



Factorisation matricielle, application à la recommandation personnalisée de préférences

Julien Delporte

► To cite this version:

Julien Delporte. Factorisation matricielle, application à la recommandation personnalisée de préférences. Informatique [cs]. INSA de Rouen, 2014. Français. NNT: 2014ISAM0002. tel-01005223

HAL Id: tel-01005223

<https://theses.hal.science/tel-01005223>

Submitted on 12 Jun 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

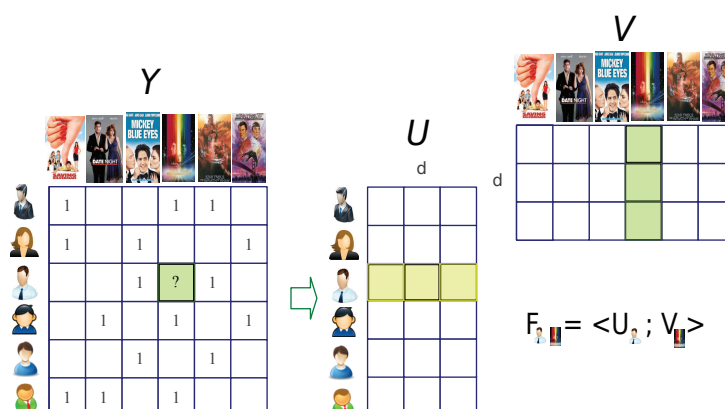
Présentée à :
L'Institut National des Sciences Appliquées de Rouen

En vue de l'obtention du titre de :
Docteur en Informatique

Par :
Julien DELPORTE

Intitulée :

Factorisation Matricielle, Application à la Recommandation Personnalisée de Préférences



soutenue le 3 février 2013

Devant le jury composé de :

Rapporteurs : Younès BENNANI
Gérard GOVAERT

Examineurs : Anne BOYER
Michèle SEBAG
Dominique FOURDRINIER

Directeur : Stéphane CANU

- Université Paris-Nord
- Université de Technologie de Compiègne
- Université de Lorraine
- Université Paris-Sud
- Université de Rouen
- INSA de Rouen

Table des matières

Introduction	1
1 Présentation des systèmes de recommandation	5
1.1 Contexte	6
1.1.1 Une histoire de la recommandation	6
1.1.2 Le challenge Netflix	7
1.2 Les problématiques usuelles	7
1.3 Les différentes approches	8
1.3.1 Les approches classiques	8
1.3.2 Les approches contextuelles	9
1.4 Les obstacles rencontrés par les systèmes de recommandation	10
1.4.1 Les types de données : des volumes massifs	10
1.4.2 L'éthique et les systèmes de recommandation	11
1.5 L'avenir des systèmes de recommandation	12
2 Factorisation et Recommandation, un état de l'art	13
2.1 Formalisation du problème de Factorisation	14
2.1.1 Notation et Formulation générale	14
2.1.2 Les différentes approches	14
2.1.3 Les problèmes sous-jacents à ces approches	15
2.1.4 Le choix du critère d'attache aux données	16
2.1.5 Le terme régularisant	17
2.1.6 La nature des problèmes	18
2.1.7 La matrice de bruit	19
2.2 La factorisation dans la recommandation	19
2.2.1 Méthode de résolution exacte sur grandes matrices creuses	20
2.2.1.1 Principe général de la décomposition en valeurs singulières sur grandes matrices creuses	20
2.2.1.2 Bi-diagonalisation	21
2.2.1.3 Diagonalisation d'une matrice bi-diagonale	22
2.2.2 Descente de gradient stochastique (SGD)	22
2.2.3 Optimisation alternée	24
2.2.4 Algorithme Multiplicatif	24
2.2.5 Approches non linéaires	26
2.2.6 Factorisation Tensorielle	26
2.3 Les méthodes d'évaluation	29
2.3.1 Le choix des modèles	29
2.3.2 La méthode d'évaluation	29

2.3.3	La métrique d'évaluation	30
2.4	Conclusion	30
3	Tout est une histoire de contexte	33
3.1	Les types de données contextuelles	34
3.1.1	Les variables utilisateurs et produits	34
3.1.2	Les données temporelles	34
3.1.3	Les données sociales	36
3.1.4	Informations contextuelles implicites	36
3.2	Les différents paradigmes de contextualisation	36
3.2.1	La contextualisation par prétraitement	36
3.2.2	La contextualisation par post-traitement	37
3.2.3	La modélisation contextuelle	37
3.3	Un cas d'étude précis : la vente d'outils	38
3.3.1	Présentation du problème	38
3.3.2	Présentation des approches contextuelles	38
3.3.2.1	Un contexte temporel	39
3.3.2.2	Un contexte de chantier	39
3.3.3	Protocole expérimental	39
3.3.3.1	Méthodes de factorisation choisies	40
3.3.3.2	Évaluation	41
3.3.3.3	Protocole de contextualisation temporelle	41
3.3.3.4	Protocole de contextualisation par chantier préliminaire	41
3.3.3.5	Protocole de contextualisation par chantier évolué	42
3.4	Discussion	42
4	Recommandation dans les réseaux sociaux	45
4.1	Les Concepts	46
4.2	État de l'art sur la recommandation dans les réseaux sociaux	47
4.2.1	Notations	48
4.2.2	Ajout de contraintes au problème	48
4.2.3	Modification de la fonction de décision	50
4.3	D'une pierre deux coups : recommandation et apprentissage d'influence	50
4.3.1	L'influence des amis	50
4.3.2	Optimisation	51
4.3.2.1	Mise à jour de U	52
4.3.2.2	Mise à jour de V	53
4.3.2.3	Mise à jour de A	53
4.3.2.4	Algorithme	54
4.3.2.5	Prédiction	55
4.3.3	Expérimentations	55
4.3.3.1	Les Problèmes	56
4.3.3.2	Les données	56
4.3.3.3	Protocole d'évaluation	57
4.3.3.4	Méthodes en comparaison	57
4.3.3.5	Résultats d'expériences	58

4.4	Discussion	63
5	Sélection de modèle pour la factorisation	65
5.1	La sélection de modèle	66
5.1.1	Un cadre de sélection de modèle	66
5.1.2	Estimation du risque	66
5.1.3	Choix du critère, de la pénalité et du coût	67
5.2	Sélection de modèle dans la factorisation	67
5.2.1	État de l'art	68
5.2.2	Formalisation du problème	68
5.2.3	Pénalité et estimateur du risque	69
5.2.4	Estimateur sans biais du risque	70
5.2.5	Calcul de la divergence	70
5.2.6	Limites du modèle actuel	71
5.3	Adaptation et élargissement de la méthode	71
5.3.1	Amélioration relative à la fonction de seuillage	71
5.3.1.1	Fonctions de seuillage proposées	71
5.3.1.2	Protocole expérimental	72
5.3.1.3	Quelques résultats expérimentaux	73
5.3.2	Passage à l'échelle	76
5.3.2.1	Approximation de la divergence	76
5.3.2.2	Protocole expérimental	78
5.3.2.3	Résultats expérimentaux	79
5.3.3	Estimation de la variance	81
5.3.3.1	Protocole expérimental	81
5.3.3.2	Quelques résultats	82
5.4	Discussion	84
	Conclusion	85
	A Article sur la recommandation contextuelle	89
	B Annexes diverses	99
B.1	Calcul de la divergence	100
B.2	Introduction à la notion de différentielle pour les matrices	106
B.3	Calcul de la fonction de répartition de la loi de Marchenko-Pastur	107
	Références	109

Introduction

*Chaque fois que la science avance d'un pas, c'est
qu'un imbécile la pousse, sans le faire exprès.*

– Émile Zola

Les travaux présentés dans ce manuscrit s'articulent autour de la problématique d'optimisation à grande échelle, et plus particulièrement autour des méthodes de factorisation matricielle pour des problèmes de grandes tailles. L'objectif principal des méthodes de factorisation de grandes matrices est d'extraire des variables latentes qui permettent d'expliquer les données dans un espace de dimension inférieure. Nous nous sommes intéressés au domaine d'application de la recommandation et plus particulièrement au problème de prédiction de préférences d'utilisateurs.

Les systèmes de recommandation en un mot

Un système de recommandation est un outil logiciel qui vise à prédire l'intérêt qu'un utilisateur pourrait avoir pour tel ou tel article, ce à partir d'un certain nombre d'informations, et à faire état de ce potentiel intérêt à l'utilisateur d'une manière ou d'une autre. Un exemple simple et pertinent de système de recommandation est celui mis en place par le site de vente en ligne *Amazon*¹. Celui-ci suggère à ses clients, lorsqu'ils ont effectué un achat ou ont mis des articles dans leur panier, un certain nombre de produits susceptibles de les intéresser. On peut encore citer l'exemple du service de recommandation de publicités de *Google*, qui suggère à l'utilisateur des publicités ciblées en se basant sur un historique de recherche ou de navigation.

Le problème de prédiction de préférences peut être vu comme un problème d'optimisation. Soit M la fonction qui à un utilisateur i donné et à un article j donné associe l'intérêt réel que i a pour j . Cette fonction est inconnue mais nous supposons que les données observées Y dépendent de cette fonction M . Le problème de prédiction de préférence consiste à chercher une fonction $\hat{M}$ qui approxime M à partir des observations Y . Cette fonction $\hat{M}$ associe à un utilisateur i et à un article j un score d'appétence de i pour j , ce score ($\hat{M}_{ij}$) dénote le niveau d'intérêt prédit de l'utilisateur i pour l'article j . Ce qui correspond à un problème d'optimisation si l'on pose un modèle sur les données (par exemple $Y = M + R$, où R est un bruit quelconque) et si l'on se fixe une fonction objectif à optimiser (par exemple $\min \|Y - \hat{M}\|^2 + \lambda \|\hat{M}\|^2$).

