TABLE DES MATIÈRES

LISTE DES ABRÉVIATIONS

BBR Prévision Bayésienne de Rangs Bivariés

BPMF Factorisation de Matrice Probabiliste Bayésienne

E-M Espérance-Maximisation

FC Filtrage Collaboratif

MAP Maximum a posteriori

MCMC Monte-Carlo par chaînes de Markov

NMF Factorisation de Matrice Non-négative

NPCA Analyse en Composantes Principales Non-paramétrique

PCA Analyse en Composantes Principales

PMF Factorisation de Matrice Probabiliste

PPCA Analyse en Composantes Principales Probabiliste

PREA Outil Personnalisé de Recommandation d'Algorithmes

RegSVD Régularisation de la Décomposition en Valeurs Singulières

SVD Décomposition en valeurs singulières

LISTE DES PSEUDO-CODES

LISTE DES FIGURES

RÉSUMÉ

Dans ce mémoire, nous présenterons un nouvel algorithme de prévision basé sur les rangs et à des fins de comparaison plusieurs algorithmes de système de recommandation de la famille du filtrage collaboratif. Nous utiliserons, entre autres, des notions d'algèbre matricielle, des modèles stochastiques, l'algorithme du gradient, l'algorithme E-M et des méthodes de Monte-Carlo par chaînes de Markov.

Nous comparerons ensuite ces algorithmes en faisant des simulations sur la base de données MovieLens et nous utiliserons la distance de Kendall comme statistique de comparaison. Nous chercherons à vérifier si la technique utilisant des copules se démarque des autres.

Mots clés : systèmes de recommandation, filtrage collaboratif, factorisation de matrice, copule bayésienne, rangs, MCMC, simulations, MovieLens, PREA

INTRODUCTION

Les systèmes de recommandation sont des outils qui servent à proposer des produits à un client selon ses goûts et ce de manière automatique. Ils sont maintenant utilisés dans plusieurs sphères de notre quotidien comme les achats en ligne ou la consommation de médias, tels des films et de la musique. On utilise ces outils aussi bien de manière commerciale que dans le cadre de recherches universitaires. (Resnick et Varian, 1997)

L'estimation de la performance empirique de différents systèmes de recommandation constitue la première étape d'une implantation commerciale. Le choix des statistiques de comparaison doit être fait judicieusement car il pourra faire pencher la balance vers un outil plutôt qu'un autre, il faut donc bien identifier ses critères pour prendre une décision éclairée. (Wing, 2008)

Le Prix Netflix a accéléré considérablement la recherche dans ce domaine. Netflix, site de recommandation de films très connu, offrait 1 million de dollars à l'équipe qui réussirait à améliorer leur algorithme par une marge d'au moins 10%. Les équipes de mathématiciens et d'informaticiens du monde entier ont mis plus de 2 ans pour arriver au standard demandé. Plus de 5000 équipes provenant de 186 pays ont soumis une solution valide mais une seule a réussi à atteindre le standard. C'est l'équipe BellKor's Pragmatic Chaos, une coalition de chercheurs de plusieurs laboratoires de statistique et d'informatique, qui a remporté les honneurs. (Benneth et Lanning, 2007) (Töscher *et al.*, 2009)

Ce mémoire a pour objectif de présenter une méthode récente de prévision basée sur les rangs, de même que plusieurs techniques de systèmes de recommandation

déjà existantes et reconnues dans la littérature à des fins pédagogiques. Nous comparerons ensuite ces différents algorithmes.

Dans le premier chapitre, nous introduisons le filtrage collaboratif. Dans les chapitres deux à sept, nous décrivons les algorithmes de filtrage collaboratif avec lesquels nous comparons le nouvel algorithme basé sur les rangs qui est présenté au huitième chapitre.

Le neuvième chapitre présente la comparaison des différents algorithmes de recommandation. Nous y développons la méthodologie utilisée, notamment l'utilisation de plusieurs outils et langages informatiques dont Java, Matlab et R. Nous y présentons aussi nos résultats qui confirment que la nouvelle technique est une alternative viable et particulièrement compétitive sur de petits jeux de données.

Nous introduisons également certaines notions mathématiques préalables dans plusieurs sections précédant les chapitres concernés.

CHAPITRE I

FILTRAGE COLLABORATIF

Le filtrage collaboratif est une famille de systèmes de recommandation qui utilise les données connues d'un individu ou d'un item, pour prédire les goûts ou les choix futurs d'un usager. L'ensemble des techniques présentées dans ce mémoire sont issues de cette famille. Elles utilisent les notes données à certains films par des usagers pour prévoir comment ils noteront les films dans le futur.

Il existe plusieurs types de filtrage collaboratif (FC) dans la littérature. Ils sont souvent divisés en trois, les méthodes de FC basées sur la mémoire, les méthodes de FC basées sur les modèles ainsi que les modèles hybrides (Su et Khoshgoftaar, 2009).

Pour avoir une meilleure vue d'ensemble, nous avons réorganisé les algorithmes avec une approche statistique tel que présenté à la figure 1.1. D'abord le filtrage collaboratif non-paramétrique, qui ne repose pas sur un modèle contenant des paramètres et qui sont souvent les plus simples. Ensuite viennent les modèles déterministes, ceux-ci ne reposent pas sur des lois de probabilités. Finalement, il y a les modèles stochastiques que nous séparons en deux, ceux avec une approche par maximum de vraisemblance et ceux ayant une approche bayésienne, dont fait partie le nouvel algorithme basé sur les rangs.

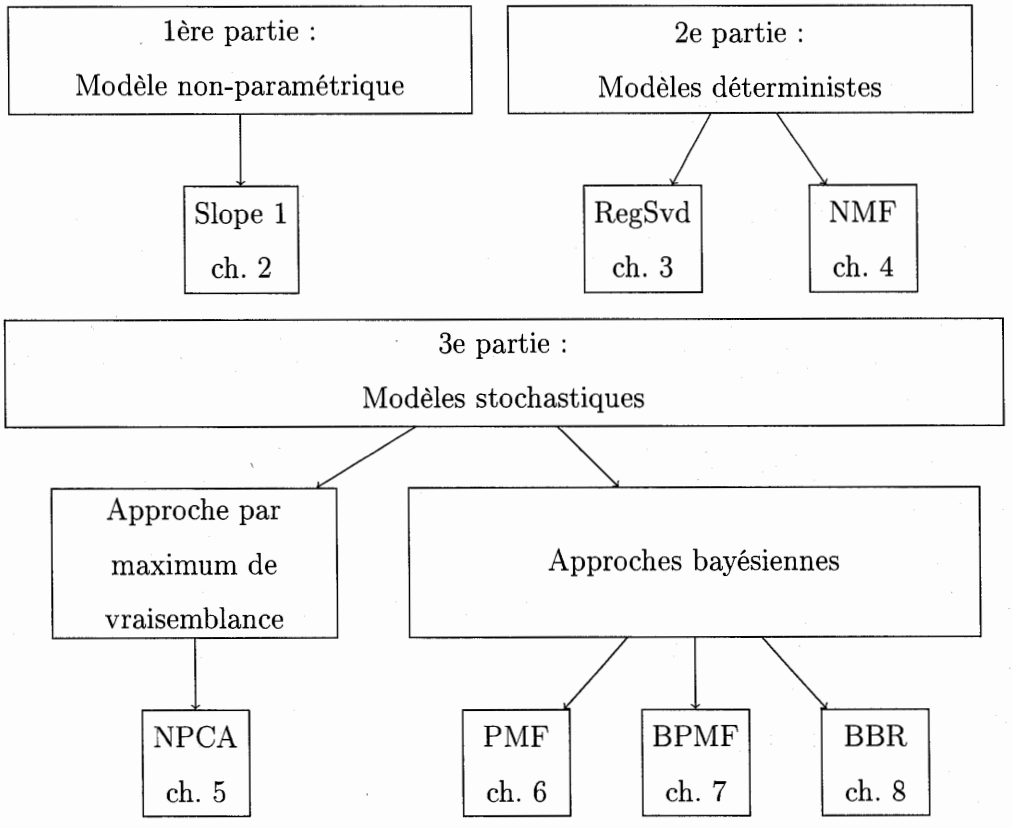


Figure 1.1 Organigramme des algorithmes présentés dans ce mémoire classés par modèles de filtrage collaboratif

Les techniques de filtrage collaboratif utilisent souvent un schéma de données avec une liste d'usagers et une liste d'items. Le concept de notes ou d'étoiles (de 1 à 5) est le plus commun et c'est celui que nous utiliserons lors de nos comparaisons. D'autres systèmes sont plus binaires, par exemple, les "j'aime"/"je n'aime pas", ou encore, un système par chaînes, dans lequel on fera des liens entres les produits ou les pages d'arrivées ou de sorties. Si un client a acheté un produit A et a aussi acheté les produits B et C, nous pourrions donc proposer B et C à un autre client qui a acheté ou montré un intérêt pour le produit A. (Linden *et al.*, 2003)(Breese *et al.*, 1998)

Ces techniques servent dans plusieurs contextes tels que des sites d'achats ou de divertissements, aussi bien pour des vidéos, des films ou de la musique. Certaines de ces entreprises, comme Netflix, Amazon, Facebook, YouTube ou MovieLens, font partie du quotidien de plusieurs millions d'usagers. (Resnick et Varian, 1997)

On peut vouloir trouver la note qu'un usager précis donnera à un film en particulier ou aussi bien trouver les n meilleurs films que celui-ci préférera. Par exemple, un top 10 des films qu'il est susceptible d'écouter et d'aimer. (Cremonesi *et al.*, 2010)

1.1 Problématiques

Plusieurs problématiques sont impliquées dans le choix de la technique utilisée et dans le développement de celle-ci.

La plupart du temps on travaille avec des matrices de données énormes et qui sont souvent creuses ou incomplètes. Ces matrices ont généralement une densité très faible. C'est-à-dire qu'elles sont remplies presque entièrement de données qui ne sont pas connues, auxquelles on attribuera des zéros. (Su et Khoshgoftaar, 2009)

Un second problème survient lorsqu'on veut produire des prédictions pour des usagers ou une population dont on ne connaît que très peu d'informations, voire

aucune information, comme pour un nouvel usager ou un nouveau film. (Schein *et al.*, 2002)

Finalement, un problème majeur est le temps de calcul des outils de prédiction. Il faut donc trouver un compromis entre le temps de calcul et la complexité ou la précision de l'outil que l'on va implanter. Par exemple, on ne voudra pas faire attendre un client pendant qu'on lui prépare sa liste d'achats potentiels. On voudra que celle-ci soit déjà calculée ou puisse se calculer très rapidement. (Bundy, 2007)

NOTATIONS UTILES AU FILTRAGE COLLABORATIF

Dans cette section nous présentons les notations générales utilisées dans la suite du mémoire ainsi que quelques rappels d'algèbre utiles à la compréhension des algorithmes de filtrage collaboratif.

Pour la comparaison des systèmes de recommandation, nous travaillons avec une matrice de données incomplète ou creuse contenant les notes données aux films par les usagers. Les notes données à un film par un usager vont de 1 à 5, on attribue un "0" aux films non-notés. Les lignes représentent les usagers et les colonnes représentent les films.

On utilise les lettres majuscules pour dénoter les matrices et les lettres minuscules pour les vecteurs colonnes. $Y \in \mathbb{R}^{M \times N}$ est une matrice, avec son élément Y_{ij}, le vecteur de la j-ème colonne $y_{\cdot j}$ et le vecteur de la i-ème ligne $y_{i\cdot}$. $\overline{y_{i\cdot}}$ est la moyenne du vecteur $y_{i\cdot}$. Sa transposée est Y^t, la trace, si applicable, est $tr(Y)$ et son déterminant est $det(Y)$.

Pour distinguer les objets observés ou non,
$O(Y)$ dénote les indices des éléments observés d'une matrice,
$card(O(Y))$ est le nombre d'observations dans la matrice Y,
$O(y_{i\cdot})$ contient les indices des éléments observés sur la i-ème ligne et,
$O_{jk}(Y)$ contient les indices des usagers ayant noté le film j et le film k.

On a également le complémentaire de ces ensembles : $O(Y)^C$ dénote les indices des éléments non-observés d'une matrice Y, $card(O(Y)^C)$ est le nombre d'éléments non-observés de cette matrice et $O(y_{i\cdot})^C$ contient les indices des éléments non-

observés dans la rangée i.

Exemple 6×4

$$Y = \begin{bmatrix} 5 & 0 & 3 & 5 \\ 4 & 5 & 0 & 0 \\ 0 & 5 & 0 & 3 \\ 4 & 0 & 3 & 1 \\ 0 & 3 & 5 & 5 \\ 4 & 4 & 5 & 0 \end{bmatrix}$$

Ici, $Y \in \mathbb{R}^{6\times 4}$, le vecteur pour l'usager 2 est $y_{2\cdot} = (4, 5, 0, 0)$, le vecteur pour le film 2 est $y_{\cdot 2} = (0, 5, 5, 0, 3, 4)^t$ et la note que l'usager 2 a donné au film 2 est $Y_{22} = 5$.

Le nombre d'éléments observés $card(O(Y)) = 16$ et leurs indices
$O(Y) = \{(1,1), (1,3), (1,4), (2,1), (2,2), (3,2), (3,4), (4,1), (4,3), (4,4), (5,2),$
$(5,3), (5,4), (6,1), (6,2), (6,3)\}$

et le nombre d'éléments non-observés $card(O(Y)^C) = 8$ et leurs indices
$O(Y)^C = \{(1,2), (2,3), (2,4), (3,1), (3,3), (4,2), (5,1), (6,4)\}$.

Pour l'usager 2, le nombre de films qu'il a noté $card(O(y_{2\cdot})) = 2$ et ces indices $O(y_{2\cdot}) = \{1, 2\}$.

Finalement, les usagers ayant noté les films 2 et 3 sont $O_{23}(Y) = \{5, 6\}$.

Nous nous référerons à nouveau à cet exemple dans le mémoire.

Distance utilisée

On retrouve la distance Euclidienne entre A et B au carré

$$\|A - B\|^2 = \sum_{i=1}^{M} \sum_{j=1}^{N} (A_{ij} - B_{ij})^2$$

ou

$$\|A - B\| = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} (A_{ij} - B_{ij})^2}.$$

Cette formule est bornée inférieurement par zéro, avec égalité si et seulement si $A = B$.

Nous notons aussi

$$\|a_{i\cdot}\|^2 = a_{i\cdot} a_{i\cdot}^t = \sum_{j=1}^{N} (A_{ij})^2.$$

PREMIÈRE PARTIE
MÉTHODE NON-PARAMÉTRIQUE

CHAPITRE II

SLOPE ONE

L'algorithme Slope One est proposé par Daniel Lemire et Anna Maclachlan (2005). Il est reconnu comme étant facile à implémenter et sert souvent à améliorer d'autres outils de prédictions. Bien qu'il soit simple d'utilisation, il se compare favorablement à des outils d'intelligence artificielle plus complexes (Wang et Ye, 2009) (Jiang et Lu, 2013) (You *et al.*, 2015).

Nous présentons d'abord quelques techniques non-paramétriques de filtrage collaboratif qui permettent de mieux comprendre et de voir l'origine de la méthode Slope One.

2.1 Moyenne de l'Usager

Cette technique consiste simplement à prendre la moyenne des films notés par l'usager. Donc, tous les films non-notés auront le même score

$$P_{ij} = \overline{y}_{i\cdot} = \frac{1}{card(O(y_{i\cdot}))} \sum_{j \in O(y_{i\cdot})} Y_{ij}, \qquad \forall j \in O(y_{i\cdot})^C,$$

où $O(y_{i\cdot})$ sont les indices des films notés par l'usager i.