Le chapitre 1 présente les systèmes de recommandation. Après un historique des moteurs de recommandation existants, nous explicitons les problématiques liées à ces systèmes ainsi que les obstacles et difficultés qui peuvent être rencontrés. Nous présentons également les différentes approches proposées quant à l'appréhension des systèmes de recommandation, en précisant la position des méthodes de factorisation dans ces approches.

1. www.amazon.fr

Factorisation matricielle

Un problème de factorisation matricielle est un problème d'optimisation où l'on suppose que le modèle M est le produit de matrices de facteurs latents, comme par exemple $M_{ij} = \langle U_i^*, V_j^* \rangle$, où U_i^* est le vecteur (de taille d) des variables latentes explicatives de l'utilisateur i et V_j^* est le vecteur des variables latentes explicatives de l'article j . Généralement, on considère que le nombre de variables latentes d est très inférieure à la taille des données. Le but de la factorisation matricielle est d'estimer ces matrices U^* et V^* afin d'obtenir une reconstruction qui jouera le rôle de fonction score $\hat{M}_{ij} = \langle U_i, V_j \rangle$. Le principal intérêt de la factorisation (particulièrement dans la recommandation) est de fournir une représentation de faible rang d'un modèle de grande dimension.

Le chapitre 2 dresse un état de l'art des méthodes de factorisation en général et lorsqu'elles sont appliquées au domaine de la recommandation. Nous présentons comment se formalise un problème de factorisation à partir d'un problème de recommandation donné, notamment comment bien choisir le critère d'attache aux données (qui pénalise un écart entre les données et l'estimation du modèle) ou le terme de régularisation (qui contrôle la complexité de l'estimation du modèle). Nous énumérons les méthodes de factorisation classiquement utilisées pour résoudre de tels problèmes, en détaillant les différents algorithmes, en précisant dans quel cas ils sont plus efficaces ou mieux adaptés pour résoudre le problème posé. Nous y présentons également les méthodes d'évaluation de modèles généralement utilisées dans le domaine de la recommandation.

Contributions

Les contributions de ce manuscrit s'orientent autour de deux principaux axes : la recommandation contextuelle et la sélection de modèle pour la factorisation.

Recommandation contextuelle

En recommandation, il peut être intéressant de placer les utilisateurs (ou les articles) dans un contexte précis afin d'en extraire plus d'informations et d'obtenir une meilleure prédiction des préférences. Nous nous sommes intéressés à cette contextualisation à travers un cas d'étude réel où, à travers un protocole expérimental, nous montrons l'intérêt d'une contextualisation ainsi que la difficulté à la choisir. En effet, suivant le problème posé, la qualité de prédiction du modèle pourra dépendre fortement du contexte dans lequel on choisit de se placer.

Nous nous sommes également intéressés à une contextualisation sociale d'un problème de recommandation à grande échelle dans le domaine des réseaux sociaux en ligne. Nous proposons une approche novatrice permettant à la fois de faire de la recommandation d'articles en tirant avantage du réseau social et de construire un graphe de confiance à partir du graphe social existant. Nous montrons expérimentalement que l'apprentissage de ce graphe de confiance améliore la qualité de la recommandation par rapport à des méthodes de l'état de l'art qui prennent le graphe d'amitié tel qu'il est.

Sélection de modèle

Nous nous intéressons au problème de sélection de modèle dans la factorisation où l'on cherche à déterminer de façon automatique le rang de la factorisation par estimation de risque. Quelques méthodes existent et nous cherchons à adapter l'une d'elle (proposée dans [Candès *et al.* 2012]) à

des problèmes de grande taille. Cette méthode vise à faire de la sélection de modèle pour des problèmes de reconstruction liés à la décomposition en valeurs singulières. Nous proposons plusieurs voies d'amélioration afin, à la fois d'améliorer l'estimation du risque tout en permettant le passage à l'échelle, ce qui dans le domaine de la recommandation est un besoin primordial.

Organisation des contributions

Les chapitres 3, 4 et 5 présentent nos différentes contributions dans le domaine de la recommandation et de la factorisation matricielle. Les chapitres 3 et 4 présentent des approches contextuelles pour des problèmes de recommandation donnés. Nous nous intéressons ici à améliorer les performances de recommandation en tirant avantage de données contextuelles. Le chapitre 5 présente une approche de sélection de modèle pour la factorisation, où nous cherchons à déterminer le nombre de variables latentes permettant d'expliquer au mieux les données.

Dans le chapitre 3, nous dressons un état de l'art de la contextualisation dans la recommandation. Nous y détaillons donc les différents types de données contextuelles dont on peut disposer et les différentes approches de contextualisation existantes. Nous proposons ensuite, autour d'un cas d'étude précis, une approche qui tire avantage de données temporelles et de données implicites extraites à partir d'une connaissance experte.

Dans le chapitre 4, nous présentons nos travaux de contextualisation dans un cadre de recommandation à travers un réseau social. Nous y présenterons les différents concepts sociologiques sur lesquels l'on s'appuie généralement, ainsi que l'état de l'art du domaine. Enfin nous présentons notre approche et validons son intérêt à travers un protocole expérimental.

Nous nous intéressons dans le chapitre 5 au problème de sélection de modèle dans la factorisation, ce qui consiste à sélectionner le bon nombre de facteurs latents pour la factorisation. Nous y présentons la sélection de modèle de façon globale, ainsi qu'un état de l'art de la sélection de modèle pour des problèmes de factorisation. Ensuite nous présentons l'approche choisie. Enfin nous présentons les voies d'amélioration proposées que nous validons expérimentalement.

Publications

Voici la liste des publications relatives aux travaux de recherche présentés dans ce manuscrit.

[Pradel *et al.* 2011] Bruno Pradel, Savaneary Sean, Julien Delporte, Sébastien Guérif, Céline Rouveirol, Nicolas Usunier, Françoise Fogelman-Soulié et Frédéric Dufau-Joel. *A case study in a recommender system based on purchase data*. In Proceedings of the 17th ACM international conference on Knowledge discovery and data mining (SIGKDD'11), pages 377–385. ACM, 2011.

[Delporte *et al.* 2013] Julien Delporte, Alexandros Karatzoglou, Tomasz Matuszczyk et Stéphane Canu. *Socially Enabled Preference Learning from Implicit Feedback Data*. In Proceedings of the 13th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD'13), pages 145–160. Springer, 2013.

[Delporte *et al.* 2014] Julien Delporte, Alexandros Karatzoglou et Stéphane Canu. *Apprentissage et factorisation pour la recommandation*. Revue des Nouvelles Technologies de l'Information (RNTI). vol. RNTI-A-6, 2014.

Présentation des systèmes de recommandation

La prédiction d'un événement a pour résultat de faire arriver ce qu'elle a prédit.

– Paul Watzlawick

Sommaire

1.1	Contexte	6
1.1.1	Une histoire de la recommandation	6
1.1.2	Le challenge Netflix	7
1.2	Les problématiques usuelles	7
1.3	Les différentes approches	8
1.3.1	Les approches classiques	8
1.3.2	Les approches contextuelles	9
1.4	Les obstacles rencontrés par les systèmes de recommandation	10
1.4.1	Les types de données : des volumes massifs	10
1.4.2	L'éthique et les systèmes de recommandation	11
1.5	L'avenir des systèmes de recommandation	12

Nous présenterons dans ce chapitre les systèmes de recommandation, en retraçant brièvement leur histoire, les problématiques qu'ils soulèvent, les approches proposées pour résoudre ces problématiques ainsi que les obstacles auxquels nous pouvons être confrontés lorsque que l'on travaille dans la recommandation.

1.1 Contexte

1.1.1 Une histoire de la recommandation

La problématique des systèmes de recommandation a émergé comme un domaine de recherche propre dans les années 90. L'année 1992 voit l'apparition du système de recommandation de documents Tapestry et la création du laboratoire de recherche GroupLens qui travaillent explicitement sur le problème de la recommandation automatique dans le cadre des forums de news de Usenet. Tapestry avait pour but de recommander à des groupes d'utilisateurs des documents sur les newsgroups susceptibles de les intéresser. L'approche choisie est de type « plus proche voisin » à partir de l'historique de l'utilisateur. On parle alors de filtrage collaboratif, comme une réponse au besoin d'outils pour le filtrage de l'information énoncé à la même époque. Ce filtrage collaboratif n'avait pas le sens qu'il a maintenant, il s'agissait en fait d'une action collaborative des utilisateurs qui recommandaient aux autres utilisateurs des documents en attribuant des notes à ces documents selon certains critères (nous définirons ce que nous appelons actuellement filtrage collaboratif dans le paragraphe 1.3). En 1995 apparaissent successivement Ringo, un système de recommandation de musique, basé sur les appréciations des utilisateurs et Bellcore un système de recommandation de vidéos. La même année, GroupLens crée la société Net Perceptions dont un des premiers clients a été Amazon. C'est en 1996 qu'eut lieu le premier workshop dédié aux systèmes de recommandation qui est ensuite devenue la conférence ACM RecSys¹, référence dans le domaine.

Aujourd'hui, il existe de nombreux logiciels de recommandation disponibles dont certains sous licence libre comme Myrix, Duine, Apache Mahout ou EasyRec, ou d'autres sous licence propriétaire comme les solutions proposées par Kxen². La plupart des sites de vente en ligne s'intéressent aux systèmes de recommandation et intègrent des moteurs de recommandation sur leur site afin de prédire les préférences de leurs clients. Les principaux objectifs de tels moteurs sont de simplifier la navigation du client, de le fidéliser en lui recommandant les produits qui seraient les plus susceptibles de lui convenir. Mais la vente en ligne n'est pas le seul secteur intéressé par les systèmes de recommandation. Certaines entreprises mettent en place dans leur magasin des systèmes de recommandation. C'est le cas notamment de « la Boîte à outils »³, qui était démonstrateur au sein du projet CADI⁴ soutenu par l'ANR (Agence Nationale de la Recherche).

Les réseaux sociaux s'intéressent également aux systèmes de recommandation et mettent en œuvre des moteurs permettant entre autres de recommander des amis ou des centres d'intérêts (lieux, musiques, films...) mais permettant également d'afficher des publicités personnalisées. Le but ultime d'un système de recommandation est de jouer le rôle de l'expert compétent et de l'ami bienveillant à qui l'on peut faire confiance. Ce second rôle est d'ailleurs devenu une réelle difficulté, tant les systèmes de recommandation de plus ou moins bonne qualité se sont multipliés, poussant plutôt l'utilisateur à résister aux suggestions et même à les ignorer.

1. recsys.acm.org

2. myrrix.com, sourceforge.net/projects/duine/, mahout.apache.org/, easyrec.org/, www.kxen.com/

3. www.la-bao.fr/

4. www.agence-nationale-recherche.fr/projet-anr/?tx_lwmsuivibilan_pi2%5BCODE%5D=

1.1.2 Le challenge Netflix

Devant la multiplicité des approches proposées, la société Netflix a organisé en 2007 une compétition de recommandation mettant en jeu un million de dollars⁵. Autour de cet événement se sont cristallisées les principales avancées de la recherche de ces dernières années dans le monde de la recommandation. Netflix est une entreprise américaine qui offre un service de location de DVD en ligne. Chaque client peut, après avoir visionné un film, donner son avis sur ce dernier. Ainsi, il peut attribuer une note comprise entre un et cinq au film. Netflix disposait, avant la compétition, d'un système de recommandation permettant de suggérer aux clients un certain nombre de films qu'ils seraient susceptibles d'aimer.

Jugeant qu'une bonne recommandation était un moyen efficace pour, à la fois, fidéliser leur clientèle et augmenter leur chiffre d'affaires, ils voulurent améliorer leur propre moteur de recommandation (cinematch). Pour ce faire, ils mirent à contribution les membres de la communauté scientifique qui voulaient s'y intéresser en lançant ce désormais célèbre défi. Les participants disposaient d'un certain nombre de données : notamment la matrice contenant l'ensemble des votes des clients (notons que cette matrice contenait presque 500000 utilisateurs et 17000 films, cependant seulement 1% des valeurs de la matrice étaient non nulles), un certain nombre de données contextuelles sur les utilisateurs et sur les films et des informations temporelles sur ces données. Pour chaque client, Netflix avait caché le dernier vote effectué. La règle était simple : il fallait prédire pour chacun des clients, quelle était la valeur de ce vote et améliorer les performances en termes de RMSE⁶ de 10% par rapport au moteur cinematch utilisé par Netflix.

Le défi a finalement été remporté, presque trois ans après son lancement, par l'équipe « BellKor's Pragmatic Chaos ». La solution proposée consiste en un mélange de plus de 100 modèles, tous relativement simples à mettre en œuvre. Ils sont décrits dans [Bell *et al.* 2009, Piote & Chabbert 2009, Töschner *et al.* 2009]. Malheureusement cet agrégat de modèles s'est avéré beaucoup trop coûteux en temps et en énergie pour pouvoir être mis en production. Ce challenge a néanmoins permis de mettre en évidence l'intérêt des méthodes de factorisation pour la résolution de problèmes de recommandation, notamment grâce à la possibilité d'intégrer des informations complémentaires telles que des évaluations implicites, des effets temporels et des niveaux de confiance (Bell & Koren 2007).

1.2 Les problématiques usuelles

Le problème de recommandation consiste, pour un utilisateur donné, à prédire son score d'aptitude (ou d'utilité) pour une liste d'articles, dans le but de les ordonner (*ranking*). Ces scores sont personnalisés et chaque utilisateur dispose de ses propres recommandations. D'autres problèmes consistent à effectuer des recommandations plus complexes liées à un contexte (recommander à court terme/long terme, recommander une suite ou un groupe d'articles). On peut également chercher à identifier des comportements communs d'utilisateurs et d'articles, c'est le problème de classification croisée (*co-clustering*).

En considérant la nature des données disponibles, on peut distinguer deux principaux types de recommandations : la recommandation d'articles (au sens large⁷) où les données de base sont des

5. www.netflixprize.com

6. *Root Mean Square Error* : $RMSE(f, y) = \sqrt{\frac{\sum_{i=1}^n (y_i - f_i)^2}{n}}$.

7. Ces articles peuvent être de natures variées : produits de consommation, site web (pour un moteur de recherche), catégorie (pour du référencement d'articles sur un wiki), mot-clé pour l'annotation de produit...

couples article/utilisateur, et la recommandation d'amis à partir d'un graphe (social) pratiquée principalement par les réseaux sociaux (cet aspect est notamment présenté par [Aggarwal 2011]) où l'on cherche à recommander à un utilisateur un autre utilisateur. L'archétype du problème de recommandation peut se formaliser comme un problème de complétion de matrice. Un certain nombre de spectateurs ont noté des films et l'on souhaite prédire la note qu'ils donneraient aux films qu'ils n'ont pas encore vus. Il s'agit de valeurs manquantes dans une matrice spectateur/film qu'il faut reconstruire. Dans tous les cas, on cherche à identifier des variables cachées permettant de déterminer un profil (utilisateur, article ou ami) ou de prédire un comportement de groupe. On appelle ces variables cachées des variables latentes.

La plupart des systèmes de recommandation ont pour but principal d'augmenter la quantité de ventes, la quantité de fois qu'une information est lue, ou encore le nombre de clic sur une publicité. Mais ce n'est pas le seul objectif. Les systèmes de recommandation peuvent viser à diversifier les suggestions (de façon pertinente). Les problèmes de type longue traîne (*long tail*) s'intéressent à ce genre de recommandation en cherchant à améliorer la qualité de la prédiction en suggérant des articles n'intéressant que peu d'utilisateurs mais toutefois pertinent pour l'utilisateur ciblé. Un système de recommandation peut également s'intéresser à l'utilisateur lui-même et plus particulièrement à sa satisfaction et à sa fidélité. En effet, un système de recommandation satisfaisant ses utilisateurs sera plus à même d'être utilisé et un utilisateur fidélisé reviendra plus régulièrement solliciter le système de recommandation. Enfin, un système de recommandation peut être utilisé afin de mieux comprendre le comportement des utilisateurs et leurs goûts, afin par exemple d'améliorer la gestion du stock.

1.3 Les différentes approches

Le problème de recommandation peut se formaliser comme l'estimation d'une fonction d'utilité (score d'appétence) d'un article pour un utilisateur. Cette fonction permet la reconstruction de la matrice utilisateur/article et sa complétion. Autour de ce problème s'est développée une certaine terminologie visant à distinguer différents types d'approches (voir [Ricci *et al.* 2011] pour plus de détails). On peut distinguer trois grandes classes d'approches selon la nature des données prises en compte.

1.3.1 Les approches classiques

Il existe trois grandes classes d'approches dites classiques.

La première regroupe les approches orientées contenu (*content-based*) où l'on va chercher à recommander à un utilisateur donné des articles similaires à ceux précédemment appréciés par cet utilisateur. Un système purement orienté contenu va baser sa recommandation sur un modèle construit uniquement à partir de l'analyse du contenu des articles appréciés par cet utilisateur. Ce type d'approche est intéressant d'un point de vue computationnel et cette vision plutôt simpliste pouvait dans un contexte industriel où la rapidité de calcul sur des données de dimensions gigantesques prévaut aux performances du modèle. Cette rapidité de calcul a un prix et un certain nombre d'informations disponibles ne sont pas mises à profit pour effectuer la recommandation (notamment le comportement des autres utilisateurs). Cette classe regroupe des approches communes avec la recherche d'information (*information retrieval*) comme la classification (*clustering*), les arbres de décision ou encore les réseaux de neurones.

La deuxième classe d'approches regroupe celles de type filtrage collaboratif. Le principe de ces approches est de recommander à un utilisateur des articles que d'autres utilisateurs, similaires à cet utilisateur, ont appréciés. Un système purement orienté filtrage collaboratif ne prendra pas en compte les informations sur les articles, il ne connaîtra que les identifiants des articles et travaillera exclusivement sur la matrice utilisateur/article. Cette classe d'approches regroupe notamment les méthodes dites orientées utilisateur (*user-based*) ou orientées produit (*item-based*) dans lesquelles figurent notamment les modèles élaborant la construction de matrices d'association entre utilisateurs ou entre articles. De nombreux travaux se sont intéressés à ces approches, comme par exemple [Sawar *et al.* 2001, Wang *et al.* 2006, Castagnos & Boyer 2006]. Ces matrices d'association (ou de similarité) peuvent néanmoins s'avérer difficile à calculer pour des raisons de complexités temporelles et spatiales. C'est une approche très intuitive mais qui ne prend pas en compte la problématique liée à la taille des données. D'autres méthodes, qui souffrent beaucoup moins de ce problème computationnel mais gardent une puissance d'expression intéressante, font partie des approches de type filtrage collaboratif. Ces approches sont les méthodes dites orientées modèle (*model-based*). On peut citer parmi ces approches les méthodes de factorisation qui cherchent à extraire des variables latentes à la fois pour les utilisateurs et pour les articles.

Enfin, la dernière classe d'approches est en fait une combinaison des deux types d'approches précédemment cités. Cette approche hybride a pour but de tirer avantage, à la fois des informations de préférences de l'ensemble des utilisateurs (comme en filtrage collaboratif) et d'autres informations qui ne seront pas issues de la matrice utilisateur/article et seront contextuelles. L'utilisation de données contextuelles définit un nouveau type d'approche qui est décrit dans le paragraphe suivant.