Par exemple, si $y_{1\cdot} = (5, 0, 3, 5)$ alors $P_{12} = \frac{1}{3}(5 + 3 + 5) = 4.3333$

2.2 Biais de la moyenne

Un peu comme la technique de la moyenne de l'usager, on utilise la moyenne des films notés par l'usager en question mais on y ajoute l'écart de chaque film noté par rapport aux autres usagers qui ont noté ce film. La note prédite du film j par l'usager i est alors

$$P_{ij} = \overline{y}_{i\cdot} + \frac{1}{card(O(y_{i\cdot}))} \sum_{k \in O(y_{i\cdot})} (Y_{ik} - \overline{y}_{\cdot k}).$$

Par exemple, si $y_{1\cdot} = (5, 0, 3, 5)$ et la moyenne des autres usagers pour les films sont $\overline{y}_{\cdot j} = (4.25, 4.25, 4, 3.5)$ alors $P_{12} = 4.333 + \frac{1}{3}((5-4.25)+(3-4)+(5-3.5)) = 4.333 + 1.25 = 5.583$

2.3 Similarités cosinus ajustées

Le nom de cette technique vient du cosinus de l'angle θ entre deux vecteurs $a \in R^n$ et $b \in R^n$, $cos(\theta) = \frac{a \cdot b}{\|a\| \ \|b\|}$.

Supposons deux items j et k, la similarité cosinus sera :

$$\text{sim}_{jk} = \frac{\sum_{i \in O_{jk}(Y)} (Y_{ij} - \overline{y}_{i\cdot})(Y_{ik} - \overline{y}_{i\cdot})}{\sum_{i \in O_{jk}(Y)} (Y_{ik} - \overline{y}_{i\cdot})^2 \sum_{i \in O_{jk}(Y)} (Y_{ij} - \overline{y}_{i\cdot})^2}$$

où $O_{jk}(Y)$ sont les indices des usagers ayant noté les deux items j et k.

Par exemple, pour des vecteurs $y_{\cdot 1} = (5, \mathbf{4}, 0, 4, 0, \mathbf{4})^t$ et $y_{\cdot 2} = (0, \mathbf{5}, 5, 0, 3, \mathbf{4})^t$ avec $O_{12}(Y) = \{2, 6\}$, $\overline{y}_{2\cdot} = 4.5$ et $\overline{y}_{6\cdot} = 4.333$ la similarité est

$$\text{sim}_{12} = \frac{(4-4.5)(5-4.5) + (4-4.333)(4-4.333)}{((4-4.5)^2 + (4-4.333)^2)((5-4.5)^2 + (4-4.333)^2))} = -1.068.$$

On fait ensuite la somme pondérée de toutes les similarités pour obtenir la pré-

diction suivante

$$P_{ij} = \frac{\sum_{k \in O(y_{i\cdot})} |sim_{jk}|(\alpha_{jk} Y_{ik} + \beta_{jk})}{\sum_{k \in O(y_{i\cdot})} |sim_{jk}|}.$$

avec les coefficients α_{jk} et β_{jk} qui minimisent $\sum_{i \in O_{j,k}(Y)} (\alpha_{jk} Y_{ik} \beta_{jk} - Y_{ij})^2$.

Dans l'exemple, avec $\alpha_{2\cdot} = (1, 0, 1, 1)$, $\beta_{2\cdot} = (1.125, 0, 1.4285, 1)$ et les similarités $sim_{2\cdot} = (1.068, 0, 0.6625, 0.4709)$

$$P_{12} = \frac{1.068(5 + 1.125) + 0.6625(3 + 1.4285) + 0.4709(5 + 1)}{1.068 + 0.6625 + 0.4709} = 5.5912.$$

Pour plus de détails sur la technique de similarités cosinus ajustées voir (Singhal, 2001) et (Nguyen et Bai, 2010).

2.4 L'algorithme Slope One

Slope One tient compte des films notés par l'usager, comme dans les techniques de la moyenne de l'usager à la section 2.1 et du biais de la moyenne à la section 2.2. Il tient également compte de l'information des autres usagers qui ont noté le film pour lequel on souhaite faire des prédictions, comme dans la technique des similarités cosinus ajustées développée à la section 2.3.

Slope One, que l'on peut traduire de l'anglais comme "Pente Un", porte ce nom car on cherche le meilleur prédicteur linéaire de pente égale à 1.

Pour trouver cela, supposons deux vecteurs d'items j et k. On cherche à optimiser

$$\min_b \sum_i (Y_{ik} + b - Y_{ij})^2. \tag{2.1}$$

En dérivant l'expression par rapport à b, on obtient

$$\frac{d}{db}\sum_i (Y_{ik} + b - Y_{ij})^2 = 2\sum_i (Y_{ik} + b - Y_{ij}).$$

En égalisant la dérivée à 0

$$2\sum_i (b + Y_{ik} - Y_{ij}) = 2bn + 2\sum_i (Y_{ik} - Y_{ij}) = 0,$$

ceci implique que

$$b = \sum_i \frac{Y_{ij} - Y_{ik}}{n} = D_{jk}. \tag{2.2}$$

On nomme les b entre chaque couple d'items (j, k) les "déviations", qu'on note D_{jk}.

Par exemple, si nous avons 2 usagers et 2 films. L'usager 1 a noté les deux films et l'usager 2 seulement le film 1.

$$Y = \begin{bmatrix} 5 & 4 \\ 4 & 0 \end{bmatrix}$$

Nous cherchons $P_{22} = 4 + b$, avec $b = (4 - 5)$. La note donnée par l'usager 1 au film 2 sera de $P_{22} = 4 + (4 - 5) = 3$. Bien que trivial à cette échelle, sur un grand ensemble cela s'avérera plutôt performant.

Une grande partie de sa force vient du fait que contrairement à certaines techniques décrites plus haut, Slope One utilise aussi les données externes à l'usager lui-même ou à l'item en question. Dans l'exemple ici, la donnée externe sera la note de l'usager A au film 1.

Si on généralise à une matrice $Y \in \mathbb{R}^{M \times N}$, on regarde en premier les dites déviations entre chaque film j et k. On définit la matrice de déviation $D \in \mathbb{R}^{N \times N}$ qu'on calcule au préalable avec

$$D_{jk} = \sum_{i \in O_{jk}(Y)} \frac{Y_{ij} - Y_{ik}}{card(O_{jk}(Y))}, \tag{2.3}$$

où $O_{jk}(Y)$ sont les indices des usagers ayant noté les deux items j et k.

Avec ces déviations, on définit les "prédictions conditionnelles"

$$P_{ij|k} = D_{jk} + Y_{ik}. \tag{2.4}$$

On peut interpréter cela comme étant la prédiction pour le film j de l'usager i sachant sa note pour le film k et cela en tenant compte de toutes les différences entre les deux films.

En faisant la moyenne de toutes ces prédictions conditionnelles, on obtient la prédiction finale

$$P_{ij} = \frac{1}{card(R_j)} \sum_{k \in R_j} P_{ij|k}, \tag{2.5}$$

où $R_j = \{k \mid k \in O(y_{i\cdot}),\ j \neq k,\ card(O_{jk}(Y) > 0)\}$, c'est-à-dire les films k notés par l'usager i si au moins un usager a noté j et k.

2.5 Exemple 6×4

Par exemple, nous pouvons résoudre cette matrice avec Slope One.

$$Y = \begin{bmatrix} 5 & 0 & 3 & 5 \\ 4 & 5 & 0 & 0 \\ 0 & 5 & 0 & 3 \\ 4 & 0 & 3 & 1 \\ 0 & 3 & 5 & 5 \\ 4 & 4 & 5 & 0 \end{bmatrix}$$

On calcule d'abord notre matrice de déviation entre les films.

$$D = \begin{bmatrix} 0 & -0.5 & 2/3 & 1.5 \\ 0.5 & 0 & -1.5 & 0 \\ -2/3 & 1.5 & 0 & 0 \\ -1.5 & 0 & 0 & 0 \end{bmatrix}$$

Par exemple, $D_{13} = \frac{(5-3)+(4-3)+(4-5)}{3} = 2/3$.

La matrice finale avec les prédictions sera

$$P = \begin{bmatrix} 5 & (4) & 3 & 5 \\ 4 & 5 & (4.917) & (3.75) \\ (4.5) & 5 & (5.75) & 3 \\ 4 & (2.333) & 3 & 1 \\ (4.888) & 3 & 5 & 5 \\ 4 & 4 & 5 & (3.833) \end{bmatrix}$$

Par exemple, $P_{12} = \frac{(5+0.5)+(3-2.5)+(5+0)}{3} = 4$.

2.6 Pseudo-code

Algorithme 1 Slope One

Exigences : *La matrice de données* $Y \in \mathbb{R}^{M \times N}$

Créer : Les matrices de déviations $D \in \mathbb{R}^{N \times N}$ *et de prédictions* $P \in \mathbb{R}^{M \times N}$

Pour $j = 1, ..., N$, $k = 1, ..., N$, $j \neq k$

    $D_{jk} = \sum_{i \in O_{jk}(Y)} \frac{Y_{ik} - Y_{ij}}{card(O_{jk}(Y))}$

Fin

Pour $i = 1, ..., M$ et $j = 1, ..., N$

    Pour chaque $Y_{ij} \in O(Y)^c$, $\quad P_{ij} = \frac{1}{card(R_j)} \sum_{k \in R_j} (D_{jk} + Y_{ik})$

    Pour chaque $Y_{ij} \in O(Y)$, $\quad P_{ij} = Y_{ij}$

Retourner P

DEUXIÈME PARTIE
MÉTHODES AVEC MODÈLES DÉTERMINISTES

CHAPITRE III

RÉGULARISATION SVD

L'algorithme de **régularisation SVD** (RegSVD) vient de la technique, assez bien connue en algèbre linéaire, de décomposition en valeurs singulières (SVD). Celle-ci, telle que présentée plus bas, a été améliorée par (Paterek, 2007) qui s'est basé sur le travail de (Webb et Gorrell, 2006), inspiré du traitement automatique du langage naturel. Ce domaine fut d'ailleurs prisé et rendu célèbre par Alan Turing, mathématicien et cryptologue célèbre (Saygin *et al.*, 2000). Fait intéressant, le film *The Imitation Game*, inspiré de sa vie, est disponible sur Netflix.

La technique RegSVD est essentiellement basée sur le théorème suivant

Théorème 1 (Décomposition en valeurs singulières) *Pour chaque matrice $Y \in \mathbb{R}^{M \times N}$, il existe une factorisation de la forme*

$$Y = U\Sigma V^t, \tag{3.1}$$

avec $U \in \mathbb{R}^{M \times M}$ et $V \in \mathbb{R}^{N \times N}$ des matrices orthogonales ($UU^t = I$, $VV^t = I$), et $\Sigma \in \mathbb{R}^{M \times N}$ une matrice dont les coefficients diagonaux sont positifs ou nuls.

Bien que cette représentation soit exacte, U et V peuvent être très grandes et sont parfois difficiles à trouver. On se base sur ce résultat pour construire un modèle

plus simple pour Y en réduisant l'ordre des matrices de la décomposition en (3.1). Plus précisément, on introduit un nouveau paramètre K, $1 \leq K \leq min(M, N)$, de sorte que

$$Y \approx U_{M\times K}\Sigma_{K\times K}V^t_{N\times K}. \tag{3.2}$$

On se réfère souvent à cette factorisation comme la décomposition en valeurs singulières tronquée (Hansen, 1987).

Ensuite pour simplifier, on incorpore $\Sigma = \Sigma^{1/2}(\Sigma^{1/2})^t$ dans les vecteurs U et V de telle sorte que

$$Y \approx U^*_{M\times K}(V^*_{N\times K})^t, \tag{3.3}$$

avec $U^* = U\Sigma^{1/2}$ et $V^* = V\Sigma^{1/2}$.

La notation en (3.2) et (3.3) est là pour mieux représenter la grandeur des matrices, par exemple $U_{M\times K}$ signifie que $U \in \mathbb{R}^{M\times K}$.

Pour trouver les deux matrices U et V de manière numérique, on veut se rapprocher des données de la matrice originale sans pour autant faire de surapprentissage. Le problème d'optimisation devient

$$\min_{u_{i\cdot},v_{i\cdot}} \|Y - UV^t\|^2 + \lambda\|U\|^2 + \lambda\|V\|^2, \tag{3.4}$$

où $\lambda \geq 0$, $U \in R^{M\times K}$, $V \in R^{N\times K}$. Notons que dans (3.4), la norme est calculée uniquement à partir des composantes de la matrice $(Y - UV^t)$ pour lesquelles $Y_{ij} > 0$ ou dont les indices $i, j \in O(Y)$, $i = 1, ..., M$ et $j = 1, ..., N$.

Pour résoudre cela, il est commun d'utiliser l'algorithme du gradient. Il faudra aussi choisir un bon paramètre de régularisation λ car s'il est trop petit, on se rapproche trop de Y, et s'il est trop grand, les coefficients des matrices U et V se rapprochent de zéro.

3.1 Algorithme du gradient

L'**algorithme du gradient** est une technique d'optimisation qui permet de trouver le minimum d'une fonction. On cherche

$$\min_x f(x),$$

avec $x \in \mathbb{R}^n$ et $f(x)$ une fonction différentiable.

On voudra calculer une nouvelle valeur de x en fonction de son état actuel, on aura alors

$$x_{t+1} = x_t - t_a \nabla_t,$$

avec $\nabla_t = \frac{d}{dx_t} f(x_t)$.

Le "taux d'apprentissage" t_a influence la vitesse à laquelle on arrivera potentiellement à un minimum. On se dirige vers ce minimum en se déplaçant dans la direction opposée au gradient, d'où le signe négatif.

Pour plus de détails sur l'algorithme du gradient voir (Yuan, 2008).

3.2 Algorithme du gradient pour RegSVD

La prédiction est calculée comme suit

$$P_{ij} = u_{i\cdot} v_{j\cdot}^t \,, \tag{3.5}$$

où $u_{i\cdot}$ et $v_{j\cdot}^t$ sont les coefficients des matrices U et V de l'équation (3.4).

On calcule les erreurs entre les données de la matrice Y et celles prédites par le produit scalaire comme suit

$$R_{ij} = Y_{ij} - P_{ij}.$$

Les gradients de la fonction à optimiser en (3.4) sont

$$\nabla_U = R_{ij}V_{jk} - \lambda U_{ik} \quad et \quad \nabla_V = R_{ij}U_{ik} - \lambda V_{jk}.$$

Ensuite, on actualise les données des matrices de prédictions U et V avec le taux d'apprentissage t_a et le paramètre de régularisation λ comme suit

$$U_{ik}^{(n+1)} = U_{ik}^{(n)} + t_a\nabla_U \quad et \quad V_{jk}^{(n+1)} = V_{jk}^{(n)} + t_a\nabla_V$$

(Chin *et al.*, 2015) (Paterek, 2007)

On poursuit l'entraînement des paramètres et l'actualisation des matrices jusqu'au moment où l'erreur totale est inférieure à un ϵ choisi d'avance ou jusqu'au nombre d'itérations maximales.

Les matrices U et V sont donc actualisées avec les données contenues dans la matrice d'entraînement Y.

On peut alors effectuer les prédictions pour les données manquantes par le produit scalaire des vecteurs d'intérêts pour un usager et un film en particulier.