1.3.2 Les approches contextuelles

Il existe différents types d'approche suivant la nature des informations contextuelles dont nous disposons (en plus de la matrice utilisateur/article). On peut citer notamment, les approches dites *démographiques* qui basent la personnalisation de la recommandation sur le profil de l'utilisateur, comme son âge, son sexe, sa nationalité, sa localisation géographique (Zheng *et al.* 2011)...

Les approches orientées *communauté* utilisent les informations sociales de l'utilisateur pour effectuer la prédiction. Les goûts supposés d'un utilisateur seront directement influencés par ceux de ses amis. Nous reviendrons d'ailleurs en détail sur ces approches dans le chapitre 4.

Les approches que l'on peut appeler *temporelles* se basent sur la non constance des préférences d'un utilisateur dans le temps, ainsi la recommandation pourra être fortement dépendante du moment où est faite cette recommandation.

Ces informations contextuelles peuvent être utilisées à différents niveaux dans le processus de recommandation. Elles peuvent être utilisées en prétraitement afin de ne sélectionner que les informations qui nous intéressent pour construire le modèle. Elles peuvent être utilisées en post-traitement afin de ne sélectionner, parmi les articles recommandés, que ceux qui correspondent au contexte. Enfin, ces données contextuelles peuvent être utilisées directement au cours de la construction du modèle.

Le chapitre 3 de ce manuscrit se concentre sur ces aspects contextuels de la recommandation. Nous reviendrons dessus plus en détail dans cette partie.

1.4 Les obstacles rencontrés par les systèmes de recommandation

En plus de la nature particulière des données dans les systèmes de recommandation, d'autres difficultés se posent. C'est également le cas de l'évaluation. Il est en effet difficile d'évaluer la pertinence d'un système de recommandation. [Ricci *et al.* 2011, Pu *et al.* 2012, Pradel 2013] se sont intéressés à ce problème.

1.4.1 Les types de données : des volumes massifs

La nature même des données dont nous disposons peut soulever des difficultés pour la mise en place d'un système de recommandation. Commençons par présenter ces données avant de préciser les obstacles que l'on peut rencontrer.

Les données utilisées dans les systèmes de recommandation ont des caractéristiques particulières comme leur volume car la taille des matrices d'achats/préférences est souvent très grande. Les données de Netflix contenaient environ 500 000 utilisateurs et 17 000 articles et Amazon rapportait en 2003 devoir gérer 29 millions d'utilisateurs et un catalogue de plusieurs millions d'articles. Notons par ailleurs que la nature de ces données va jouer un rôle primordial dans la formalisation du problème et dans sa résolution. En effet, une matrice de transactions, contenant donc uniquement les valeurs 1 ou 0 (1 signifiant qu'une action a été effectuée par un utilisateur donné pour un article donné), ne sera pas traitée de la même façon qu'une matrice de vote (*rating*) qui, elle, ne contient typiquement que des valeurs comprises en 0 et 5, où une valeur entre 1 et 5 indique l'appréciation explicite d'un utilisateur pour un article, et où 0 indique l'absence d'appréciation. Ainsi dans le cas des matrices de vote, 0 signifie simplement que la valeur n'est pas renseignée alors que dans le cas des matrices de transactions, la signification exacte du 0 est inconnue. Cela peut signifier que l'utilisateur n'aime pas l'article, qu'il ignore l'existence de l'article ou qu'il en a connaissance mais n'a encore effectué aucune action (on pourra choisir dans certains cas de considérer ces 0 effectivement comme une appréciation négative de l'utilisateur avec une pénalisation moindre que les 1 dans la fonction coût, ou du moins un certain nombre de ces 0). On appelle généralement ces données de transaction, des données d'appréciations implicites (*implicit feedback*). Les sens très différents des 0 dans ces deux types de matrices imposent des modèles et des manipulations différentes. De plus, ces matrices sont creuses (souvent moins de 1% de valeurs non nulles).

Ces données ne sont pas nécessairement les seules dont nous disposons. Nous avons vu que des données contextuelles peuvent être disponibles. On peut citer entre autres des données sur les utilisateurs (âge, ville, profession...) ou des données sur les articles. Certaines données sont également datées (date d'achat, historique de navigation...) et nous pouvons disposer d'un graphe social. Le regroupement de l'ensemble de ces différents types de données est appelé *Data Block* et est illustré par la figure 1.1.

On y retrouve les trois caractéristiques des volumes massifs de données (*Big Data*), qui soulèvent des difficultés :

- leur volumétrie, qui impose des choix d'algorithmes adaptés sachant tirer avantage de la nature parcimonieuse des données.
- leur variété (structurée, non structurée, multifacettes...)
- leur vélocité puisque dans de nombreux systèmes de recommandation les données doivent être utilisées au fil de l'eau dans des délais fortement contraignants.

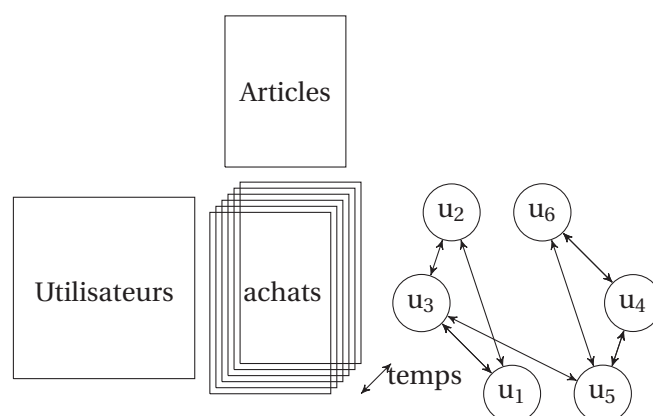


FIGURE 1.1 – Illustration d'un ensemble de données de type *Data Block*. Les matrices d'achat forment un tenseur dans le temps. Des données contextuelles sont disponibles sur les utilisateurs et sur les articles et forment des matrices supplémentaires. D'autres informations sont disponibles sous forme de graphe (réseaux sociaux).

1.4.2 L'éthique et les systèmes de recommandation

Il nous semble important d'évoquer les problèmes éthiques soulevés dans le monde de la recommandation. De nombreuses polémiques ont eu lieu quant à la façon dont étaient recueillies, stockées ou utilisées certaines données sur les utilisateurs ; en effet, la personnalisation des recommandations est d'autant plus précise qu'elle utilise des données pertinentes sur l'utilisateur, ses goûts et ses comportements. Or l'acquisition et le stockage de données personnelles peut s'avérer intrusive et constituer une violation de la vie privée, notamment lorsque les informations sont collectées de manière implicite à l'insu de l'utilisateur (sans parler des risques de divulgation inopportune de ce type d'information). De plus l'absence de transparence des entreprises quant à l'utilisation de ces données ou à la façon dont sont réalisées les recommandations n'est en rien rassurante pour les utilisateurs. On peut notamment citer Google qui est poursuivi en juin 2013 par la CNIL⁸ pour la façon jugée illégale de collecte d'informations des utilisateurs et pour l'absence de transparence de l'utilisation qui est faite de ces informations. Netflix a également fait l'objet d'une plainte en décembre 2009 pour avoir libéré les données du challenge bien qu'aucune information personnelle n'y figurait. De plus, la connaissance de l'utilisateur par le système de recommandation pourrait aller jusqu'à la personnalisation d'une véritable stratégie d'influence (Kaptein & Eckles 2010). Un système de recommandation se doit, en plus de respecter la législation, d'être transparent sur sa façon d'effectuer les recommandations pour éviter son rejet par les utilisateurs. Ainsi il devient fréquent de se voir justifier la suggestion d'un article par des phrases du type « les utilisateurs qui ont acheté ce produit, ont également acheté tel produit » rassurantes pour l'utilisateur, qui voit ainsi comment sont utilisés les informations dont disposent les entreprises.

Il reste cependant beaucoup à faire, principalement sur le manque de transparence au niveau de la collecte des données et sur leur utilisation.

8. Conseil National de l'Informatique et des Libertés

1.5 L'avenir des systèmes de recommandation

Bien que la recherche dans le domaine des systèmes de recommandation n'en est qu'à ses débuts, l'engouement des chercheurs au cours de ces vingt dernières années a fait grandement avancer les choses. Le livre *Recommender Systems Handbook*, qui présente un bilan exhaustif des systèmes de recommandation, montre déjà une certaine maturité du domaine de recherche.

Des workshops et des thèmes de recherche consacrés aux systèmes de recommandation ont fait leur apparition dans les principalement conférences internationales en apprentissage artificiel et en fouille de données ou dans des conférences spécialisées dans le domaine⁹. De nouvelles tendances de recherche apparaissent régulièrement, comme par exemple le *crowdsourcing*¹⁰ (externalisation ouverte) appliqué aux systèmes de recommandation qui fait l'objet d'un workshop dans la conférence ACM RecSys¹¹.

Le développement constant de sociétés sur Internet ayant de réels besoins en terme de recommandation assure un avenir florissant à ce domaine de recherche, où de nouvelles problématiques naissent régulièrement et où les contraintes techniques (temps de calcul, stockage des données...) nécessitent un réel apport de la communauté scientifique.

9. graphlab.org/lrsr2013/, sites.google.com/site/newsrec2013/, www.bars-workshop.org/, sydney.edu.au/engineering/it/~pizzato/workshop/resson2013/

10. fr.wikipedia.org/wiki/Crowdsourcing

11. recsys.acm.org/recsys13/crowdrec/

Factorisation et Recommandation, un état de l'art

What's past is prologue.

– William Shakespeare

Sommaire

2.1	Formalisation du problème de Factorisation	14
2.1.1	Notation et Formulation générale	14
2.1.2	Les différentes approches	14
2.1.3	Les problèmes sous-jacents à ces approches	15
2.1.4	Le choix du critère d'attache aux données	16
2.1.5	Le terme régularisant	17
2.1.6	La nature des problèmes	18
2.1.7	La matrice de bruit	19
2.2	La factorisation dans la recommandation	19
2.2.1	Méthode de résolution exacte sur grandes matrices creuses	20
2.2.2	Descente de gradient stochastique (SGD)	22
2.2.3	Optimisation alternée	24
2.2.4	Algorithme Multiplicatif	24
2.2.5	Approches non linéaires	26
2.2.6	Factorisation Tensorielle	26
2.3	Les méthodes d'évaluation	29
2.3.1	Le choix des modèles	29
2.3.2	La méthode d'évaluation	29
2.3.3	La métrique d'évaluation	30
2.4	Conclusion	30

Ce chapitre présente un état de l'art des méthodes de factorisation. Nous en ferons une présentation générale et nous attarderons sur un domaine d'application particulier qui est celui de la recommandation. Nous verrons comment sont construites les fonctions à optimiser suivant le problème posé ou les contraintes dans un cadre général, comment adapter ces problèmes dans le cadre de la recommandation. Nous présenterons également les algorithmes généralement utilisés, lesquels sont plus efficaces ou plus adaptés pour résoudre les problèmes auxquels nous nous intéressons. Enfin nous verrons quelles méthodes d'évaluation permettent d'avoir une validation correcte des modèles dans le domaine de la recommandation où cela peut être un véritable obstacle.

2.1 Formalisation du problème de Factorisation

2.1.1 Notation et Formulation générale

On suppose que l'on dispose d'une matrice de données $Y \in \mathbb{R}^{n \times p}$ concernant n utilisateurs et p articles¹. L'idée principale derrière un problème de factorisation est d'apprendre un modèle latent d'utilisateurs $U \in \mathbb{R}^{n \times d}$ et d'articles $V \in \mathbb{R}^{d \times p}$ de sorte que la reconstruction $\hat{M}_{ij} = U_i V_j$ entre un utilisateur i et un article j estime le terme Y_{ij} ou son utilité (score). Les facteurs latents U et V s'obtiennent généralement en minimisant une fonction objectif qui provient de fonctions coût régularisées (Srebro *et al.* 2005, Takacs *et al.* 2009, Weimer *et al.* 2008a) ou de modèles probabilistes (Salakhutdinov & Mnih 2008, Krohn-Grimberghe *et al.* 2012, Blei *et al.* 2003).

Dans les deux cas il s'agit d'un objectif multicritère puisque nous avons deux critères contradictoires à concilier, à savoir obtenir $\hat{M}$ proche de Y et garantir la faculté de généralisation du modèle sous-jacent en contrôlant la complexité de $\hat{M}$. Le problème peut être formalisé dans sa forme la plus simple par la minimisation du risque pénalisé :

$$\min_{\hat{M} \in \mathbb{R}^{n \times p}} \mathcal{L}(\hat{M}, Y) + \lambda \Omega(\hat{M}), \quad (2.1)$$

où $\mathcal{L}(\hat{M}, Y)$ est un terme d'attache aux données sur lequel nous reviendrons dans la partie 2.1.4, $\Omega(\hat{M})$ un terme régularisant (une pénalité) permettant d'éviter le sur-apprentissage sur lequel nous reviendrons en détail dans la partie 2.1.5, et λ un paramètre d'équilibrage dont le réglage établit le niveau de consensus entre les deux critères, nous reviendrons sur le choix de ce λ dans la partie 2.3 lorsque nous parlerons de sélection de modèle. Cette formulation est de type Lagrangienne et est également appelée régularisation de Tikhonov.

2.1.2 Les différentes approches

Il est possible d'appréhender la matrice Y de différentes façons. Généralement on voit la matrice Y comme la somme de deux matrices :

$$Y = M + R,$$

où R est une matrice de bruit sur laquelle nous reviendrons dans le paragraphe 2.1.7 et où M est la matrice contenant l'information que l'on va chercher à estimer par $\hat{M}$. On peut se placer dans différents paradigmes quant à cette matrice d'information.

Le premier consiste à supposer que des utilisateurs ou des articles ont des points communs. Ceci nous amène à penser que l'information contenue dans la matrice est redondante, et que donc cette matrice $\hat{M}$ est de faible rang. C'est la vision la plus souvent rencontrée dans la littérature.

Il est possible de trouver des variantes de ce paradigme en supposant qu'une partie de l'information est effectivement redondante mais qu'une autre partie (moins conséquente) ne l'est pas. Dans ce cas, certains auteurs (comme [Candès *et al.* 2011]) ont proposé de considérer le modèle suivant : $Y = L + S + R$, où L , une matrice de rang faible, et S , une matrice parcimonieuse², contiennent toutes les deux l'information.

1. Notons que l'on parle ici d'utilisateurs et d'articles pour les lignes et les colonnes de la matrice par anticipation pour la suite, mais ces lignes et colonnes pourraient représenter tout à fait autre chose, comme par exemple lors d'une analyse temps/fréquences d'un signal où les lignes peuvent représenter les fréquences, les colonnes le temps et où Y_{ij} désigne l'amplitude de la i -ème fréquence au j -ème instant

2. Ce que nous dénotons par parcimonie ici (et dans la suite manuscrit) est le fait que peu de valeurs de la matrice sont non nulles.

Dans un autre paradigme, on suppose que chaque utilisateur appartient à un groupe d'utilisateurs. On peut supposer la même chose des articles. On considère dans ce cas que l'information dans la matrice est regroupée sous forme de bloc et $\hat{M}$ est positive.

Suivant le paradigme dans lequel on se situe, les problèmes sous-jacents ne seront pas nécessairement les mêmes. Nous essaierons donc de déterminer dans le paragraphe suivant les différentes classes de problèmes de reconstruction suivant le paradigme choisi.

2.1.3 Les problèmes sous-jacents à ces approches

On peut identifier plusieurs types de problème de factorisation :

Problème d'approximation : le problème d'approximation consiste à chercher une matrice $\hat{M}$ de faible rang qui est la plus proche possible de la matrice Y (suivant un certain nombre de contraintes, entre autres des contraintes de rang). Dans ce cas, on cherche à minimiser la fonction objectif de l'équation 2.1 en utilisant une fonction $\mathcal{L}$ de la forme :

$$\mathcal{L}(\hat{M}, Y) = \sum_{i=1}^n \sum_{j=1}^p l(\hat{M}_{ij}, Y_{ij}), \quad (2.2)$$

où $l : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ est un coût quelconque (distance, divergence, opposé de la log vraisemblance...). Nous discutons du choix de cette fonction l dans le paragraphe 2.1.4.

Problème de complétion : un autre type de problème consiste à chercher les valeurs manquantes de la matrice Y , il s'agit d'un problème de complétion (Cravo 2009). On cherche ici aussi à estimer une matrice de faible rang. Dans ce cas, on connaît les valeurs Y_{ij} pour les couples (i, j) appartenant à un ensemble $\mathcal{Y}$, et les autres valeurs sont inconnues (et sont généralement représentées par la valeur 0 dans la matrice). Un moyen de formuler ce problème consiste à ne s'intéresser qu'aux termes connus de la matrice. Ainsi on cherche à minimiser la fonction objectif de l'équation 2.1 sur l'ensemble $\mathcal{Y}$:

$$\mathcal{L}(\hat{M}, Y) = \sum_{(i,j) \in \mathcal{Y}} l(\hat{M}_{ij}, Y_{ij}). \quad (2.3)$$

Il est possible de généraliser cette formulation en faisant apparaître des poids W qui joueront le rôle de filtre sur les données :

$$\mathcal{L}(\hat{M}, Y) = \sum_{i=1}^n \sum_{j=1}^p W_{ij} \times l(\hat{M}_{ij}, Y_{ij}), \quad (2.4)$$

en choisissant la matrice W par exemple de la forme suivante :

$$W_{ij} = \begin{cases} 0 & \text{si } (i, j) \notin \mathcal{Y}, \\ \phi(Y_{ij}) & \text{si } (i, j) \in \mathcal{Y}, \end{cases} \quad (2.5)$$

où $\phi : \mathbb{R} \rightarrow \mathbb{R}^+$ désigne une fonction positive quelconque. Ainsi ne seront prises en compte que les valeurs connues de la matrice Y , et celles-ci seront pondérées. C'est particulièrement intéressant lorsque l'on travaille sur une matrice de vote et que l'on souhaite donner plus de poids à une valeur qu'à une autre (par exemple donner plus d'importance à un 5 qu'à un 1). Dans le cas de matrices d'appréciation implicite, la formulation de l'équation 2.4 est conseillé et le choix de la pondération W sera déterminant (il est facile d'imaginer ne pas vouloir pénaliser autant les 0 dont on ne connaît pas la signification que les 1).

Problème de décomposition : Il existe également un type de problème que l'on peut appeler problème de décomposition. Ce type de problème est décrit notamment par [Candès et al. 2011] sous le

terme d'ACP robuste (Analyse en Composantes Principales). Dans ce cadre, la matrice de données Y est modélisée par :

$$Y = L + S + R,$$

avec $\hat{M} = L + S$ la somme d'une matrice de faible rang L et d'une matrice parcimonieuse S que l'on va chercher à déterminer en minimisant la matrice des résidus R . Ainsi on va imposer que L et S soient simultanément « régulières » (avec chacune leur terme régularisant propre).

$$\Omega(\hat{M}) = \lambda_L \Omega_L(L) + \lambda_S \Omega_S(S), \quad (2.6)$$

où généralement Ω_L est un critère permettant de contrôler le rang et Ω_S un critère induisant la parcimonie.