On calcule la note prédite pour l'usager 5 au film 8 comme suit

$$P_{85} = u_{8\cdot}v_{5\cdot}^t.$$

3.3 Exemple 6×4

Voici les prédictions que donnera RegSVD à l'exemple où $Y \in \mathbb{R}^{6\times4}$.

$$Y = \begin{bmatrix} 5 & 0 & 3 & 5 \\ 4 & 5 & 0 & 0 \\ 0 & 5 & 0 & 3 \\ 4 & 0 & 3 & 1 \\ 0 & 3 & 5 & 5 \\ 4 & 4 & 5 & 0 \end{bmatrix}$$

Après quelques itérations avec un taux d'apprentissage de $t_a = 0.005$, $\lambda = 0.1$ et $K = 3$,

$$U = \begin{bmatrix} 1.3675467991 & 0.950147044016 & 1.77473126826 \\ 0.74005986027 & 0.32833290662 & 1.714474046706 \\ 0.7173303874 & 1.585951249725 & 1.27518084352 \\ 0.054474151908 & 1.671807681707 & 0.789315853223 \\ 1.54118104293 & 0.00709431175702 & 1.447727463713 \\ 1.1794398271 & 1.53443473863 & 0.808920596498 \end{bmatrix}$$

$$V^t = \begin{bmatrix} 0.55908902951 & 0.64629184565 & 1.09653476269 & 1.51387914432 \\ 1.41817692639 & 1.60402347362 & 1.60312620231 & -0.102788897454 \\ 1.5685153077 & 1.28545775377 & 0.5668681142 & 1.6189811704 \end{bmatrix}$$

La matrice finale avec les prédictions sera

$$P = \begin{bmatrix} 5 & (4.69) & 3 & 5 \\ 4 & 5 & (2.31) & (3.86) \\ (4.65) & 5 & (4.05) & 3 \\ 4 & (3.73) & 3 & 1 \\ (3.14) & 3 & 5 & 5 \\ 4 & 4 & 5 & (2.94) \end{bmatrix}$$

Par exemple,

$$\begin{aligned} P_{12} = u_1.v_{.2}^t &= (1.36755, 0.95015, 1.77473)(0.64629, 1.60402, 1.28546)^t \\ &= 4.6888. \end{aligned}$$

3.4 Pseudo-code

Algorithme 2 RegSVD

Exigences : Y, $iter_{max}$, t_a, λ, K, ϵ

Créer : $P \in \mathbb{R}^{M \times N}$, $U \in \mathbb{R}^{M \times K}$, $V \in \mathbb{R}^{N \times K}$

Initialiser : $iter = 0$

Pour $iter = 1, ..., iter_{max}$

    $iter = iter + 1$

    Réinitialise $Er = 0$

    Pour chaque $i = 1, ..., N,\ j = 1, ...M$ et $Y_{ij} \neq 0$

    $P_{ij} = u_{i.} v_{j.}^t$

    $R_{ij} = Y_{ij} - P_{ij}$

    $Er = Er + R_{ij}$

        Pour chaque $k = 1, ..., K$

            $U_{ik} = U_{ik} + t_a(R_{ij} V_{jk} - \lambda U_{ik})$

            $V_{jk} = V_{jk} + t_a(R_{ij} U_{ik} - \lambda V_{jk})$

Fin Si $iter = iter_{max}$ ou $Er < \epsilon$

Pour $i = 1, ..., M$ et $j = 1, ..., N$

    Pour chaque $Y_{ij} \in O(Y)$, $P_{ij} = Y_{ij}$

Retourner P

CHAPITRE IV

FACTORISATION DE MATRICES NON-NÉGATIVE

Pour produire de bonnes prédictions, l'algorithme RegSVD nécessite le choix de bons paramètres de régularisation et d'apprentissage. L'algorithme de **factorisation de matrice non-négative** (NMF), proposé par (Lee et Seung, 1999), nous présente une façon moins arbitraire d'effectuer une factorisation de matrice similaire.

On dit qu'une matrice Y est non-négative quand $Y_{ij} \geq 0$ pour chaque couple (i, j).

On considère $Y \in \mathbb{R}^{M \times N}$ la matrice de données non-négative avec ses entrées $Y_{ij} \in \{0, ..., 5\}$. On cherchera donc une factorisation de celle-ci avec deux facteurs non-négatifs $U \in \mathbb{R}^{M \times K}$ et $V \in \mathbb{R}^{N \times K}$, tel que

$$Y \approx UV^t.$$

Plutôt que de poser une pénalité sur les normes de U et V comme en (3.4), on voudra résoudre le problème d'optimisation suivant

$$\min_{U \geq 0, V \geq 0} \|Y - UV^t\|^2, \tag{4.1}$$

où la norme $\|Y - UV^t\|^2$ est calculé uniquement sur les éléments $(i, j) \in O(Y)$.

4.1 Actualisation multiplicative

Pour résoudre l'équation (4.1) on utilise une règle d'actualisation dite "multiplicative". Pour obtenir cette règle, on peut faire un lien avec un modèle "additif". On part d'une optimisation simple par addition, très similaire à l'algorithme du gradient de sorte que

$$U_{ik}^{(n+1)} = U_{ik}^{(n)} + t_a(YV - UV^tV)_{ik}^{(n)}$$

et

$$V_{jk}^{(n+1)} = V_{jk}^{(n)} + t_b(Y^tU - VU^tU)_{jk}^{(n)}.$$

Si on prend tous les t_a et les t_b égaux et fixes, on revient à l'algorithme du gradient. Les valeurs $(YV - UV^tV)_{ik}^{(n)}$ et $(Y^tU - VU^tU)_{jk}^{(n)}$ correspondent aux gradients de la fonction à minimiser en (4.1).

Plutôt que d'être fixes, les taux d'apprentissage t_a et t_b sont considérés variables, propres à chaque couple (i,k) et (j,k) et ne sont pas déterminés d'avance. On pose les $t_a = \tau_{ik}$ et les $t_b = \eta_{jk}$ tel que

$$\tau_{ik} = \frac{U_{ik}^{(n)}}{(UV^tV)_{ik}^{(n)}} \qquad et \qquad \eta_{jk} = \frac{V_{jk}^{(n)}}{(VU^TU)_{jk}^{(n)}}.$$

On obtient alors les règles suivantes

$$U_{ik}^{(n)} + \tau_{ik}(YV - UV^tV)_{ik}^{(n)} = U_{ik}^{(n)} + \frac{U_{ik}^{(n)}}{(UV^tV)_{ik}^{(n)}}(YV - UV^tV)_{ik}^{(n)} = U_{ik}^{(n)} \frac{(YV^t)_{ik}^{(n)}}{(UV^tV)_{ik}^{(n)}}$$

et

$$V_{jk}^{(n)} + \eta_{jk}(Y^tU - VU^tU)_{jk}^{(n)} = V_{jk}^{(n)} + \frac{V_{jk}^{(n)}}{(VU^tU)_{jk}^{(n)}}(Y^tU - VU^tU)_{jk}^{(n)} = V_{jk}^{(n)} \frac{(Y^tU)_{jk}^{(n)}}{(VU^tU)_{jk}^{(n)}}.$$

L'article de (Lee et Seung, 2001) garantit la convergence de cet algorithme vers un minimum, du moins local.

À chaque itération, on réinitialise l'erreur et elle se calcule comme suit

$$Er = \sum_{i=1}^{M} \sum_{j \in O(y_{i\cdot})} (Y_{ij} - u_{i\cdot} v_{j\cdot}^{t})^2.$$

On poursuit les itérations jusqu'à ce que l'erreur Er soit inférieure à un ϵ choisi d'avance ou jusqu'au nombre d'itérations maximales.

La prédiction finale est ensuite donnée par le produit scalaire des vecteurs d'intérêts pour un usager et un film en particulier.

$$P_{ij} = u_{i\cdot} v_{j\cdot}^{t} \tag{4.2}$$

4.2 Exemple 6×4

Voici les prédictions que donnera NMF à l'exemple où $Y \in \mathbb{R}^{6\times4}$.

$$Y = \begin{bmatrix} 5 & 0 & 3 & 5 \\ 4 & 5 & 0 & 0 \\ 0 & 5 & 0 & 3 \\ 4 & 0 & 3 & 1 \\ 0 & 3 & 5 & 5 \\ 4 & 4 & 5 & 0 \end{bmatrix}$$

Après quelques itérations avec $K = 3$,

$$U = \begin{bmatrix} 10.06196502607 & 3.38540657512 & 1.6145970615 \\ 1.23611206681 & 7.35774425907 & 2.25233283511 \\ 9.25066673004 & 0.419182140751 & 3.500189916421 \\ 4.42755918896 & 5.914336420664 & 2.128947089528 \\ 3.673368151915 & 5.33351046378 & 2.086935558246 \\ 3.70340323484 & 4.91732895904 & 3.03064523482 \end{bmatrix}$$

$$V^t = \begin{bmatrix} 0.299218178038 & 0.46939669294 & 0.1024758463 & 0.380643393809 \\ 0.37199611436 & 0.270360327089 & 0.29840982494 & 0.429345599731 \\ 0.340771661691 & 0.14467546381 & 0.27845320079 & 0.069274038624 \end{bmatrix}$$

La matrice finale avec les prédictions sera

$$P = \begin{bmatrix} 5 & (5.87) & 3 & 5 \\ 4 & 5 & (2.95) & (3.79) \\ (4.12) & 5 & (2.05) & 3 \\ 4 & (3.98) & 3 & 1 \\ (3.79) & 3 & 5 & 5 \\ 4 & 4 & 5 & (3.73) \end{bmatrix}$$

Par exemple,

$$\begin{aligned} P_{12} = u_{1.}v_{2.}^t &= (10.06197, 3.3854, 1.61146)(0.46939, 0.27036, 0.14468)^t \\ &= 5.8714. \end{aligned}$$

4.3 Pseudo-code

Algorithme 3 NMF

Exigence : $Y, K, iter_{max}, \epsilon$

Créer : $P \in \mathbb{R}^{M \times N}, U \in \mathbb{R}^{M \times K}, V \in \mathbb{R}^{N \times K}$

Initialise : $iter = 0$

Pour $iter < iter_{max}$

    $iter = iter + 1$

    Réinitialise $Er = 0$

    Pour chaque $i = 1, ..., M, j = 1, ...N$ et $Y_{ij} \neq 0$

    $P_{ij} = u_i.v_{j.}^t$

    $Er = Er + (Y_{ij} - P_{ij})^2$

        Pour chaque $k = 1, ..., K$

            $U_{ik} = U_{ik} \frac{(YV^T)_{ik}}{(UV^tV)_{ik}}$

            $V_{jk} = V_{jk} \frac{(Y^tU)_{jk}}{(VU^tU)_{jk}}$

Fin Si $iter = iter_{max}$ ou $Er < \epsilon$

Pour $i = 1, ..., M$ et $j = 1, ..., N$

    Pour chaque $Y_{ij} \in O(Y)$, $P_{ij} = Y_{ij}$

    Pour chaque $Y_{ij} \in O(Y)^C$, $P_{ij} = u_i.v_{j.}^t$

Retourner P

TROISIÈME PARTIE
MÉTHODES AVEC MODÈLES STOCHASTIQUES

NOTIONS UTILES : LES MODÈLES STOCHASTIQUES

Nous présentons ici quelques notions de probabilité qui seront utiles pour les modèles stochastiques, autant ceux avec une approche par maximum de vraisemblance que bayésienne, que nous abordons dans les prochains chapitres.

Dans tout problème statistique, on a un échantillon de données $\{x_1, ..., x_n\}$ et on veut tirer des conclusions sur la population entière. L'approche basée sur un modèle consiste à supposer que, conditionnellement à la connaissance du modèle, nous sommes en mesure de préciser la loi de probabilité sous-jacente aux données.

Soit $\{f(\cdot \mid \theta) \ : \ \theta \in \Theta\}$, un ensemble de densités de probabilités paramétré par Θ. Ceci signifie que $\{x_1, ..., x_n\}$ sont des observations indépendantes et identiquement distribuées (iid) de la variable aléatoire X et X a pour densité $f(\cdot \mid \theta)$ pour un certain θ.

Avec l'**approche par maximum de vraisemblance**, on veut trouver le *bon* $\theta \in \Theta$. La façon usuelle de le faire est de considérer la fonction de vraisemblance

$$\theta \longmapsto L(\theta \mid x) = \prod_{i=1}^{n} f(x_i \mid \theta)$$

et de retourner le $\hat{\theta}$ qui maximise cette dernière comme le *bon* estimateur, qu'on appelle le **maximum de vraisemblance** (MLE). Une quantification de l'incertitude peut être donnée par une région de confiance construite en utilisant l'usuelle normalité asymptotique de $\hat{\theta}$.

Avec l' **approche bayésienne**, l'incertitude sur la valeur du paramètre θ est modélisée par une distribution π dite a priori. De ce point de vue, $\{x_i\}$ est conditionnnellement indépendant étant donné θ et ditribué selon $f(\cdot \mid \theta)$.

Une fois les données $\{x_1, ..., x_n\}$ prises en considération, on arrive à quantifier l'incertitude a posteriori au travers de la loi de probabilité

$$\pi(\theta \mid x) = \frac{L(\theta \mid x)\pi(\theta)}{\int_\Theta L(\theta \mid x)\pi(\theta)d\theta}.$$

Cette distribution sur le paramètre θ permet d'obtenir, entre autres, l'estimateur du **maximum a posteriori** (MAP). Celui-ci maximise la vraisemblance pénalisée par la distribution π. L'estimateur MAP se calcule comme suit

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, L(\theta \mid x)\pi(\theta).$$

Une quantification de l'incertitude s'obtient par la considération de régions de haute probabilité a posteriori.

Cette approche provient de la **formule de Bayes** qui nous permet d'exprimer la probabilité d'un événement A (paramètre) sachant un événement B (données) avec la probabilité de B sachant A via une pondération, et elle s'écrit comme suit

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}.$$

Une critique qu'un statisticien qui adopte l'approche par maximum de vraisemblance pourrait faire à l'approche bayésienne réside dans la subjectivité du choix de la loi a priori. Un bayésien pourrait alors répondre que le modèle en soit est un choix arbitraire.

Voici quelques modèles probabilistes qui seront utiles dans les prochains chapitres.

La **fonction Gamma** est très importante car elle permet de calculer les probabilités et les moments de plusieurs loi. On l'exprime comme

$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1}e^{-x}dx.$$

La fonction Gamma satisfait plusieurs relations utiles telle que $\Gamma(\alpha+1) = \alpha\Gamma(\alpha)$ avec $\alpha > 0$. En particulier, $\Gamma(n+1) = n!$ avec un entier $n > 1$ et $\Gamma(1) = 1$. En faisant un lien avec la loi Gaussienne, on vérifie facilement que $\Gamma(1/2) = \sqrt{\pi}$.

On dit qu'une variable aléatoire X suit une **loi Bêta** si sa densité est donné par

$$f(x \mid \alpha, \beta) = \frac{1}{B(\alpha,\beta)}x^{\alpha-1}(1-x)^{\beta-1}, \qquad x \in (0,1),\ \alpha,\beta > 0,$$

avec $B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$.

On peut voir la **loi de Dirichlet** comme une généralisation de la loi Bêta. On dit qu'un vecteur aléatoire X suit une loi de Dirichlet, écrit comme $X \sim D(\alpha)$ avec $\sum_{i=1}^k X_i = 1$ et $X_k = 1 - \sum_{i=1}^{k-1} X_i$, si sa fonction de densité est

$$f(x_1,...,x_k|\alpha_1,...,\alpha_k) = \frac{1}{B(\alpha)}\prod_{i=1}^k x_i^{\alpha_i}, \qquad \sum_{i=1}^k x_i = 1,\ \alpha_1,...,\alpha_k > 0,$$

avec $B(\alpha) = \frac{\prod_{i=1}^k \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^k \alpha_i)}$.

On dit qu'une variable aléatoire X suit une **loi Gamma** de paramètres α et β, écrit $X \sim G(\alpha,\beta)$, si elle a pour densité

$$f(x \mid \alpha, \beta) = \frac{x^{\alpha-1}e^{x/\beta}}{\Gamma(\alpha)\beta^\alpha}, \qquad x > 0,\ \alpha,\beta > 0,$$

avec $\Gamma(\alpha)$ la fonction Gamma et β le paramètre d'échelle. Le paramètre d'échelle permet d'exprimer la dispersion d'une densité ; plus le paramètre est grand plus la densité est étendue.

Une somme $\sum_{i=1}^{n} X_i$ de n variables aléatoires indépendantes de loi $G(\alpha_i, \beta)$ devient une loi $G\left(\sum_{i=1}^{n} \alpha_i, \beta\right)$.

Dans la famille de loi Gamma, on retrouve la loi de **Chi-deux** χ^2. La $\chi^2(n)$ est une loi Gamma de paramètres $\alpha = n/2$ et $\beta = 2$

On dit qu'une variable aléatoire $X \in \mathbb{R}$ suit une **loi Gaussienne** $\mathcal{N}(\mu, \sigma^2)$ si X a pour densité

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}}, \qquad x \in \mathbb{R}.$$

On a que $E(X) = \mu$ et $Var(X) = \sigma^2$.

Lorsque $X \sim \mathcal{N}(0,1)$, X^2 suit une $Gamma(1/2, 2)$ et cette loi est la $\chi^2(1)$. Si les X_i, $i = 1, ..., n$, sont iid $N(0,1)$. La variable aléatoire $Y := \sum_{i=1}^{n} X_i^2$ suit une loi du $\chi^2(n)$ avec n degrés de liberté.

On dit d'un vecteur $X \in \mathbb{R}^p$ suit une **loi Gaussienne multivariée** avec le vecteur des moyennes μ et la matrice de covariance Σ, qu'on écrit $\mathcal{N}_p(\mu, \Sigma)$, si sa densité est

$$f(x \mid \mu, \Sigma) = \frac{1}{(\sqrt{2\pi})^p det(\Sigma)^{1/2}} e^{(x-\mu)^t \Sigma^{-1} (x-\mu)/2}, \qquad x \in \mathbb{R}^p.$$

On a $E(X) = \mu$ et $E[(X-\mu)(X-\mu)^t] = Cov(Y) = \Sigma$.

La **loi de Wishart** est la loi de la matrice de covariance empirique d'un échantillon aléatoire d'une loi Gaussienne multivariée. Si $X_1, ..., X_n$ sont iid de loi Gaussienne

multivariée $\mathcal{N}_p(\mu, \Sigma)$, alors

$$Y := \sum_{i=1}^{n} (X_i - \mu)(X_i - \mu)^t$$

suit une loi de Wishart de paramètres n et Σ, qu'on écrit comme $Y \sim \mathcal{W}_p(n, \Sigma)$. Lorsque $n \geq p$, la loi a une densité sur l'espace des matrices $p \times p$ et sa densité est donnée par

$$f(y \mid n, \Sigma) = \frac{1}{C} det(y)^{(n-p-1)/2} e^{(-\frac{1}{2}Tr(\Sigma^{-1}y))}, \qquad y \in \mathbb{R}^{p \times p},$$

avec $C = 2^{np/2} det(\Sigma)^{n/2} \Gamma_p(n/2)$ et $\Gamma_p(n/2) = \pi^{p(p-1)/4} \prod_{j=1}^{p} \Gamma(\frac{n}{2} - \frac{j-1}{2})$.

Si un vecteur aléatoire X suit une loi Gaussienne multivariée et qu'on veut poser une loi a priori sur les paramètres μ et $\Lambda = \Sigma^{-1}$, on peut utiliser une loi conjuguée **Gaussienne-Wishart**, qu'on écrit $(\mu, \Lambda) \sim \mathcal{NW}_p(\mu_0, \lambda_0, \nu_0, W_0)$. Sa densité est

$$f(\mu, \Lambda \mid \mu_0, \lambda_0, \nu_0, W_0) = g(\mu \mid \mu_0, (\lambda_0 \Lambda)^{-1}) h(\Lambda \mid \nu_0, W_0)$$

avec g une Gaussienne $\mathcal{N}_p(\mu_0, (\lambda_0 \Lambda)^{-1})$ de moyennes μ_0 et de covariance $(\lambda_0 \Lambda)^{-1}$ et h une Wishart $\mathcal{W}_p(\nu_0, W_0)$ avec ν_0 degrés de liberté et une "matrice d'échelle" W_0.

Si $(\mu, \Lambda) \sim \mathcal{NW}_p(\mu_0, \lambda_0, \nu_0, W_0)$, la loi a posteriori des paramètres $\pi(\mu, \Lambda \mid x)$ sera alors une Gaussienne-Wishart $\mathcal{NW}(\mu_n, \lambda_n, \nu_n, W_n)$ avec $W_n = W_0 + S + \frac{pn}{p+n}(\mu_0 - \bar{x})(\mu_0 - \bar{x})^t$, $\mu_n = p\mu_0 + \frac{n\bar{x}}{p+n}$, $\nu_n = \nu_0 + n$, $\lambda_n = \lambda_0 + n$ et $S = \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^t$.

Pour plus de détails sur la loi Gaussienne multivariée, la loi de Wishart et la loi Gaussienne-Wishart, voir les ouvrages de (Wishart, 1928), (Rawlings *et al.*, 2001), (Rencher, 2003) et (Murphy, 2007).

APPROCHE PAR MAXIMUM DE VRAISEMBLANCE

CHAPITRE V

NPCA

L'algorithme NPCA tire son nom de l'analyse en composantes principales, une méthode d'analyse de données en statistique multivariée, et est proposé par (Yu *et al.*, 2009). Pour les auteurs, le "N" tient pour non-paramétrique car on ne fixe pas les paramètres, bien que le modèle en contient plusieurs. À ne pas confondre avec ce qu'on entend dans ce mémoire comme modèle non-paramétrique, c'est-à-dire les modèles qui ne reposent pas sur l'estimation de paramètres appartenant à un espace de dimension fini.

L'**analyse en composantes principales** (PCA), proposée initialement par (Pearson, 1901), considère les lignes d'une matrice de données comme des observations d'un vecteur aléatoire et les colonnes comme les variables. On cherche à "expliquer" les données par des variables non-observées qu'on appelle composantes principales. La PCA crée un lien entre les vecteurs des matrices $Y \in \mathbb{R}^{M\times N}$ et $U \in \mathbb{R}^{M\times K}$ à l'aide de la matrice $V \in \mathbb{R}^{N\times K}$ comme suit

$$y_{i\cdot} = u_{i\cdot}V^t + I\mu_i + \varepsilon_{i\cdot}, \tag{5.1}$$

avec la matrice d'erreurs $\varepsilon \in \mathbb{R}^{M\times N}$ et le vecteur des moyennes $\mu \in \mathbb{R}^M$.

On cherche alors les K composantes principales qui maximisent la variance. Ces

composantes sont des nouveaux axes. On peut effectuer cette tâche de plusieurs manières entre autres à l'aide de la décomposition en valeurs singulières ou de la décomposition en valeurs propres. Pour plus de détails sur la PCA voir (Wold *et al.*, 1987).

Comme les modèles de PCA ne sont pas associés à des lois de probabilités, un modèle d'**analyse en composantes principales probabilistes** (PPCA) a été proposé par (Tipping et Bishop, 1999). En reprenant l'équation (5.1), les $u_{i\cdot}$ sont maintenant des variables qui suivent une distribution $\mathcal{N}(0, I)$, les $\varepsilon_{i\cdot} \sim \mathcal{N}(0, \lambda I)$ sont les bruits Gaussiens indépendants, pour $i = 1, ..., M$, et on travaille avec des données centrées.

Le modèle **NPCA** pose $X = UV^t$, $X \in \mathbb{R}^{M \times N}$. Le vecteur $x_{i\cdot} = u_{i\cdot}V^t = (X_{i1}, ..., X_{iN})$ suit une distribution Gaussienne $\mathcal{N}_N(0, VV^t)$. Pour simplifier la notation, on pose $C = VV^t$, la covariance de $x_{i\cdot}$ et on travaille également avec des données centrées, donc on soustrait les moyennes par usager μ_i à chaque Y_{ij}. On considère que

$$y_{i\cdot} \sim \mathcal{N}_N(0, C_{O(y_{i\cdot})O(y_{i\cdot})} + \lambda I) \qquad i = 1, ..., M, \tag{5.2}$$

avec $O(y_{i\cdot})$ les indices des films notés par l'usager i et $C_{O(y_{i\cdot})O(y_{i\cdot})}$ une matrice carré contenant les lignes et les colonnes associées à l'usager i.

La vraisemblance marginale $L(C, \lambda \mid y)$ est

$$L(C, \lambda \mid y) = \prod_{i=1}^{M} f(y_{i\cdot} \mid 0, C_{O(y_{i\cdot})O(y_{i\cdot})} + \lambda I). \tag{5.3}$$

Pour trouver le **maximum de vraisemblance**, on utilisera l'algorithme itératif E-M.

5.1 L'algorithme E-M

L'**algorithme E-M** est une technique d'optimisation par itérations pour trouver le maximum de vraisemblance, particulièrement pour les problèmes avec des données manquantes, qui fut proposé par (Dempster *et al.*, 1977).

Plutôt que de calculer la vraisemblance $L(\theta \mid y)$ avec des données manquantes, on augmente les données avec une variable latente non-observée x (d'où le $X = UV^t$ dans le modèle) pour avoir des données complètes (y, x). On regarde alors la vraisemblance comme la marginale de la loi conjointe avec les données augmentées

$$L(\theta \mid y) = g(y \mid \theta) = \frac{L(\theta \mid x, y)}{k(x \mid \theta, y)}.$$

avec $L(\theta \mid x, y) = f(x, y \mid \theta)$ et $k(x \mid \theta, y)$ la loi conditionnelle $X \mid y$.

En posant le log on obtient

$$\log L(\theta \mid y) = \log L(\theta \mid x, y) - \log k(x \mid \theta, y)$$

Bien que $L(\theta \mid y)$ ne dépende pas de X, individuellement les membres de droites en dépendent, on peut alors prendre l'espérance par rapport à $k(x \mid \theta', y)$, ce qui nous donne

$$\log L(\theta \mid y) = E(\log L(\theta \mid X, y) \mid \theta', y) - E(\log k(X \mid \theta, y) \mid \theta', y).$$

Pour trouver le maximum de vraisemblance, on veut maintenant maximiser

$$E(\log L(\theta \mid x, y) \mid \theta', y).$$

L'algorithme itératif comporte **deux phases**, la phase d'"espérance" et la phase de "maximisation". On fixe un $\theta^{(0)}$ et on crée une suite de $\theta^{(n)}$ jusqu'à convergence.

Dans la **phase d'espérance**, on calcule l'espérance de la Log vraisemblance complète

$$E(\log L(\theta \mid X, y) \mid \theta^{(n)}, y).$$

Dans la **phase de maximisation**, on calcule le θ qui maximise l'espérance de la Log vraisemblance trouvée dans la phase précédente

$$\theta^{(n+1)} = \underset{\theta}{\operatorname{argmax}}\, E(\log L(\theta \mid X, y) \mid \theta^{(n)}, y).$$

Dépendamment de la façon dont les paramètres sont actualisés, on n'est pas toujours obligé de calculer explicitement l'espérance à la première étape **E**. Comme l'espérance est calculée en fonction de $k(x \mid \theta', y)$, dans certains cas et grâce au théorème de factorisation (Casella et Berger, 2002), on peut seulement **calculer ses statistiques exhaustives**.

À l'aide des données trouvées en phase **M**, on reprend les calculs de la phase **E**.

Pour plus de détails sur les algortihmes E-M en général, voir (Lehmann et Casella, 2006).

5.2 Algorithmes E-M pour NPCA

Une application de l'algorithme E-M au modèle NPCA est décrite avec les deux phases suivantes.

E. Comme on travaille avec des lois Gaussiennes, plutôt que de calculer explicitement $E(\log L(\theta \mid x, y) \mid \theta^{(n)}, y)$, on calcule les statistiques exhaustives de la distribution marginale $f(x_{i\cdot} \mid y_{i\cdot}, C, \lambda)$, $i = 1, ..., M$

$$E(x_{i\cdot}) = C_{\cdot O(y_{i\cdot})}(C_{O(y_{i\cdot})O(y_{i\cdot})} + \lambda I)^{-1} y_{iO(y_{i\cdot})}$$

et

$$D_i = Cov(x_{i\cdot}) = C - C_{\cdot O(y_{i\cdot})}(C_{O(y_{i\cdot})O(y_{i\cdot})} + \lambda I)^{-1} C_{O(y_{i\cdot})\cdot}$$

M. Suite aux résultats obtenus à l'étape E, on actualise nos paramètres à l'aide de ces statistiques exhaustives.

$$C^{(n+1)} = C^{(n)} + \frac{1}{M}\sum_{i=1}^{M}\left[Cov(x_{i\cdot}) + E(x_{i\cdot})E(x_{i\cdot})^t\right]$$

et

$$\lambda^{(n+1)} = \lambda^{(n)} + \frac{1}{card(O(Y))}\sum_{(i,j)\in O(Y)}\{D_{ij} + [Y_{ij} - E(X_{ij})]^2\},$$

où D_{ij} est le j^{ieme} élément de la diagonale de $Cov(x_{i\cdot})$, donc la variance de X_{ij}.

Dans le but de simplifier l'algorithme et de le rendre plus rapide, on incorpore λI dans le C de tel sorte que $C \leftarrow C^* = VV^t + \lambda I$ et on initialise une matrice $B \in \mathbb{R}^{N\times N}$ avec des zéros pour sauvegarder l'information locale durant les calculs. De cette façon, l'étape **E** revient maintenant à calculer

$$B_{O(y_{i\cdot})O(y_{i\cdot})} = B_{O(y_{i\cdot})O(y_{i\cdot})} - G_i + z_{O(y_{i\cdot})O(y_{i\cdot})}z^t_{O(y_{i\cdot})O(y_{i\cdot})}$$

et

$$b_{O(y_{i\cdot})} = b_{O(y_{i\cdot})} + z_{O(y_{i\cdot})},$$

où $G_i = (C_{O(y_{i\cdot})O(y_{i\cdot})})^{-1}$ et $z_{O(y_{i\cdot})O(y_{i\cdot})} = G_i y^t_{iO(y_{i\cdot})}$. Il n'est donc plus nécessaire de calculer explicitement la matrice de covariance, ce qui permet de sauver beaucoup de temps, surtout sur de grandes bases de données.

Suite aux résultats obtenus à l'étape **E**, on actualise nos paramètres dans la nouvelle version de l'étape **M**,

$$C^{(n+1)} = C^{(n)} + \frac{1}{M}C^{(n)}BC^{(n)}$$

et

$$\mu^{(n+1)} = \mu^{(n)} + \frac{1}{M}b.$$

Après avoir effectué le nombre d'itérations maximales, on calcule la prédiction comme

$$P_{ij} = E(Y_{ij}) = C_{jO(y_{i\cdot})}(C_{O(y_{i\cdot})O(y_{i\cdot})})^{-1}(y_{O(y_{i\cdot})} - \mu_{O(y_{i\cdot})}) + \mu_j. \tag{5.4}$$

Cette version de l'algorithme se nomme **NPCA rapide** et c'est la version que nous utilisons dans nos simulations. Pour plus de détails sur les algorithmes E-M décrits ici, voir (Tipping et Bishop, 1999) et (Yu *et al.*, 2009).

5.3 Exemple 6×4

Voici les prédictions que donne l'algorithme NPCA rapide à l'exemple où $Y \in \mathbb{R}^{6\times4}$.

$$Y = \begin{bmatrix} 5 & 0 & 3 & 5 \\ 4 & 5 & 0 & 0 \\ 0 & 5 & 0 & 3 \\ 4 & 0 & 3 & 1 \\ 0 & 3 & 5 & 5 \\ 4 & 4 & 5 & 0 \end{bmatrix}$$

Après quelques itérations,

$$C = \begin{bmatrix} 0.8978562115 & -0.043451083491 & 0.064648219761 & 0.3296171588 \\ -0.04345108349 & 0.801101927273 & -0.306263405986 & -0.310107736 \\ 0.06464821976 & -0.306263405986 & 1.06564234724 & 0.7615329249 \\ 0.32961715887 & -0.3101077306 & 0.76153292495 & 2.727349198 \end{bmatrix}$$

$$\mu = (4.2501541333, 4.247380182, 4.237543768, 3.352934102)$$

et la matrice finale avec les prédictions sera

$$P = \begin{bmatrix} 5 & (4.49) & 3 & 5 \\ 4 & 5 & (3.93) & (2.98) \\ (4.21) & 5 & (3.93) & 3 \\ 4 & (4.66) & 3 & 1 \\ (4.45) & 3 & 5 & 5 \\ 4 & 4 & 5 & (3.81) \end{bmatrix}$$

Par exemple,

$$\begin{aligned} P_{12} &= C_{2O(y_{1\cdot})}(C_{O(y_{1\cdot})O(y_{1\cdot})})^{-1}(y_{1O(y_{1\cdot})} - \mu^t_{O(y_{1\cdot})} + \mu_2 \\ &= \begin{bmatrix} -0.0435 \\ -0.3063 \\ -0.3101 \end{bmatrix}^t \begin{bmatrix} 0.8979 & 0.0646 & 0.3296 \\ 0.0646 & 1.0656 & 0.7615 \\ 0.3296 & 0.7615 & 2.7273 \end{bmatrix}^{-1} \begin{bmatrix} 5 - 4.25 \\ 3 - 4.24 \\ 5 - 3.35 \end{bmatrix} + 4.25 \\ &= 4.49. \end{aligned}$$

5.4 Pseudo-code

Algorithme 4 NPCA rapide

Exigence : $Y, iter_{max}$

Créer : $C \in \mathbb{R}^{N \times N}, B \in \mathbb{R}^{N \times N}, b \in \mathbb{R}^{N}, \mu \in \mathbb{R}^{N}$

Initialise : $iter = 0, C$

Pour $n = 1, ..., iter_{max}$

    Réinitialise $B, b = 0$

    Pour $i = 1, ..., M$

        $G_i = C^{-1}_{O(y_{i\cdot})O(y_{i\cdot})}$

        $z_{O(y_{i\cdot})O(y_{i\cdot})} = G_i(y_{iO(y_{i\cdot})} - \mu_{iO(y_{i\cdot})})^t$

        $b_{O(y_{i\cdot})} = b_{O(y_{i\cdot})} + z_{O(y_{i\cdot})O(y_{i\cdot})}$

        $B_{O(y_{i\cdot})O(y_{i\cdot})} = B_{O(y_{i\cdot})O(y_{i\cdot})} - G_i + z_{O(y_{i\cdot})O(y_{i\cdot})} z^t_{O(y_{i\cdot})O(y_{i\cdot})}$

    Fin

    $C^{(n+1)} = C^{(n)} + \frac{1}{M} CBC$ **et** $\mu^{(n+1)} = \mu^{(n)} + \frac{1}{M} b$

Fin

Pour $i = 1, ..., M$ et $j = 1, ..., N$

    Si $Y_{ij} \in O(Y)^C$, $P_{ij} = C_{jO(y_{i\cdot})}(C_{O(y_{i\cdot})O(y_{i\cdot})})^{-1}(y_{O(y_{i\cdot})} - \mu_{O(y_{i\cdot})}) + \mu_j$

    Si $Y_{ij} \in O(Y)$, $P_{ij} = Y_{ij}$

Fin

Retourner P

APPROCHES BAYÉSIENNES

CHAPITRE VI

FACTORISATION DE MATRICE PROBABILISTE

L'algorithme **factorisation de matrice probabiliste** (PMF) est proposé par (Salakhutdinov et Mnih, 2008b). Cette méthode se veut une alternative aux autres factorisations de matrices et propose une mise à l'échelle en fonction du nombre d'observations disponibles.