Problème de classification croisée : Le problème de classification croisée (ou *co-clustering*) ne vise pas à extraire des variables latentes permettant de déterminer des comportements individuels. Il s'agit ici d'identifier des comportements de groupes et d'associer les utilisateurs et les articles à ces groupes. Cette identification de groupes d'utilisateurs et d'articles se fait simultanément à la différence de la classification classique. La formulation du problème reste un problème de minimisation de la même forme que précédemment, la différence réside dans la nature de $\hat{M}$. Par exemple dans [Labiod & Nadif 2011b], le problème est formulé de telle sorte que $\hat{M}$ soit de la forme $\hat{M} = UCV$, où les vecteurs $C_{i\cdot}$ représentent les clusters d'utilisateurs, les vecteurs $C_{\cdot j}$ représentent les clusters d'articles, et les facteurs U et V sont des matrices binaires indiquant l'appartenance ou non à tel ou tel cluster.

Problème d'ordonnancement : Le problème d'ordonnancement (*ranking*) consiste, lors de la reconstruction de la matrice, à obtenir pour chaque utilisateur un ordre total sur les articles. La formulation du problème reste identique à celle d'un problème d'approximation ou de complétion, la différence réside dans le fait que les scores $\hat{M}_{i\cdot}$ sont utilisés afin de déterminer un ordre total entre les articles pour cet utilisateur i .

Ceux-ci sont les principaux problèmes qui peuvent être rencontrer, nous invitons le lecteur à consulter [Ricci et al. 2011] pour de plus amples informations.

2.1.4 Le choix du critère d'attache aux données

Le terme d'attache aux données traduit la confiance que l'on accorde à la prédiction $\hat{M}$ ayant observé Y . Notons que dans le cadre d'une modélisation probabiliste cette approche est dite de type maximum de vraisemblance et la fonction l est l'opposé de la log vraisemblance. Dans ce cas, on aurait $l(y, f) = -\log(p(f|y))$. Dans un cadre non probabiliste, on peut citer parmi les fonctions plus utilisés le coût quadratique $l(y, f) = \frac{1}{2}(y - f)^2$. On retrouve celui-ci dans la plupart des cas quelque soit le type de problème. Ceci principalement pour la raison qu'il fait partie des coûts à la fois différentiables et convexes, et qu'il n'entraîne pas les problèmes computationnels de certains autres coûts. Si l'on choisit un coût logistique $l(y, f) = \log(1 + e^{-yf})$, un coût charnière $l(y, f) = \max(0, 1 - yf)$ ou encore un coût tangente $l(y, f) = (2 \arctan(yf) - 1)^2$, la répartition des valeurs à la reconstruction aura tendance à être bimodale et pourrait permettre d'effectuer un classement (*classification*) des données. Il est à noter que l'utilisation d'un coût charnière implique que $\forall (i, j) \in \mathcal{Y}, Y_{ij} \in \{-1; 1\}$ et que l'utilisation d'un coût logistique implique que $\forall (i, j) \in \mathcal{Y}, Y_{ij} \in \{0; 1\}$.

D'autres critères, issus de la recherche d'information (*Information Retrieval*), plus sophistiqués mais non-différentiables sont également utilisés pour l'ordonnancement. C'est le cas du nDCG (*normalized Discounted Cumulative Gain*) et de ses variantes utilisées par [Valizadegan et al. 2009]. Le

coût associé au nDCG peut s'écrire, pour $Y_{ij} \in \{0; 1\}$ ou pour $Y_{ij} \in \{0; 1; \dots; a\}$, de la façon suivante :

$$\mathcal{L}(Y, \hat{M}) = \frac{1}{n} \sum_{i=1}^n \frac{1}{p} \sum_{j=1}^p \frac{2^{Y_{ij}} - 1}{\log(1 + r_{ij})},$$

où r_{ij} est le rang attribué à l'article j pour l'utilisateur i par ordonnancement des $\hat{M}_{i\bullet}$ et $Y_{ij} = 1$ si l'article j est effectivement pertinent pour l'utilisateur i . C'est également le cas de la MAP (*Mean Average Precision*) utilisé dans [Shi et al. 2012].

Il est également possible d'utiliser des divergences et en particulier les divergences de Bregman pour la pénalisation comme celles de Kullback-Leibler ($l(y, f) = y \log \frac{y}{f} - y + f$) et d'Itakura-Saito ($l(y, f) = \frac{y}{f} - \log \frac{y}{f} - 1$). Ces critères sont fréquemment utilisés pour la factorisation avec contraintes de positivité (Lee & Seung 2001, Févotte et al. 2009).

2.1.5 Le terme régularisant

Dans le cas de la factorisation, le terme de régularisation Ω est généralement de la forme $\Omega(F) = \Omega(U, V) = \Omega_1(U) + \mu \Omega_2(V)$, où μ est un paramètre d'équilibrage et les Ω_i sont généralement des normes. Le but de ce terme est comme son nom l'indique d'obtenir une solution plus régulière (plus simple), mais également de transformer un problème initialement mal posé en un problème bien posé. Ceci se traduit principalement par un contrôle des valeurs de la solution (réduction de rang, parcimonie...). Deux classes de normes sont traditionnellement associées aux matrices : les normes induites associées à la matrice en tant qu'opérateur et les normes associées au vecteur de taille $n \times p$ formé par tous les éléments de la matrice.

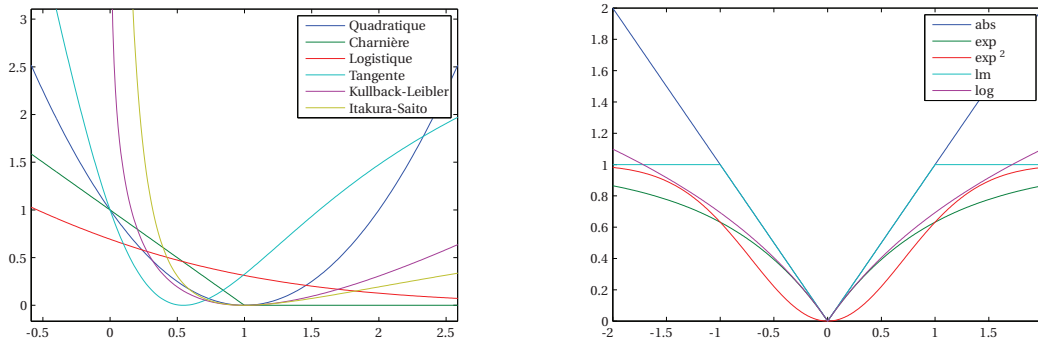
Les normes induites (ou d'opérateur) se définissent de la façon suivante :

$$\|F\|_{p \rightarrow q} = \sup_{u \in \mathbb{R}^p} \frac{\|Fu\|_q}{\|u\|_p}, \quad (2.7)$$

où $(p, q) \in \{1, 2, \dots, \infty\}^2$ dans la plupart des cas. Lorsque $p = q = 1$ il s'agit de la norme L_1 et $\|F\|_{1 \rightarrow 1} = \max_j \sum_i |F_{ij}|$, lorsque $p = q = \infty$ il s'agit de la norme L_∞ et $\|F\|_{\infty \rightarrow \infty} = \max_i \sum_j |F_{ij}|$, et enfin lorsque $p = q = 2$ il s'agit de la norme euclidienne et $\|F\|_{2 \rightarrow 2} = \max_i |\sigma_i|$ (σ_i désigne les valeurs singulières de F). On retrouve l'utilisation de normes de ce type notamment dans [Srebro & Shraibman 2005], où est définie la norme max de la façon suivante :

$$\|F\|_{\max} = \min_{U, V} \|U\|_{2 \rightarrow \infty} \|V\|_{2 \rightarrow \infty}. \quad (2.8)$$

D'autres normes ou pénalités présentent des avantages et sont très souvent utilisées. La plus répandue est la norme de Frobenius : $\Omega(F) = \|F\|_{Fro}^2 = \sum_i \sum_j F_{ij}^2$. Sa convexité et sa différentiabilité la rendent intéressante. Un autre terme régularisant très utilisé est $\Omega(F) = \text{rang}(F)$ mais on pourra également trouver une relaxation que l'on appelle la norme nucléaire : $\Omega(F) = \|F\|_* = \sum \sigma_i$, où σ_i désigne la i -ème valeur singulière de F . Cette relaxation est utilisée par [Recht et al. 2010]. D'autres normes permettent de régulariser les matrices tout en induisant la parcimonie. Il s'agit notamment de la norme zéro définie par $\Omega(F) = \|F\|_0 = \sum_i \sum_j \mathbb{1}_{F_{ij} \neq 0}$, c'est-à-dire le nombre de valeurs non nulles dans la matrice (comme dans [Lee & Seung 2001]). Cependant cette norme est non convexe et discontinue, on préférera souvent utiliser des relaxations continues du problème, en utilisant entre autres, celle que nous désignerons comme étant la norme $\|\cdot\|_1$ (qui ne désignera pas la norme d'opérateur L_1 noté $\|\cdot\|_{1 \rightarrow 1}$) et qui s'écrit $\|F\|_1 = \sum_i \sum_j |F_{ij}|$ (notamment utilisée par [Candès et al. 2011]). On pourra trouver d'autres approximations de la norme zéro, comme l'approximation exponentielle $\|F\|_e = \sum_i \sum_j (1 - e^{-a|F_{ij}|})$, $a > 0$ (Bradley & Mangasarian 1998), l'approximation

FIGURE 2.1 – Réprésentation de différents coûts (pour $y = 1$) et de normes utilisées

linéaire par morceaux $\|F\|_{lm} = \sum_i \sum_j \min(1, \frac{|F_{ij}|}{a})$, $a > 0$, ou encore l'approximation logarithmique $\|F\|_{\log} = \sum_i \sum_j \log(1 + \frac{|F_{ij}|}{a})$, $a > 0$ (Weston *et al.* 2003). On pourra également trouver des versions dérivables en 0 de ces normes mais qui n'induiront plus la parcimonie, comme la norme exponentielle carrée $\|F\|_{e^2} = \sum_i \sum_j (1 - e^{-a F_{ij}^2})$, $a > 0$. Il est aussi possible de définir sur les matrices des normes mixtes de la forme $\|F\|_{2,1} = \sum_i (\sum_j F_{ij}^2)^{1/2}$ que l'on peut généraliser avec des ordres quelconques q et q' (Kowalski 2009) :

$$\|F\|_{q,q'} = \sum_i (\sum_j F_{ij}^q)^{1/q'}. \quad (2.9)$$

Enfin on peut trouver d'autres types de termes régularisants tel que le MCP (*Minimax Concave Penalty* introduit par [Bruce & Gao 1996]), le SCAD (*Smoothly Clipped Absolute Deviation* introduit par [Fan & Li 2001]), l'adaptive lasso (introduit par [Zou 2006]). Ces pénalités servent à induire la parcimonie comme la norme $\|\cdot\|_1$ tout en cherchant à supprimer le biais de cette dernière. Nous reviendrons sur ces pénalités dans le chapitre 5 car elles sont intéressantes en sélection de modèles.