On essaie de s'ajuster en fonction de l'uniformité de l'échantillon. Un échantillon où les usagers ont noté un nombre de films similaires serait considéré uniforme. À l'inverse, on aurait un échantillon où un usager aurait noté très peu de films comparativement à un autre usager qui aurait noté presque la totalité des films.

De manière similaire à l'algorithme NPCA, on définit un modèle linéraire avec un bruit gaussien. On a soustrait au préalable la moyenne générale $\overline{Y}$ à chaque entrée Y_{ij}.

$$Y_{ij} = u_{i\cdot} v_{j\cdot}^{t} + \varepsilon_{ij}, \qquad \textit{pour chaque } (i,j) \in O(Y). \tag{6.1}$$

Pour trouver les estimateurs des coefficients des matrices $U \in \mathbb{R}^{M \times K}$ et $V \in \mathbb{R}^{N \times K}$, l'algorithme PMF utilise le **maximum a posteriori** (MAP) décrit à la page 33.

6.1 Maximum a posteriori pour PMF

Dans le modèle PMF, on a besoin de la vraisemblance

$$L(U, V \mid y, \sigma^2) = \prod_{i=1}^{M} \prod_{j=1}^{N} \left[\mathcal{N}(Y_{ij} \mid u_{i\cdot} v_{j\cdot}^t, \sigma^2)\right]^{I_{ij}},$$

où $\mathcal{N}(\cdot \mid \mu, \sigma^2)$ est la fonction de densité d'une loi Gaussienne avec moyenne μ et variance σ^2, et $I_{ij} = \{\, 1 \quad \textit{pour chaque } (i,j) \in O(Y),\ 0 \quad \textit{sinon}\}$.

On choisit aussi des lois a priori sur les usagers et sur les films.

$$g(U \mid \sigma_U^2) = \prod_{i=1}^{M} \mathcal{N}_K(u_{i\cdot} \mid 0, \sigma_U^2)$$

et

$$h(V \mid \sigma_V^2) = \prod_{j=1}^{N} \mathcal{N}_K(v_{j\cdot} \mid 0, \sigma_V^2).$$

Trouver les estimateurs MAP pour les $u_{i\cdot}, v_{j\cdot}$,

$$\underset{u_{i\cdot}, v_{j\cdot}}{\operatorname{argmax}} \log L(U, V \mid Y, \sigma^2, \sigma_U^2, \sigma_V^2) g(U \mid \sigma_U^2) h(V \mid \sigma_V^2) \tag{6.2}$$

revient à calculer

$$\underset{u_{i\cdot}, v_{j\cdot}}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij} (Y_{ij} - u_{i\cdot} v_{j\cdot}^t)^2 + \frac{\lambda_U}{2} \sum_{i=1}^{M} \|u_{i\cdot}\|^2 + \frac{\lambda_V}{2} \sum_{j=1}^{N} \|v_{j\cdot}\|^2, \tag{6.3}$$

où $\lambda_U = \frac{\sigma^2}{\sigma_U^2}$ et $\lambda_V = \frac{\sigma^2}{\sigma_V^2}$, $I_{ij} = 1$ si $(i,j) \in O(Y)$ et $I_{ij} = 0$ sinon.

Trouver l'estimateur MAP sur les matrices U et V, contenant les facteurs latents des films et des usagers avec leurs hyperparamètres fixés, revient à minimiser la somme des erreurs carrées de la fonction objective avec des facteurs de régularisations quadratiques en (6.3). Ce modèle se veut une extension probabiliste du modèle SVD.

6.2 Algorithme du gradient pour PMF

Pour résoudre l'équation (6.3), on utilise l'algorithme du gradient tel que décrit à la section 3.1 page 21 et similaire à celui utilisé avec RegSVD à la section 3.2.

On calcule d'abord les erreurs entre les données de la matrice Y et celles prédites par le produit scalaire $P_{ij} = u_i v_j^t$. Une différence notable est que l'on soustrait initialement la moyenne globale à chaque donnée observée

$$R_{ij} = (Y_{ij} - \overline{Y}) - P_{ij}.$$

Les gradients de la fonction à optimiser sont

$$(\nabla_U)_{ik} = R_{ij} V_{jk} - \lambda_U U_{ik}$$

et

$$(\nabla_V)_{jk} = R_{ij} U_{ik} - \lambda_V V_{jk}.$$

Ensuite on actualise les données des matrices de prédictions U et V avec le taux d'apprentissage t_a et les paramètres de régularisation λ_U et λ_V comme suit

$$U^{(n+1)} = U^{(n)} + t_a \nabla_U$$

et

$$V^{(n+1)} = V^{(n)} + t_a \nabla_V.$$

On poursuit l'entraînement des paramètres et l'actualisation des matrices jusqu'à ce que l'erreur totale soit inférieure à un ϵ choisi d'avance, que l'erreur soit supérieure à l'erreur de l'itération précédente ou jusqu'au nombre d'itérations maximales.

Les matrices U et V sont donc actualisées avec les données contenues dans la matrice d'entraînement Y.

On peut alors trouver les vides qui nous intéressent par le produit scalaire des vecteurs d'intérêts pour un usager et un film en particulier et en ajoutant la moyenne globale soustraite au départ comme suit

$$P_{ij} = u_{i.} v_{j.}^t + \overline{Y}. \tag{6.4}$$

6.3 Exemple 6×4

Voici les prédictions que donnera PMF à l'exemple où $Y \in \mathbb{R}^{6\times 4}$.

$$Y = \begin{bmatrix} 5 & 0 & 3 & 5 \\ 4 & 5 & 0 & 0 \\ 0 & 5 & 0 & 3 \\ 4 & 0 & 3 & 1 \\ 0 & 3 & 5 & 5 \\ 4 & 4 & 5 & 0 \end{bmatrix}$$

On a $\overline{Y} = 4$.

Après quelques itérations avec $K = 3$,

$$U = \begin{bmatrix} 1.250305808532 & 0.456810792539 & -0.0803847087761 \\ 0.3974320969755 & -0.2428154601022 & -0.4279566174811 \\ -0.1688612442272 & -0.950944493683 & -0.732813815204 \\ -1.394667652751 & -2.30783038495 & -1.425306995467 \\ 0.0964993324344 & 1.268301384449 & 1.010051490343 \\ -0.518429235422 & 0.0683854951797 & 0.1659363956027 \end{bmatrix}$$

$$V^t = \begin{bmatrix} 0.35624743105 & 0.9459631994 & -1.5122135365 & -1.40891267862 \\ 0.22323079062 & -0.07659793488 & -0.06316900982 & 0.89886909547 \\ 0.1642802027 & -0.5705064246 & 0.10627855681 & 0.436564366156 \end{bmatrix}$$

et la matrice finale avec les prédictions sera

$$P = \begin{bmatrix} 5 & (5.19) & 3 & 5 \\ 4 & 5 & (3.37) & (3.04) \\ (3.61) & 5 & (4.24) & 3 \\ 4 & (3.67) & 3 & 1 \\ (4.48) & 3 & 5 & 5 \\ 4 & 4 & 5 & (4.86) \end{bmatrix}$$

Par exemple,

$$\begin{aligned} P_{12} = u_1.v_{2.}^t &= (1.2503, 0.4568, -0.0803)(0.946, -0.0766, -0.5705)^t + 4 \\ &= 5.1936 \end{aligned}$$

6.4 Pseudo-code

Algorithme 5 PMF

Exigence : $Y,\ t_a,\ \lambda_U,\ \lambda_V,\ iter_{max}$

Créer : $M \in \mathbb{R}^{M \times k},\ U \in \mathbb{R}^{M \times K},\ V \in \mathbb{R}^{N \times K},\ \Theta_U \in \mathbb{R}^n,\ \Theta_V \in \mathbb{R}^m$

Initialise : $iter = 0,\ Err = 10000$

Pour $iter = 1, ..., iter_{max}$

Échantillonner chaque $U_{ik} \sim 0.1\mathcal{N}(0,1)$ **et** $V_{jk} \sim 0.1\mathcal{N}(0,1)$

    $Err = Er$ **puis** $Er = 0$

    Pour chaque $i = 1, ..., M$ et $j = 1, ..., N$ $\quad (i,j) \in O(Y)$

        $M_{ij} = u_{i\cdot} v_{j\cdot}^t$

        $R_{ij} = (Y_{ij} - \overline{Y}) - M_{ij}$

        $Er = Er + R_{ij}^2 + \frac{1}{2}\lambda_U \sum_{i=1}^{M} \|u_{i\cdot}\|^2 + \frac{1}{2}\lambda_V \sum_{j=1}^{N} \|v_{i\cdot}\|^2$

        Pour chaque $k = 1, ..., K$

            $\nabla_{U_{ik}} = 2R_{ij}V_{jk} + \lambda_U U_{ik} v$ **et** $\nabla_{V_{jk}} = 2R_{ij}U_{ik} + \lambda_V V_{jk}$

    Si $Err > Er$ **alors** $U = U - t_a \nabla_U$ **et** $V = V - t_a \nabla_V$

Fin si $iter = iter_{max}$, $Er < \epsilon$ ou $Err < Er$

Pour $i = 1, ..., M$ **et** $j = 1, ..., N$

    Pour chaque $(i,j) \in O(Y)^c$ $\quad P_{ij} = u_{i\cdot} v_{j\cdot}^t + \overline{Y}$

Retourner P

NOTIONS UTILES : MÉTHODES DE MONTE-CARLO

Les méthodes de Monte-Carlo sont des outils statistiques qui permettent entre autres d'évaluer de grandes intégrales à l'aide de simulations sur des lois probabilistes. Certaines de ces méthodes se basent sur des chaînes de Markov.

Une **chaîne de Markov** est un type de processus stochastique. On peut la voir comme évoluant dans le temps.

Définition 1 (Chaîne de Markov) *Soit X_n avec $n \geq 1$, une suite de variables aléatoires qui prennent leurs valeurs dans $\mathcal{X}$. On appelle cette suite une chaîne de Markov s'il existe un noyau de transition K tel que pour chaque $B \in \mathcal{X}$ et chaque $x_1, ..., x_{n-1}, x \in \mathcal{X}$*

$$P(X_{n+1} \in B \mid X_1 = x_1, ..., X_n = x) = K(x, B) = P(X_{n+1} \in B \mid X_n = x).$$

On remarque que cette dernière probabilité est indépendante de la valeur de n. Certains auteurs appellent cela une chaîne de Markov homogène.

Les **méthodes de Monte-Carlo par chaînes de Markov (MCMC)** sont des techniques qui produisent une chaîne de Markov X_t dont la distribution stationnaire est f.

Si on a

$$J = E_f(h(x)) = \int h(x)f(x)dx.$$

alors

$$\hat{J}_T = \frac{1}{T}\sum_{t=1}^{T} h(x_t).$$

Autrement dit, à l'aide des états $x_t, t = 1, .., T$, générés par la chaîne de Markov, on approxime l'espérance d'une fonction $h(x)$ en faisant la moyenne échantillonnale $\hat{J}_T$.

Par la loi des grands nombres, $\hat{J}_T \to J$ lorsque $n \to \infty$.

L'**échantillonnage de Gibbs** est une méthode MCMC qui produit une chaîne de Markov dont la distribution stationnaire est la vraie distribution conjointe. Elle est similaire à l'algorithme E-M, mais au lieu de maximiser les distributions conditionnelles, elle échantillonne à partir de celle-ci. (Friedman *et al.*, 2001)

Soit X et Y, deux variables aléatoires dont les densités conditionnelles sont $f_{X|Y}$ et $f_{Y|X}$.

Supposons qu'il soit facile de simuler de ces dernières, alors l'algorithme de Gibbs consiste à générer des observations successives $x_1, y_1, x_2, y_2, \ldots$

Par exemple,

$$X_{i+1} \mid Y_i = y_i \sim f_{X|Y}(\cdot \mid Y = y_i),$$

$$Y_{i+1} \mid X_{i+1} = x_{i+1} \sim f_{Y|X}(\cdot \mid X = x_{i+1}),$$

ainsi de suite.

L'algorithme de **recuit simulé** sert à approximer le maximum ou le mode d'une fonction. Il est inspiré de la physique, particulièrement de la métallurgie, et fait intervenir une température. Intuitivement, lorsque la température est grande, le paramètre a tendance à varier facilement et à se promener dans la fonction. Lorsque l'on fait baisser la température, il se stabilise à un maximum.

À chaque itération, on échantillonne le paramètre à partir de sa loi instrumentale

$$\theta' \sim \pi(\theta).$$

À la différence d'une chaîne de Markov homogène, on n'accepte pas nécessairement le nouvel état du paramètre. On a plutôt un taux d'acceptation en lien avec la température T_n, qui peut dépendre de l'itération n. On a alors

$$\theta^{(n+1)} = \begin{cases} \theta' & \text{avec} \quad p = min(1, exp(\frac{p(\theta^{(n)}) - p(\theta')}{T_n})) \\ \theta^{(n)} & \text{sinon.} \end{cases}$$

Par exemple, si on veut que la température dépende de l'itération n, on pourrait avoir $T_n = 1/\log n$, $n > 1$.

Pour plus de détails sur les méthodes de Monte-Carlo, voir l'ouvrage de (Robert et Casella, 1999).

CHAPITRE VII

FACTORISATION DE MATRICE PROBABILISTE BAYÉSIENNE

Comme son nom l'indique, l'algorithme de **factorisation de matrice probabiliste bayésienne** (BPMF) découle de PMF. Il est proposé par (Salakhutdinov et Mnih, 2008a).