La figure 2.1 représente quelques-unes des normes énumérées dans ce paragraphe.

2.1.6 La nature des problèmes

Suivant le choix du critère ou du terme régularisant, la nature du problème peut varier. En effet, celui-ci peut être convexe ou non convexe ainsi que différentiable ou non. Lorsque le problème est différentiable, on va chercher à annuler le gradient de la fonction Lagrangienne du problème pour trouver, dans le cas convexe le minimum global, et dans le cas non-convexe un minimum local. Dans le cas non-différentiable (notamment lorsque l'on cherche à induire la parcimonie), il n'est pas possible de calculer le gradient. Ainsi pour un problème convexe, il faut trouver les solutions pour lesquelles zéro appartient à la sous-différentielle du Lagrangien (Hiriart-Urruty & Lemarechal 1993). Pour un problème non-convexe il faudra trouver les solutions pour lesquelles zéro appartient à la sous-différentielle au sens de Clarke (Sra 2011) ou des variations de cette sous-différentielle comme celle de Penot introduite dans [Penot 1984]. La nature du problème dépend aussi du choix du critère et du terme régularisant (par exemple, l'utilisation d'une norme $\|\cdot\|_1$ rendra le problème non différentiable, mais celui-ci restera convexe s'il l'était déjà).

Le tableau 2.1 résume la nature du problème suivant le choix du critère d'attache aux données et du terme régularisant. La somme d'un problème convexe avec un non convexe est non convexe, et la somme d'un problème différentiable avec un non différentiable est non différentiable.

		$l(y,f)$	reg term
convexe	diff	$\frac{1}{2}(y-f)^2$ $1 + e^{-yf}$ Kullback-Leibler Itakura-Saito	$\ \cdot\ _{Fro}$ $\ \cdot\ _{2 \rightarrow 2}$
	non-diff	$\max(0, 1 - yf)$	$\ \cdot\ _{1 \rightarrow 1}$ $\ \cdot\ _{1 \rightarrow q}$ $\ \cdot\ _{q \rightarrow 1}$ $\ \cdot\ _1$ $\ \cdot\ _*$ AdaLasso
non-convexe	diff	$(2 \arctan(yf) - 1)^2$	$1 - e^{(-af^2)}$
	non-diff	nDCG MAP	$\ \cdot\ _0$ $rg(\cdot)$ $\ \cdot\ _e$ $\ \cdot\ _{log}$ $\ \cdot\ _{lm}$ MCP SCAD

TABLE 2.1 – tableau récapitulatif de la nature du problème suivant le critère et le terme régularisant

De plus la nature du problème de factorisation avec optimisation simultanée (ou alternée) de U et V fait que le problème est non-convexe. Cependant des preuves de convergences globales ont été faites pour la plupart des algorithmes (descente de gradient stochastique, algorithmes multiplicatifs) sur lesquels nous reviendrons dans le paragraphe 2.2.

2.1.7 La matrice de bruit

Nous nous intéressons ici à la matrice de bruit R du modèle $Y = M + R$. Deux hypothèses sont généralement faites quant à cette matrice de bruit : R est une matrice aléatoire centrée et R ne dépend pas de M .

Il y a ensuite trois grands niveaux de généralité de loi de probabilité que peut suivre cette matrice aléatoire. Le premier niveau et le plus générale regroupe les lois à symétrie elliptique. Le deuxième niveau regroupe les lois à symétrie sphérique (comme la loi de Kotz ou la loi de Student par exemple). Le dernier niveau de généralité garantit l'indépendance des variables et regroupe des lois comme la loi normale.

Pour chacune des lois de probabilité, il y a trois façons d'appréhender cette matrice aléatoire. La première approche est une définition matricielle de la loi, on considère ici la matrice R comme étant une réalisation de la loi de probabilité. La deuxième approche est une vision vectorielle, et l'on considère ici le vecteur $\text{vec}(R)$ comme étant une réalisation de la loi de probabilité. Une description de ces deux approches est disponible dans [Fang 2004]. Enfin, la troisième approche est une vision scalaire de la loi et l'on considère que chaque élément de la matrice est une réalisation de la loi de probabilité.

2.2 La factorisation dans la recommandation

Nous allons maintenant survoler certaines des méthodes communément utilisées en factorisation pour la recommandation suivant la nature du problème. Des informations complémentaires peuvent être trouvées dans [Singh & Gordon 2008] (table 1 page 364) ou dans le tutoriel de KDD 2012

de Lars Schmidt-Thieme et Steffen Rendle³.

2.2.1 Méthode de résolution exacte sur grandes matrices creuses

Nous allons présenter dans ce paragraphe des algorithmes permettant d'effectuer une décomposition en valeurs singulières de la matrice Y , et essentiellement celui proposé par [Larsen 1998] qui utilise un processus de Lanczos. Mais avant cela, nous allons repartir du problème originel.

Historiquement, l'un des premiers problèmes abordés dans ce cadre est celui de l'approximation matricielle de faible rang qui s'écrit :

$$\begin{aligned} \min_{\hat{M} \in \mathbb{R}^{n \times p}} \quad & \|Y - \hat{M}\|_F^2, \\ \text{s.t.} \quad & \text{rang}(\hat{M}) = k. \end{aligned} \quad (2.10)$$

On cherche ici à obtenir la matrice de rang le plus faible possible qui soit le plus proche possible de la matrice Y (à un ϵ près). Le problème formulé dans l'expression lagrangienne 2.11 est équivalent au problème 2.10.

$$\min_{\hat{M} \in \mathbb{R}^{n \times p}} \quad \|Y - \hat{M}\|_F^2 + \lambda \text{rang}(\hat{M}). \quad (2.11)$$

La solution optimale unique à ce problème est donnée par la décomposition en valeurs singulières (SVD) partielle comme le montre [Eckart & Young 1936]. Comme la décomposition n'est que partielle la solution n'est pas exacte, mais le serait pour $k = \text{rang}(Y)$. Une fois que l'on a obtenu $U \in \mathbb{R}^{n \times k}$ et $V \in \mathbb{R}^{p \times k}$ pour les k premiers facteurs de la décomposition correspondant aux k plus grandes valeurs singulières, on reconstruit la matrice $\hat{M}$ de la façon suivante :

$$\hat{M} = UV^\top \quad (= U'\Sigma V'^\top),$$

où $\Sigma \in \mathbb{R}^{k \times k}$ est la matrice diagonale contenant les k premières valeurs singulières de Y , $U' \in \mathbb{R}^{n \times k}$ et $V' \in \mathbb{R}^{p \times k}$ sont les matrices contenant les vecteurs singuliers associés aux valeurs singulières de Y .

2.2.1.1 Principe général de la décomposition en valeurs singulières sur grandes matrices creuses

Pour obtenir la décomposition en valeurs singulières de la matrice Y , on calcule généralement les valeurs propres de la matrices $Y^\top Y$ ou $Y Y^\top$ et on en déduit les valeurs singulières de la matrice Y . L'algorithme choisi dans ce cas est un algorithme de la puissance itérée.

Dans le cas où les dimensions de la matrice Y sont grandes, un algorithme de la puissance itérée pour calculer la décomposition en valeurs singulières n'est plus vraiment une option intéressante pour deux raisons : Le calcul explicite de la matrice $Y^\top Y$ ou de la matrice $Y Y^\top$ est trop coûteux et ces matrices ne sont plus parcimonieuses. On utilisera plutôt un algorithme comme celui initialement proposé par [Golub & Kahan 1965] mais repris et rendu utilisable par [Larsen 1998]⁴.

Dans un premier temps, on effectue une bi-diagonalisation de la matrice Y . Il s'agit de calculer deux matrices orthogonales W et Z telles que $W^\top Y Z = B$, avec B une matrice bi-diagonale de la forme

3. www.ismll.uni-hildesheim.de/aktuelles/kdd_en.html

4. Larsen fournit un package matlab appelée PROPACK qui implémente sa méthode de façon optimisée pour les problèmes de grandes matrices creuses (soi.stanford.edu/~rmunk/PROPACK/)

$$B = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & \ddots \\ & & & & \beta_{p-1} \\ & 0 & & & & \alpha_p \end{pmatrix}.$$

Dans un second temps, on cherche à diagonaliser la matrice B , il s'agit ici aussi de trouver deux matrices orthogonales S et T telles que $S^\top B T = \Sigma$, avec Σ la matrice diagonale contenant les valeurs singulières de B qui sont également celles de Y si les matrices S , T , W et Z sont normalisées. Ainsi, nous obtenons notre décomposition en valeurs singulières $Y = U \Sigma V^\top$, où $U = WS$ et $V = ZT$.