Comme pour l'algorithme PMF, on pose des lois a priori pour les usagers et pour les films :

$$p(U \mid \mu_U, \Lambda_U) = \prod_{i=1}^{N} \mathcal{N}_K(u_{i\cdot} \mid \mu_U, \Lambda_U^{-1}) \tag{7.1}$$

et

$$p(V \mid \mu_V, \Lambda_V) = \prod_{j=1}^{M} \mathcal{N}_K(v_{i\cdot} \mid \mu_V, \Lambda_V^{-1}). \tag{7.2}$$

La différence intervient lorsqu'on ajoute une loi a priori Gaussienne-Wishart sur les hyperparamètres $\Theta_U = \{\mu_U, \Lambda_U\}$ et $\Theta_V = \{\mu_V, \Lambda_V\}$ tel que

$$p(\Theta_U \mid \Theta_0) = p(\mu_U \mid \Lambda_U)p(\Lambda_U) = \mathcal{N}(\mu_U \mid \mu_0, (\beta_0\Lambda_U)^{-1})\mathcal{W}(\Lambda_U \mid W_0, \nu_0) \tag{7.3}$$

et

$$p(\Theta_V \mid \Theta_0) = p(\mu_V \mid \Lambda_V)p(\Lambda_V) = \mathcal{N}(\mu_V \mid \mu_0, (\beta_0\Lambda_V)^{-1})\mathcal{W}(\Lambda_V \mid W_0, \nu_0), \tag{7.4}$$

où $\mathcal{W}$ est la loi Wishart avec ν_o degrés de liberté et une matrice d'échelle W_0. Pour simplifier les choses, $\Theta_0 = \{\mu_0, \nu_0, W_0\}$, $\mu_0 = 0$, $\nu_0 = K$ et W_0 sera la matrice

identité dans les deux cas, que ce soit pour les films ou pour les usagers. On travaille encore avec des données centrées au préalable.

Pour trouver nos matrices finales U et V, on utilise une méthode de **Monte-Carlo par chaînes de Markov** (MCMC) tel que décrit à la page 53. On fait un échantillonnage de Gibbs à partir des lois de probabilité. Dans ce cas-ci, comme on utilise des lois conjuguées, il est plus facile de faire l'échantillonnage.

7.1 Échantillonnage de Gibbs pour BPMF

L'**échantillonnage de Gibbs** pour BPMF débute avec les matrices U et V obtenues de la méthode PMF vue précédemment. On choisit d'abord le nombre de cycles G et on initialise $\mu_0 = 0$, $\nu_0 = K$ et W_0 comme la matrice identité. La matrice de prédiction finale est calculée à la fin de la chaîne, avec les vecteurs $u_{i\cdot}^f, v_{j\cdot}^f$ obtenus à la fin de la dernière étape de l'échantillonnage Gibbs. À noter que la notation liée aux itérations est en exposant pour ne pas confondre avec la notation liée aux éléments des matrices.

Pour chacune de nos étapes, on procède d'abord à un échantillonnage des hyper-paramètres à partir des matrices U, V de l'étape précédante, avec

$$\Theta_U^k \sim p(\Theta_U \mid U^k, \Theta_0)$$

et

$$\Theta_V^k \sim p(\Theta_V \mid V^k, \Theta_0).$$

Ensuite, on procéde à l'échantillonnage des nouveaux vecteurs des matrices U, V tel que

$$u_{i\cdot}^{k+1} \sim p(u_{i\cdot} \mid Y, V^k, \Theta_U^t)$$

et

$$v_{j\cdot}^{k+1} \sim p(v_{j\cdot} \mid Y, U^k, \Theta_V^t),$$

et ce pour tous les $i = 1, ..., M$ et les $j = 1, ..., N$.

On poursuit l'entraînement des paramètres et l'actualisation des matrices jusqu'au nombre d'itérations maximales.

Les $u_{i\cdot}^g, v_{j\cdot}^g$ sont sauvegardés après chaque cycle complet de l'échantillonnage de Gibbs.

On calcule la prédiction finale comme suit

$$P_{ij} = \frac{1}{G} \sum_{g=1}^{G} u_{i\cdot}^g v_{j\cdot}^g + \overline{Y}. \tag{7.5}$$

7.2 Exemple 6×4

Voici les prédictions que donnera BPMF à l'exemple où $Y \in \mathbb{R}^{6\times 4}$.

$$Y = \begin{bmatrix} 5 & 0 & 3 & 5 \\ 4 & 5 & 0 & 0 \\ 0 & 5 & 0 & 3 \\ 4 & 0 & 3 & 1 \\ 0 & 3 & 5 & 5 \\ 4 & 4 & 5 & 0 \end{bmatrix}$$

Après quelques itérations avec $K = 3$, la matrice finale avec les prédictions sera

$$U = \begin{bmatrix} 0.4193798540502 & -0.204787893906 & -0.154874230549 \\ -0.2058562450807 & 0.831126381014 & -0.0509253596661 \\ -1.480881580327 & 1.476546274727 & -0.1313209146637 \\ -0.293572003342 & 1.739870126974 & -1.403888961498 \\ -0.1947859252564 & -2.28477489169 & 1.39713226344 \\ -0.073872189881 & -1.11926583911 & 0.656163422034 \end{bmatrix}$$

$$V^t = \begin{bmatrix} 0.47203699146 & -0.07853949266 & -0.7522026847 & 0.65365469 \\ 0.003527846597 & 0.5408514302 & 0.09991694777 & -0.5424354892 \\ -0.3270477388 & -0.10206512203 & 0.9120933406 & -0.286983085 \end{bmatrix}$$

$$P = \begin{bmatrix} 5 & (3.87) & 3 & 5 \\ 4 & 5 & (4.19) & (3.43) \\ (3.35) & 5 & (5.14) & 3 \\ 4 & (5.11) & 3 & 1 \\ (3.44) & 3 & 5 & 5 \\ 4 & 4 & 5 & (4.37) \end{bmatrix}$$

Par exemple,

$$\begin{aligned} P_{12} = u_{1.} v_{2.}^t + \overline{Y} &= (0.4194, -0.2048, -0.1549)(-0.0785, 0.5409, -0.1021)^t + 4 \\ &= 3.8721 \end{aligned}$$

7.3 Pseudo-code

Algorithme 6 BPMF

Exigence : $Y,\ \Theta_0,\ G,\ iter_{max}$

Créer : $P \in \mathbb{R}^{M \times N},\ U \in \mathbb{R}^{M \times K},\ V \in \mathbb{R}^{N \times K},\ \Theta_U \in \mathbb{R}^{M \times 2},\ \Theta_V \in \mathbb{R}^{N \times 2}$

Initialise : $iter = 0,\ moy = \overline{Y}$

Initialise : $U,\ V$ avec le modèle PMF

Pour $g = 1, ..., G$

  Pour $k = 1, ..., iter_{max}$

    Échantillonner $\Theta_U^k \sim p(\Theta_U \mid U^k, \Theta_0)$

    Échantillonner $\Theta_V^k \sim p(\Theta_V \mid V^k, \Theta_0)$

    Pour chaque $i = 1, ..., M$

      Échantillonner $u_{i\cdot}^{k+1} \sim p(u_{i\cdot} \mid Y, V^k, \Theta_U^t)$

    Pour chaque $j = 1, ..., N$

      Échantillonner $v_{j\cdot}^{k+1} \sim p(v_{j\cdot} \mid Y, U^{k+1}, \Theta_V^t)$

  Fin

$u_{i\cdot}^g = u_{i\cdot}^k$

$v_{j\cdot}^g = v_{j\cdot}^k$

Fin

Pour $i = 1, ..., M$ **et** $j = 1, ..., N$

  Pour chaque $Y_{ij} \in O(Y)^c$, $P_{ij} = P_{ij} = \frac{1}{G} \sum_{g=1}^{G} u_{i\cdot}^g v_{j\cdot}^g$

  Pour chaque $Y_{ij} \in O(Y)$, $P_{ij} = Y_{ij}$

Retourner P

NOTIONS UTILES : PERMUTATIONS ET RANGS COMPATIBLES

Dans cette section, nous introduisons la notion de permutation et de rangs compatibles qui seront utiles dans les prochains chapitres.

Définition 2 (Permutation) *Soit $\mathcal{S}_n$ le groupe de permutation de l'ensemble $N = 1, ..., n$. Une permutation $\sigma \in \mathcal{S}_n$ est une bijection de N vers lui-même, $\sigma = (\sigma(1), ..., \sigma(n))$. On pose aussi la permutation identité $e = (1, 2, 3..., n)$. Il y a $n!$ éléments dans $\mathcal{S}_n$.*

Par exemple, si nous prenons les permutations des rangs dans $\mathcal{S}_3$, nous obtenons les $3! = 6$ possibilités suivantes $(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2)(3,2,1)$.

Définition 3 (Rangs) *Soit $D_n = \{x \in \mathbb{R}^n \mid x_{(1)} < ... < x_{(n)}\}$ l'ensemble des vecteurs dans $\mathbb{R}^n$ qui n'ont pas d'égalité. Le vecteur de rang associé à $x \in D_n$ est défini comme*

$$rang(x_i) = \sum_{j \in N} \mathbb{1}(x_i \geq x_j), \qquad i \in N.$$

On peut voir le rang d'un objet comme le nombre d'éléments plus petits ou égaux à celui-ci.

Comme nous travaillons avec des notes de films, nous modifions les rangs pour que la meilleure note corresponde au rang 1, de cette façon

$$r_x(i) = n + 1 - rang(x_i),$$

$$r_x = (r_x(1), ..., r_x(n)).$$

On a $r_x \in \mathcal{S}_n$ pour tout $x \in D_n$. Dans l'exemple de 10 films, le rang de la plus petite note est égal à 10 et le rang de la plus grande note est égal à 1.

On voudra avoir une compatibilité entre des rangs incomplets r_y^* et des rangs complets r_x. C'est pourquoi on introduit la notion de **rangs compatibles** entre r_x et r_y^*.

Définition 4 (Rangs compatibles) *Soit l'ensemble des rangs compatibles*

$$\mathcal{RC}(r_y^*, M^*) = \{r_x \in \mathcal{S}_n : m + 1 - rang(r_x(i_1), ..., r_x(i_m)) = r_y^*\}$$

en tenant compte de la sous-permutation r_y^* pour les $M^* = \{i_1, ..., i_m\}$ films inclus dans N tel que

$$r'_x = r_y^* \circ (r_x^*)^{-1} \in \mathcal{RC}(r_y^*, M^*),$$

avec $\circ$ une composition de fonction tel que $r \circ g = r(g(x))$.

Par exemple, si on prend $r_y^* = (1, 4, 3, 2)$ avec $M^* = \{3, 5, 6, 7\}$, alors r_x doit devenir de la forme

$$r'_x = (-, -, 1, -, 4, 3, 2).$$

Si $r_x = (2, 1, 5, 3, 4, 7, 6)$ alors $r_x^* = (5, 4, 7, 6)$ et devient alors

$$r_x^* = (1, 4, 3, 2) \circ (5, 4, 7, 6)^{-1} = (1, 4, 3, 2) \circ (4, 5, 6, 7) = (4, 7, 6, 5).$$

et r_x devient

$$r'_x = (2, 1, \mathbf{4}, 3, \mathbf{7}, \mathbf{6}, \mathbf{5}) \in \mathcal{RC}(r_y^*, M^*).$$

CHAPITRE VIII

PRÉVISION BAYÉSIENNE DE RANGS BIVARIÉS

On présente ici un nouvel algorithme de prévision bayésienne de rangs bivariés (BBR) proposé par (Guillotte *et al.*, 2018). C'est la méthode que l'on cherche à comparer avec celles des chapitres précédents.

Dans la plupart des systèmes de recommandation, les films ont des notes et celles-ci sont transformées en rangs pour pouvoir proposer les meilleurs films à l'usager. On peut facilement imaginer que les films sont originalement en rangs et qu'on puisse faire la prévision directement sur ses rangs. On peut alors voir les notes (qui sont observées dans les méthodes précédentes) comme des variables latentes (non-observées dans notre modèle).

Comme on cherche à prédire les rangs des films que la personne n'a pas vu, on introduit les rangs d'un expert. Cet expert peut être un critique de film ou un site qui propose des classements et des notes de films tel que *Imdb* (www.imdb.com) et *Rotten tomatoes* (www.rottentomatoes.com).

On suppose donc que les rangs proviennent de notes que l'on ne voit pas et que celles-ci sont continues et on pose X les notes de l'expert et Y les notes de l'usager. Le vecteur aléatoire (X, Y) a une fonction de répartition conjointe H et selon le théorème de Sklar (Nelsen, 1999) on sait qu'il existe une copule unique C qui est la

fonction de répartition du vecteur aléatoire $(U, V) = (F(X), G(Y))$. La copule sert à capturer le lien (la dépendance) entre les goûts de l'expert et ceux de l'usager.

Supposons que l'usager ait vu tous les films ordonnés par l'expert, on pourrait alors observer $R_X = (r_X(1), ..., r_X(n))$ et $R_Y = (r_Y(1), ..., r_Y(n))$, les rangs aléatoires donnés respectivement par l'expert et l'usager, où $R_X = n + 1 - rang(X)$, de même pour R_Y.

En pratique, l'usager n'a pas vu tous les films. On n'observe pas R_y (qui serait une permutation de $\mathcal{S}_n$ sur n films) mais plutôt R_y^* sur un sous-ensemble de m films qu'on note M^*. R_y^* est une permutation de $\mathcal{S}_m$, soit les rangs que l'usager a donné aux m films. On va se servir des rangs donnés, de la copule C et d'une approche bayésienne pour faire la prévision.

8.1 Vraisemblance basée sur les rangs

Pour l'instant, continuons à regarder les rangs comme si on observait tout. Puisque nous nous intéressons uniquement au rang, nous montrons à l'aide du lemme suivant qu'on peut supposer que les notes latentes de l'expert et de l'usager sont marginalement de loi uniformes $(0, 1)$.

Lemme 2 *Si une variable aléatoire X possède une fonction de répartition continue alors $U = F(X)$ a les mêmes rangs que X avec probabilité 1, c'est-à-dire*

$$P(R_X = R_U) = 1. \tag{8.1}$$

Preuve. Sachant que F est continue et que $X_1, ..., X_n$ sont iid de distribution F, nous avons que les $U_i = F(X_i)$, $i = 1, ..., n$, sont iid $U(0, 1)$ et pour $i \neq j$, $P(X_i = X_j) = 0 = P(F(X_j) = F(X_j))$.

En particulier, nous avons que $P(R_X \in \mathcal{S}_n) = 1 = P(R_{F(X)} \in \mathcal{S}_n)$, où $R_{F(X)} =$

$n + 1 - rang(F(X_1), ..., F(X_n))$, avec $\mathcal{S}_n$ l'ensemble des permutations possibles, qu'on a définies à la page 61.

En tenant compte de ces observations et de la monotonie de F, nous avons que pour chaque $r \in \mathcal{S}_n$

$$\begin{aligned} P(R_X = r, R_{F(X)} = r) &= P(X_{r^{-1}(1)} > \cdots > X_{r^{-1}(n)}, F(X_{r^{-1}(1)}) > \cdots > F(X_{r^{-1}(n)})) \\ &= P(X_{r^{-1}(1)} > \cdots > X_{r^{-1}(n)}, F(X_{r^{-1}(1)}) \geq \cdots \geq F(X_{r^{-1}(n)})) \\ &= P(X_{r^{-1}(1)} > \cdots > X_{r^{-1}(n)}) \\ &= P(R_X = r). \end{aligned}$$

En faisant la somme des probabilités sur l'ensemble des $r \in \mathcal{S}_n$ on obtient $P(R_X = R_{F(X)}) = 1$. □

Étant donné une copule C et que $(X, Y) \sim C$, on cherche maintenant à calculer la probabilité de rang conjoint

$$P_C(R_X = r_x, R_Y = r_y) = P(R_X = r_x, R_Y = r_y \mid C), \qquad r_x, r_y \in \mathcal{S}_n,$$

Ceci pour évaluer la vraisemblance de la copule C étant donné les rangs observés r_x, r_y. Dans la notation usuelle on peut écrire

$$L(C \mid R_X = r_x, R_Y = r_y) = P_C(R_X = r_x, R_Y = r_y). \tag{8.2}$$

Sauf dans des cas simples, voir (Guillotte *et al.*, 2018), l'évaluation de (8.2) est difficile analytiquement. Il faut donc trouver une manière numérique ou stochastique. Pour ce faire, il est commode de définir une statistique de rang

$$S(X, Y) = R_Y \circ R_X^{-1}.$$

Cette statistique est exhaustive pour la copule C dans notre approche.

En effet,

$$\begin{aligned} L(C \mid R_X = r_x, R_Y = r_y) &= P_C(R_X = r_x, R_Y = r_y) \\ &= \frac{1}{n!} P_C(R_X = e, R_Y = r_y \circ r_x^{-1}) \\ &= \frac{1}{n!} P_C(S = r_y \circ r_x^{-1}), \end{aligned}$$

avec la permutation identité $e = (1, 2, 3..., n)$. Posons $s = r_y \circ r_x^{-1}$, on a que

$$P_C(S = s) = \frac{1}{n!} E\left(\prod_{i=1}^{n} c\left(\sum_{j=1}^{i} W_{1j}, \sum_{j=1}^{s(i)} W_{2j} \right) \right), \quad s \in \mathcal{S}_n. \tag{8.3}$$

avec $(W_{11}, ..., W_{1N})$ et $(W_{21}, ..., W_{2N})$, deux vecteurs indépendants suivant une loi de Dirichlet et c la densité de la copule C. À noter que les deux sommes dans le produit sont croissantes.

On appelle $L(C \mid R_X = r_x, R_Y = r_y)$ la vraisemblance basée sur les rangs. Pour calculer cette espérance, on utilise la loi des grands nombres. On va générer les vecteurs W_1, W_2, calculer le produit et prendre la moyenne échantillonnale.

La prédiction des rangs complets de l'usager est basée sur le mode de la distribution a posteriori

$$\hat{r}_y = \underset{r_y}{\operatorname{argmax}} P_C(R_Y = r_y \mid R_X = r_x, R_Y^* = r_y^*). \tag{8.4}$$

On utilise alors ces rangs pour lui proposer les meilleurs films.

8.2 Description de l'algorithme BBR

Voici maintenant les différentes étapes des algorithmes utilisés. Pour calculer numériquement l'équation (8.3), on utilise la méthode de Monte-Carlo de recuit si-

mulé (décrit à la page 54) avec comme point de départ un rang $S' \in \mathcal{RC}(S^*, M^*)$ trouvé avec un algorithme de rang compatible.

La forme que prend c à l'équation (8.3) dépend de la famille de copules choisie et de la loi a priori du paramètre pour une famille paramétrique.

Dans nos simulations, nous utilisons une copule de la **famille gaussienne** qui a la forme suivante

$$c_\theta(u,v) = \frac{1}{2\pi\sqrt{1-\theta^2}} exp\left(-\frac{u^2+v^2-2\theta uv}{2(1-\theta^2)}\right) \qquad (u,v) \in (0,1)^2,\ \theta \in (-1,1).$$

Pour la loi a priori sur le paramètre θ, il est convenable d'utiliser une transformation de la loi Bêta telle que $\theta = 2T - 1$ avec $T \sim Beta(\alpha, \beta)$.

8.2.1 Algorithme de rang compatible

L'algorithme de rang compatible consiste en une chaîne de Markov.

Soit $S \in \mathcal{RC}(S^*, M^*)$ l'état présent et compatible de la chaîne.

Pour reprendre l'exemple avec $n = 7$, on pourrait commencer avec

$$S = (2,1,4,3,7,6,5).$$

À chaque itération, on choisit au hasard entre les deux étapes suivantes,

A. On échange les rangs de deux objets non-notés par l'usager.

Pour ce faire, on tire une paire $(i,j) \in M^c$ avec $i < j$ avec une probabilité de $\frac{1}{\binom{n-m}{2}}$

On procèdera ensuite à l'échange,

$$S'(i) = S(j), \quad S'(j) = S(i) \quad et \quad S'(t) = S(t), \quad \forall t \in \{1,...,n\} \setminus \{i,j\}$$

Par exemple, si on tire la paire $(1,2)$ alors $S' = (\mathbf{1},\mathbf{2},4,3,7,6,5)$.

B. On échange les rangs d'un objet noté et d'un objet non-noté par l'usager

De façon similaire à A, on tire une paire (l,j) avec $l \in \{1,...,m\}$ et $j \in M^c$ avec une probabilité de $\frac{1}{m(n-m)}$.

On procède ensuite à l'échange,

$$S'(j) = S'(i_l), \quad S'(i_l) = S'(j) \quad et \quad S'(t) = S(t), \quad \forall t \in \{1,...,n\} \setminus \{l,j\}.$$

On réarrange ensuite les $S(i_k)$ pour qu'ils soient compatibles avec les rangs de l'usager.

Par exemple, si on tire la paire $(1,5)$, $S' = (\mathbf{7},2,4,3,\mathbf{1},6,5)$. On les réarrange pour qu'ils soient compatibles à $r_y^* = (1,4,3,2)$, alors

$$S' = (7,2,\mathbf{1},3,\mathbf{6},\mathbf{5},\mathbf{4}).$$

8.2.2 Algorithme de recuit simulé

Soit $S \in \mathcal{RC}(S^*, M^*)$, l'état de départ de la chaîne, qu'on choisit comme les rangs compatibles de l'expert.

A. On tire d'abord un S' à l'aide de l'algorithme de base vu précédemment.

Par exemple, $S' = (7,2,\mathbf{1},3,\mathbf{6},\mathbf{5},\mathbf{4})$.

B. Pour les itérations $k = 1,...,K$, on échantillonne $(W_{1l},...,W_{(n+1)l}) = (W'_{1l},...,W_{nl}, 1 - \sum_{j=1}^{n} W')$, pour $l = 1,2$ d'une loi de Dirichlet(1,...,1) et on tire indépendamment $\theta' \sim \pi$. On évalue alors

$$p_k(S') = \frac{1}{n!} \prod_{i=1}^{n} c_\theta \left(\sum_{j=1}^{i} W', \sum_{j=1}^{S'(i)} W'_{2,j} \right).$$

Dans l'exemple, pour la première itération, si on tire les variables de Dirichlet suivantes

$$W_1' = (0.0225, 0.2723, 0.4338, 0.1145, 0.0244, 0.1221, 0.0104),$$

$$W_2' = (0.0016, 0.2540, 0.0942, 0.1807, 0.0306, 0.0252, 0.4136),$$

et on tire $\theta' = 0.6810$. On a

$$p_1(S') = \frac{1}{7!}\prod_{i=1}^{7} c_\theta \left(\sum_{j=1}^{i} W_{1j}', \sum_{j=1}^{S'(i)} W_{2j}' \right) = \frac{1}{7!} 8.9253 \times 10^{-7}..$$

Après avoir fait les K itérations, on calcule $\hat{p}(S') = \frac{1}{K}\sum_{i=1}^{K} p_k(S')$

C. Une fois $\hat{p}(S')$ calculé, on procédera au changement $S_{t+1} = S'$ avec une probabilité de

$$\min\left(1, \exp\left(\frac{\hat{p}(S') - \hat{p}(S_t)}{T_t}\right)\right), \qquad T_t = 1/\log t, \quad t > 1.$$

Autrement, la chaîne restera inchangée à cette étape $S_{t+1} = S_t$.

8.3 Pseudo-codes

Algorithme 7 BBR Échantillonnage de rangs compatibles

En parallèle pour chaque usager

Exigence : $r_x \in \mathbb{R}^n$, $r_y^* \in \mathbb{R}^m$, $M^* \in \mathbb{R}^m$

Créer : $S \in \mathbb{R}^n, S' \in \mathbb{R}^n$

On commence avec un $S \in \mathcal{RC}(S^*, M^*)$

Pour chaque itération

Choisir au hasard entre A et B

Étape A

    On tire $(i,j) \in M^c$, $i < j$ avec $p = \frac{1}{\binom{n-m}{2}}$

    $S'(i) = S(j), S'(j) = S(i)$ et $S'(t) = S(t) \forall t \in \{1, ..., n\} \setminus \{i, j\}$

Étape B

    On tire (l,j), $l \in \{1, ..., m\}$ et $j \in M^c$ avec $p = \frac{1}{m(n-m)}$

    $S'(j) = S'(i_l)$ et $S'(i_l) = S'(j)$

    On rend compatible $S(i_k)$

Retourner à la fin S'

Algorithme 8 BBR Recuit simulé

En parallèle pour chaque usager

Exigence : $r_x \in \mathbb{R}^n$, $r_y^* \in \mathbb{R}^m$, $M^* \in \mathbb{R}^m$

Créer : $S \in \mathbb{R}^n, S' \in \mathbb{R}^n$

On commence avec $S \in \mathcal{RC}(S^*, M^*)$ les rangs compatibles de l'expert

Pour chaque itération

Étape 1

On tire S' avec l'algorithme de base

Étape 2

    Pour $k = 1, ..., K$

    Étape 2.1

        Échantillonner $(W_1, ..., W_{n+1}) \sim Dirichlet(1, ..., 1)$, pour $l = 1, 2$

        Échantilloner $\theta' \sim \pi(\theta)$

        $p_k(S') = \frac{1}{n!} \prod_{i=1}^{n} c_\theta \left(\sum_{j=1}^{i} W' \sum_{j=1}^{S'(i)} W'_{2,j} \right)$

    Étape 2.2

        $\hat{p}(S') = \frac{1}{K} \sum_{i=1}^{K} p_k(S')$

Étape 3

    $T_t = \frac{1}{\log t}$

    $S_{t+1} = S'$ avec $p = \min \left(1, exp \left(\frac{\hat{p}(S') - \hat{p}(S_t)}{T_t} \right) \right)$

    $S_{t+1} = S_t$ sinon

Retourner à la fin S_t

QUATRIÈME PARTIE
SIMULATIONS ET CONCLUSION

CHAPITRE IX

SIMULATIONS

Nous voulons comparer les algorithmes présentés dans les chapitres précédents. Nous présentons la méthodologie utilisée ainsi que les résultats obtenus lors des simulations.

9.1 L'outil PREA

Les simulations ont débuté avec la librairie PREA en Java. **PREA** est un **outil de recommandation d'algorithmes personnalisé**. Cette librairie sert à comparer certains algorithmes du marché pour choisir le type de système de recommandation qu'on utilisera selon l'usage qu'on veut en faire. Celui-ci sera surtout un outil pédagogique ou encore un outil pour de nouveaux utilisateurs des systèmes de recommandation. Pour plus de détails sur l'outil PREA voir (Lee *et al.*, 2012a) et (Lee *et al.*, 2012b)

On utilise cet outil par le biais des logiciels Matlab et Eclipse. Il faut y inclure notre base de données contenant des notes de 1 à 5, et 0 pour des données inconnues. On sélectionne ensuite les algorithmes à comparer et l'outil rend plusieurs statistiques. On peut également sélectionner plusieurs types d'échantillonnages.

Comme à la base, il ne rend pas de données brutes mais simplement des statistiques générales, nous avons modifié PREA pour pouvoir effectuer nos simulations.

Plusieurs modifications ont été faites au code Java et Matlab pour extraire les données désirées ainsi qu'au niveau des algorithmes choisis. Une bonne connaissance de ces langages informatiques a été nécessaire.

9.2 MovieLens

MovieLens est un site internet de recommandation de films. Il est géré par l'équipe de recherche GroupLens Research basée à l'Université du Minnesota. Il utilise principalement des outils de filtrage collaboratif sur les données obtenues du site Movie Data Base (www.themoviedb.org)(www.grouplens.org/dataset/movielens).

Nous avons effectué des simulations sur la base de données **movielens100k**. Celle-ci contient 1682 films et 943 usagers, parmi ceux-ci nous retrouvons 100000 notations de films. (Harper et Konstan, 2015)

9.3 Méthodologie

Nous avons sélectionné une partie de la base de données qui contenait un grand nombre de données. Pour ce faire, nous avons sélectionné les 100 usagers et les 35 films avec le plus grand nombre de notations. Sur les 3500 notes possibles, la sous-base en contenait 2683.

Pour faire les simulations, nous avons voulu voir l'effet de la densité de la matrice, soit avec 5%, 10%, 15%, 20%, 25%, 50% et 75% des données disponibles pour chaque usager.

Par exemple, si un usager avait noté 20 films sur les 35, alors pour le même usager nous avons comparé les algorithmes avec 1, 2, 3, 4, 5, 10, et 15 notes.

Pour la comparaison qui nous intéresse dans ce mémoire, nous voulions nous concentrer sur les permutations finales. C'est-à-dire, l'ordre dans lequel les films

sont suggérés à l'utilisateur potentiel.

Nous avons choisi de comparer les algorithmes avec la distance de Kendall, qui tient compte de l'ordre dans lequel les films sont présentés.

9.3.1 Statistique de comparaison

La **distance de Kendall** se définit comme suit (Diaconis, 1988)

$$d_\tau(s, s') = \frac{2}{n(n-1)} \sum_{1 \leq i < j \leq n} \mathbb{1}(s' \circ s^{-1}(i) > s' \circ s^{-1}(j)), \qquad s, s' \in \mathcal{S}_n \qquad (9.1)$$

Pour deux permutations parfaitement égales, on attribuera la distance de 0 et pour deux permutations totalement inverses, on attribue la distance de 1.

Pour simplifier, prenons un exemple avec $n = 4$ et pour les deux vecteurs $A, B \in \mathcal{S}_4$

$A = (1, 2, 3, 4)$ et $B = (3, 1, 4, 2)$

Pour $A_i > A_j$, on comptera 1 si $B_i < B_j$, 0 si $B_i > B_j$ et pour $A_i < A_j$, on comptera 1 si $B_i > B_j$, 0 si $B_i < B_j$.

Dans l'exemple, on compte trois différences, pour les couples (i, j) suivant $(1, 2)$, $(1, 4)$, $(3, 4)$

$$d_\tau(A, B) = \frac{2}{4(4-1)} 3 = 0.3$$

La relation entre cette distance et le tau de Kendall τ est

$$0 \leq d_\tau(s, s') = (1 - \tau(s, s')/2 \leq 1$$

9.3.2 Détails pour BBR

La loi a priori du paramètre θ de la copule gaussienne (à la section 8.2) a été spécifiée comme une loi Bêta avec des paramètres $\alpha = 6$ et $\beta = 2$. Pour la trouver, nous avons utilisé la relation $\theta = sin(\tau\pi/2)$ où τ est le tau de Kendall. À l'aide du logiciel R, nous avons alors calculé la distribution empirique du τ entre les usagers et la moyenne générale sur l'ensemble de la base de données MovieLens qu'on a estimé avec une loi Bêta.

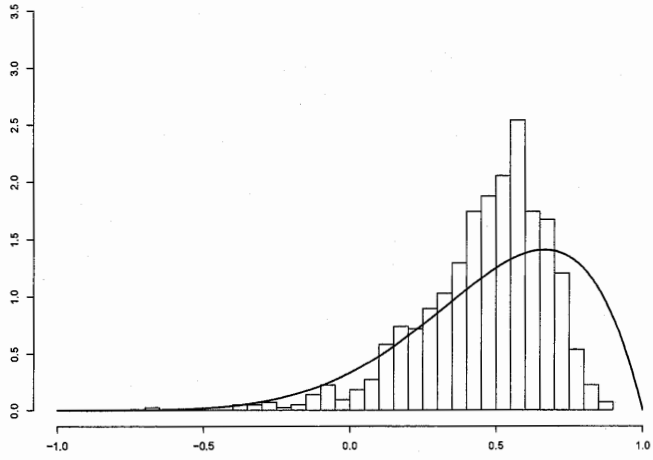


Figure 9.1 Histogramme de la distribution empirique de τ et densité de la variable aléatoire $\theta = 2T - 1$, avec T une loi Bêta(6,2)

Une interprétation est que la plupart des usagers notent les films de manière similaire. Un film populaire récolte des notes élevées chez la majorité des usagers. On utilise cette intuition dans le choix de la loi a priori pour θ dans la relation entre l'expert et l'usager.

De plus, nous avons utilisé les rangs donnés par Slope One à titre d'expert dans nos simulations.

9.3.3 Gestion des égalités

Comme les notes données aux films sont non-continues, soit de 1 à 5, nous nous retrouvons avec des égalités dans nos données. Les égalités furent traitées de deux façons différentes.

La première manière est une **gestion selon le rang**. Puisque les films avaient déjà été positionnés dans l'ordre de la moyenne générale, les égalités ont été gérées de la même manière.

Si deux films ont une note de 5, alors le film avec la meilleure moyenne générale se verra attribuer une note de 5.0002 tandis que le deuxième film se verra attribuer une note de 5.0001 .

$$M = \begin{bmatrix} 5 \\ 5 \\ 2 \\ 5 \\ 4 \\ 3 \\ 4 \end{bmatrix} \rightarrow \begin{bmatrix} 5.0003 \\ 5.0002 \\ 2 \\ 5.0001 \\ 4.0002 \\ 3 \\ 4.0001 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 2 \\ 7 \\ 3 \\ 4 \\ 6 \\ 5 \end{bmatrix}$$

Figure 9.2 Gestion des égalités selon le rang

Une méthode de gestion des égalités qui nous semblait plus juste est la **gestion aléatoire**.

On attribue une permutation à notre sous-échantillon de 5. On pourrait obtenir $(1, 2, 3)$ ou bien $(2, 3, 1)$, etc.

La figure 9.3 pourrait être une solution.

$$M = \begin{bmatrix} 5 \\ 5 \\ 2 \\ 5 \\ 4 \\ 3 \\ 4 \end{bmatrix} \rightarrow \begin{bmatrix} 5.0001 \\ 5.0003 \\ 2 \\ 5.0002 \\ 4.0001 \\ 3 \\ 4.0002 \end{bmatrix} \rightarrow \begin{bmatrix} 3 \\ 1 \\ 7 \\ 2 \\ 5 \\ 6 \\ 4 \end{bmatrix}$$

Figure 9.3 Gestion des égalités de manière aléatoire

De cette manière, il n'y a plus aucun doute à savoir si une technique était avantagée par l'ordre de la moyenne générale.

9.3.4 Techniques comparées

Nous avons comparé la technique BBR avec plusieurs techniques disponibles sur le logiciel PREA (Lee *et al.*, 2012b). Les techniques choisies sont, tel que vu dans les chapitres précédents, SlopeOne, RegSVD, NPCA, NMF, PMF et BPMF. Ces techniques sont parmi les plus connues et les plus simples d'usage. La figure 9.4 nous rappelle quelques liens entre les différents algorithmes.

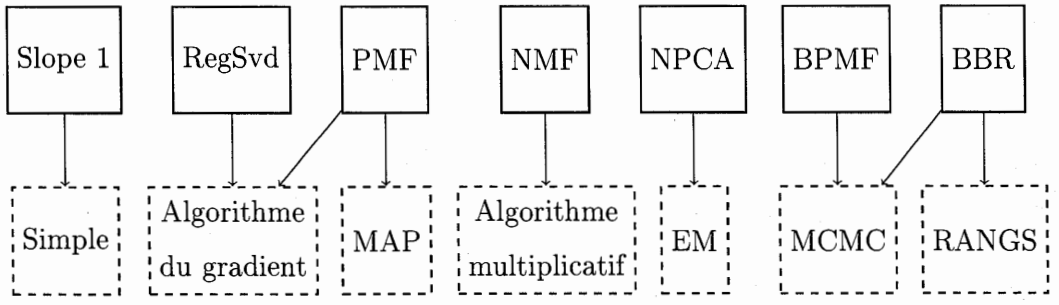


Figure 9.4 Organigramme des algorithmes

Contrairement à la technique BBR qui utilise les données d'un usager et d'un

expert et qui renvoie les rangs finaux, ces techniques utilisent les données de tous les usagers à la fois et renvoient les notes estimées.

Nous avons donc calculé les rangs à partir des notes estimées par ces techniques. Pour ce faire, nous avons utilisé la formule du calcul des rangs tel que décrit à la page 61 et la gestion des égalités de manière aléatoire.

9.4 Résultats

Pour une proportion p des données disponibles fixées dans la matrice, et pour chaque usager u, s_u sera le vecteur des vrais rangs et $\hat{s}_{u,k,p}$ sera le vecteur des rangs prédits basés sur le pourcentage p des données gardées, et ce pour les répétitions $k = 1, \ldots, 30$.

Par exemple, si nous voulons conserver 50% des données pour un usager qui a noté 20 films, à chaque répétition on choisit 10 films aléatoirement et on attribue un zéro aux 10 autres.

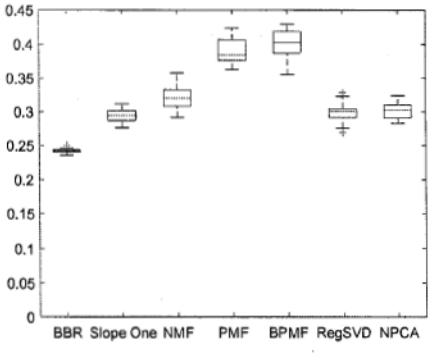
La figure 9.5 montre les distributions échantillonnales des répétitions k de la statistique

$$d(p,k) = \frac{1}{100} \sum_{u=1}^{100} d_\tau(\hat{s}_{u,k,p}, s_u). \tag{9.2}$$

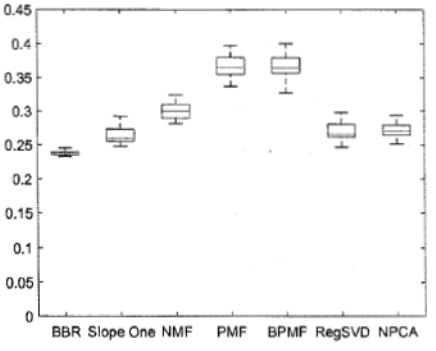
Pour avoir une idée globale de la performance de chaque algorithme, la figure 9.6 ci-dessous montre les valeurs de la statistique moyenne

$$d(p) = \frac{1}{30} \sum_{k=1}^{30} d(p,k), \tag{9.3}$$

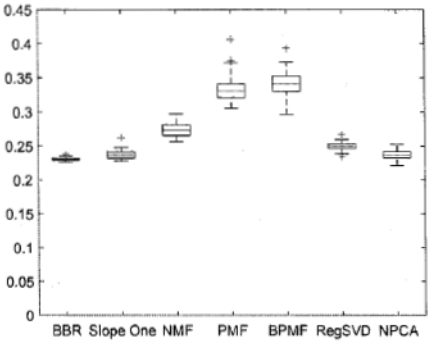
pour les pourcentages p se situant dans les valeurs données plus hauts.



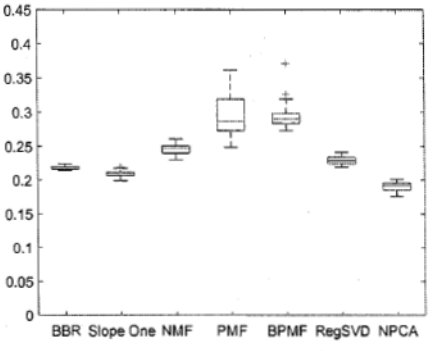
(a) $p = 0.05$



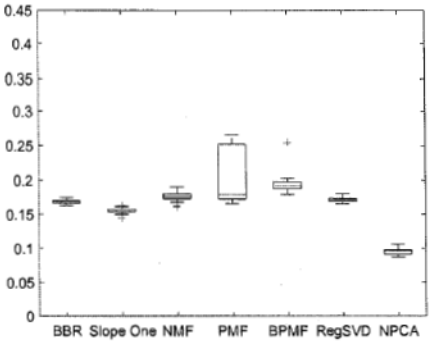
(b) $p = 0.1$



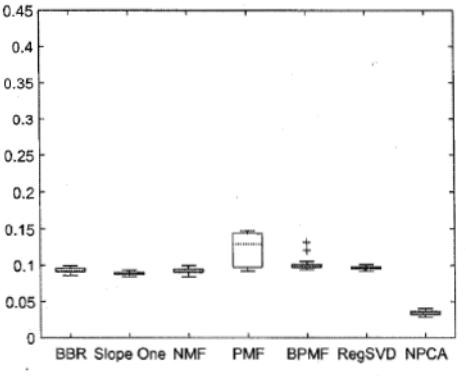
(c) $p = 0.15$



(d) $p = 0.25$



(e) $p = 0.5$



(f) $p = 0.75$

Figure 9.5 Distribution de $d(p, k)$ pour différentes valeurs de p.

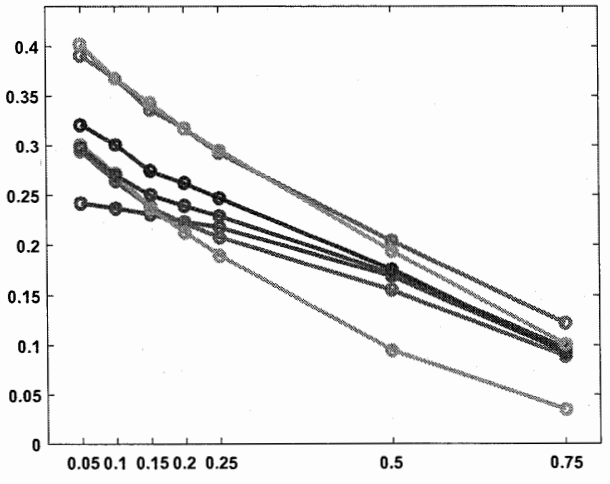


Figure 9.6 Valeurs de $d(p)$ pour différentes valeurs de p.

À la figure 9.6, la méthode BBR est en bleu, NPCA en jaune, Slope One en orange, RegSVD en mauve, NMF en noir, PMF en gris foncé, et BPMF en gris pâle.

On peut voir que la méthode BBR se démarque et semble plus performante que les autres pour des bases contenant très peu de données. Cette force pourrait être expliquée par la loi a priori choisie ainsi que par le fait qu'elle utilise l'intuition d'un expert, en l'occurrence Slope One. On remarque que pour de petits pourcentages, BBR est plus performante que Slope One. Dans les autres cas, elle reste à un niveau similaire.

Lorsqu'on augmente le pourcentage de données disponibles, la plupart des autres techniques semblent converger. Seul la technique NPCA apparait supérieure aux autres lorsque les pourcentages de données disponibles dépassent 25%.

On peut également noter sur la figure 9.5 que la technique BBR montre une variance inférieure aux autres, donc plus stable.

CONCLUSION

Le but premier de ce mémoire était de comparer la nouvelle technique d'inférence bayésienne basée sur les rangs bivariés à plusieurs algorithmes reconnus de filtrage collaboratif. Nous avons décrit des outils de filtrage collaboratif de plusieurs types, une méthode non-paramétrique, deux méthodes déterministes, et plusieurs méthodes stochastiques.

Nous avons constaté que plusieurs de ces algorithmes utilisaient des outils statistiques similaires sans pour autant arriver aux mêmes résultats. Notamment avec RegSVD et PMF qui utilisent un algorithme du gradient équivalent, à NPCA et PMF qui partent d'un modèle avec erreurs Gaussiennes ou encore à BPMF et BBR qui utilisent une méthode de Monte-Carlo par chaînes de Markov.

La technique d'inférence Bayésienne basée sur rangs bivariés se démarque et elle est assez compétitive si on se compare aux techniques du marché qui utilisent les données de tous les usagers. Elle semble également plus stable et c'est la seule qui utilise l'information d'un expert, en l'occurrence Slope One.

Nous pouvons croire que son utilisation est très intéressante particulièrement lorsqu'il n'y a que très peu d'information sur un client ou encore dans les débuts d'implémentation d'un système de recommandation. Avec une légère modification dans l'algorithme, elle pourrait potentiellement être performante pour prédire un top-n films par usagers.

RÉFÉRENCES

Benneth, J. et Lanning, S. (2007). The netflix prize. *Proceedings of KDD Cup and Workshop*.

Breese, J. S., Heckerman, D. et Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*.

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing*, *1*(2), 67–69.

Casella, G. et Berger, R. L. (2002). *Statistical inference*, volume 2. Duxbury Pacific Grove, CA.

Chin, W.-S., Zhuang, Y., Juan, Y.-C. et Lin, C.-J. (2015). A learning-rate schedule for stochastic gradient methods to matrix factorization. Dans *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 442–455. Springer.

Cremonesi, P., Koren, Y. et Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. Dans *Proceedings of the fourth ACM conference on Recommender systems*, 39–46. ACM.

Dempster, A. P., Laird, N. M. et Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, 1–38.

Diaconis, P. (1988). *Group representation in probability and statistics*. Institue of Mathematical Statistics Lecture Notes-Monograph Series.

Friedman, J., Hastie, T. et Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

Guillotte, S., Perron, F. et Segers, J. (2018). Bayesian inference for bivariate ranks. *ArXiv e-prints*.

Hansen, P. C. (1987). The truncatedsvd as a method for regularization. *BIT Numerical Mathematics*, *27*(4), 534–553.

Harper, F. M. et Konstan, J. A. (2015). The movielens datasets : History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *5*, 1–19.

Jiang, T. Q. et Lu, W. (2013). Improved slope one algorithm based on time weight. Dans *Applied Mechanics and Materials*, volume 347, 2365–2368. Trans Tech Publ.

Lee, D. D. et Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Letters to Nature*, *401*.

Lee, D. D. et Seung, H. S. (2001). Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*.

Lee, J., Sun, M. et Lebanon, G. (2012a). A comparative study of collaborative filtering algorithms. *ArXiv Report*.

Lee, J., Sun, M. et Lebanon, G. (2012b). Prea : Personalized recommendation algorithms toolkit. *Journal of Machine Learning Research*.

Lehmann, E. L. et Casella, G. (2006). *Theory of point estimation*. Springer Science & Business Media.

Linden, G., Smith, B. et York, J. (2003). Amazon.com recommendations : Item-to-item collaborative filtering. *IEEE Internet computing*, *7*(1), 76–80.

Murphy, K. P. (2007). Conjugate bayesian analysis of the gaussian distribution. *def*, *1*($2\sigma 2$), 16.

Nelsen, R. B. (1999). *An introduction to copulas*. Springer Science & Business Media.

Nguyen, H. V. et Bai, L. (2010). Cosine similarity metric learning for face verification. Dans *Asian conference on computer vision*, 709–720. Springer.

Paterek, A. (2007). Improving regularized singular value decomposition collaborative filtering. *Proceedings of KDD Cup and Workshop*.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *6*(2), 559–572.

Rawlings, J. O., Pantula, S. G. et Dickey, D. A. (2001). *Applied regression analysis : a research tool*. Springer Science & Business Media.

Rencher, A. C. (2003). *Methods of multivariate analysis*, volume 492. John Wiley & Sons.

Resnick, P. et Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, *40*(3), 56–58.

Robert, C. et Casella, G. (1999). *Monte Carlo statistical methods*. Springer New York.

Salakhutdinov, R. et Mnih, A. (2008a). Bayesian probabilistic matrix factorization using markov chain monte carlo. *Proceedings of the 25th International Conference on Machine Learning*.

Salakhutdinov, R. et Mnih, A. (2008b). Probabilistic matrix factorization. *Advances in Neural Information Processing Systems 20, Cambridge, MA : MIT Press*.

Saygin, A. P., Cicekli, I. et Akman, V. (2000). Turing test : 50 years later. *Minds and Machines*, *10*(4), 463–518.

Schein, A. I., Popescul, A., Ungar, L. H. et Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. Dans *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 253–260. ACM.

Singhal, A. (2001). Modern information retrieval : A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, *24*(4).

Su, X. et Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence Volume*.

Tipping, M. E. et Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, *3*, 611–622.

Töscher, A., Jahrer, M. et Bell, R. M. (2009). The bigchaos solution to the netflix grand prize. *Netflix prize documentation*, 1–52.

Wang, P. et Ye, H. (2009). A personalized recommendation algorithm combining slope one scheme and user based collaborative filtering. Dans *Industrial and Information Systems, 2009. IIS'09. International Conference on*, 152–154.

Webb, B. et Gorrell, G. (2006). Generalized hebbian algorithm for incremental latent semantic analysis. *Proceedings of Interspeech*.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A : mathematical, physical and engineering sciences*, *366*(1881), 3717–3725.

Wishart, J. (1928). The generalised product moment distribution in samples from a normal multivariate population. *Biometrika*, 32–52.

Wold, S., Esbensen, K. et Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, *2*(1-3), 37–52.

You, H., Li, H., Wang, Y. et Zhao, Q. (2015). An improved collaborative filtering recommendation algorithm combining item clustering and slope one scheme. Dans *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, 313–316.

Yu, K., Zhu, S., Lafferty, J. et Gong, Y. (2009). Fast nonparametric matrix factorization for large-scale collaborative filtering. *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval.*

Yuan, Y.-x. (2008). Step-sizes for the gradient method. *AMS IP Studies in Advanced Mathematics*, *42*(2), 785.