2.2.1.2 Bi-diagonalisation

La forme adaptée aux matrices creuses de la factorisation bi-diagonale à été proposée par [Paige 1974] sous le nom de bi-diagonalisation de Lanczos. Si l'on considère la décomposition bi-diagonale de la matrice Y , on peut montrer que

$$Y Z = W B \quad \text{et} \quad Y^\top W = Z B^\top.$$

À partir de ces formules et en raisonnant colonne par colonne on en déduit d'une manière générale que

$$\begin{aligned} \beta_k X(:, i+1) &= Y^\top W(:, i) - \alpha_i Z(:, i), \\ \alpha_{i+1} W(:, i+1) &= Y Z(:, i+1) - \beta_i W(:, i), \end{aligned}$$

où $W(:, i)$ représente toujours la i -ème colonne de la matrice W . On peut ainsi construire les suites β_i , $Z(:, i+1)$, α_{i+1} et $W(:, i+1)$ à partir d'un vecteur initial $Z(:, 1)$.

Entrées : $Y \in \mathbb{R}^{n \times p}$, $z \in \mathbb{R}^n$, $k \in \mathbb{R}$

$Z(:, 1) \leftarrow z$

$w \leftarrow Y Z(:, 1)$

$\alpha_1 \leftarrow \text{norme}(w)$

$W(:, 1) \leftarrow w / \alpha_1$

Pour $i = 1$ to $k - 1$ **faire**

$w \leftarrow Y^\top W(:, i) - \alpha_i Z(:, i)$

$\beta_i \leftarrow \text{norme}(w)$

$Z(:, i+1) \leftarrow w / \beta_i$

$w \leftarrow Y Z(:, i+1) - \beta_i W(:, i)$

$\alpha_{i+1} \leftarrow \text{norme}(w)$

$W(:, i+1) \leftarrow w / \alpha_{i+1}$

Fin Pour

Retourner W, Z, α, β

Algorithme 1 – Factorisation bi-diagonale

L'efficacité de cet algorithme tient du fait qu'il utilise des produits matrice-vecteur et tire ainsi partie de la nature creuse de la matrice Y . Cependant, si ce schéma s'avère séduisant, il souffre néanmoins d'un inconvénient majeur : la perte de l'orthogonalité due à des problèmes numériques. Ainsi les vecteurs des matrices $W(:, i)$ et $Z(:, i)$ qui sont censés être orthogonaux aux vecteurs déjà calculés ne le sont pas dans la pratique. C'est ici qu'intervient [Larsen 1998] en proposant un schéma de réorthogonalisation partielle qui permet aux matrices W et Z de garder leur orthogonalité tout en optimisant le temps de calcul.

2.2.1.3 Diagonalisation d'une matrice bi-diagonale

Maintenant que la bi-diagonalisation est effectuée, il nous reste à calculer la décomposition en valeurs singulières de B . Mais ce problème s'avère moins complexe que la bi-diagonalisation car la taille de la matrice B et son caractère bi-diagonal la rendent plus facilement abordable. En effet, si l'on souhaite calculer les k plus grandes valeurs singulières, il suffit d'arrêter la bi-diagonalisation après k itérations. La matrice B est donc de taille k . Une première approche consiste à calculer les valeurs propres de la matrice tri-diagonale $B^T B$ en utilisant des rotations de Givens avec décalage. Ceci montre d'une certaine manière que ce problème est en effet moins complexe que celui de la bi-diagonalisation. Mais le calcul explicite de $B^T B$ peut s'avérer dangereux d'un point de vue numérique comme expliqué par [Golub & Van Loan 1989], où des instabilités numériques risquent d'apparaître. Les auteurs suggèrent un autre schéma toujours basé sur des rotations de Givens mais appliqué à la matrice B directement.

Cette méthode de factorisation est très adaptée dans un problème d'approximation sur des matrices creuses de grande taille.

Nous recommandons l'utilisation du package PROPACK⁵ développé pour Matlab qui est plus performant en terme de temps de calcul et de mémoire que la fonction *svd*s disponible sous Matlab⁶.

2.2.2 Descente de gradient stochastique (SGD)

Dans un problème de factorisation, lorsque le coût est différentiable, les méthodes les plus simples et les plus efficaces en terme de temps de calcul sont très certainement les méthodes de type descente de gradient, car elles tirent très bien avantage de l'aspect parcimonieux de la matrice. La formulation du problème est la suivante : $\min_{U,V} J(Y, U, V) = \mathcal{L}(UV, Y) + \Omega(U, V)$ avec $\mathcal{L}$ et Ω qui sont tous les deux différentiables.

La formulation la plus simple des règles de mise à jour des facteurs dans le cas d'une factorisation est la suivante :

$$\begin{aligned} - U_i^{NEW} &= U_i^{OLD} - \rho \nabla_{U_i} J(Y, U, V) \quad \forall i, \\ - V_j^{NEW} &= V_j^{OLD} - \rho \nabla_{V_j} J(Y, U, V) \quad \forall j, \end{aligned}$$

où ρ désigne le pas, celui ci pouvant être fixé (généralement dans les cas où la taille des données rend la recherche du pas prohibitive) ou évolutif (méthodes de type *linesearch* respectant diverses conditions comme celles de Wolfe ou de Goldstein). Dans les types de problèmes qui nous concernent, la taille des données interdit souvent la recherche d'un pas et celui-ci est sélectionné par validation croisée au cours du processus d'optimisation. Beaucoup d'algorithmes de factorisation sont basés sur ces règles de mise à jour comme la *Maximum Margin Matrix Factorisation* (MMMF) comme le proposent [Srebro et al. 2005, Weimer et al. 2008b].

5. soi.stanford.edu/~rmunk/PROPACK/

6. dgleich.wordpress.com/2013/10/19/svd-on-the-netflix-matrix/

Mais la formulation la plus utilisée désormais est celle proposée par Funk (2006)⁷. Elle se base sur le caractère parcimonieux de la matrice et les variables latentes pour un utilisateur i donné ou un article j donné ne sont pas mises à jour simultanément. Le détail de ce processus est décrit dans [Takacs et al. 2009] et présenté dans l'algorithme 2 pour un terme régularisant $\Omega(U, V) = \lambda_1 \|U\|^2 + \lambda_2 \|V\|^2$ (le terme de droite des règles de mise à jour peut changer pour un terme régularisant différent).

```

Entrées :  $Y, \rho, \lambda_1, \lambda_2, d$ 
Initialiser  $U$  et  $V$ 
Pour  $k \in \{1, \dots, d\}$  faire
    Pour  $(i, j) \in \mathcal{Y}$  faire
         $e_{ij} \leftarrow \nabla \mathcal{L}(U_{ik} V_{kj}, Y_{ij})$ 
         $U_{ik} \leftarrow U_{ik} + \rho(e_{ij} V_{kj} - \lambda_1 U_{ik})$ 
         $V_{kj} \leftarrow V_{kj} + \rho(e_{ij} U_{ik} - \lambda_2 V_{kj})$ 
    Fin Pour
Fin Pour

```

Algorithme 2 – Processus de Funk

Une version alternative de cet algorithme est parfois utilisée. Cette version vise à entrelacer encore plus les mises à jour des valeurs des matrices U et V et il a été remarqué que le taux de convergence était meilleur. Cet version est présentée dans l'algorithme 3.

```

Entrées :  $Y, \rho, \lambda_1, \lambda_2, d$ 
Initialiser  $U$  et  $V$ 
Pour  $k \in \{1, \dots, d\}$  faire
    Pour  $(i, j) \in \mathcal{Y}$  faire
         $e_{ij} \leftarrow \nabla \mathcal{L}(U_{ik} V_{kj}, Y_{ij})$ 
         $U_{ik} \leftarrow U_{ik} + \rho e_{ij} V_{kj}$ 
         $V_{kj} \leftarrow V_{kj} + \rho e_{ij} U_{ik}$ 
         $U_{ik} \leftarrow U_{ik} - \rho \lambda_1 U_{ik}$ 
         $V_{kj} \leftarrow V_{kj} - \rho \lambda_2 V_{kj}$ 
    Fin Pour
Fin Pour

```

Algorithme 3 – Processus de Funk 2.0

Cependant, il est important de noter que ce type de méthode présente un inconvénient majeur : malgré un algorithme rapide et performant, il se trouve que la phase de sélection de modèle (réalisée la plupart du temps par validation croisée) s'avère difficile en pratique à cause de la quantité de paramètres, leur grande dépendance aux données, et leur forte instabilité (en particulier le paramètre ρ appelé pas d'apprentissage ou *learning rate* en anglais). On peut trouver des travaux récents

7. <http://sifter.org/~simon/journal/20061211.html>

qui tentent de contourner cette difficulté comme [Schaul *et al.* 2012] qui adapte légèrement le pas d'apprentissage au cours du processus d'optimisation.

Cette méthode est utilisée aussi bien pour des problèmes d'approximation, de complétion que de ranking.

2.2.3 Optimisation alternée

Tout comme dans la descente de gradient stochastique, nous partons d'une fonction objectif différentiable de formulation standard : $\min_{U,V} \mathcal{L}(UV, Y) + \Omega(U, V)$.

L'idée de cette méthode de résolution de factorisation est d'obtenir les règles de mise à jour des facteurs en calculant le gradient par rapport à ce facteur, et en trouvant une solution analytique. Ceci se fait généralement en annulant le gradient et en isolant le terme à mettre à jour pour connaître sa règle de mise à jour.

Prenons l'exemple du coût suivant : $\min_{U,V} J = \frac{1}{2} \sum_{ij} (U_{i\bullet} V_{\bullet j} - Y_{ij})^2 + \lambda(\|U\|_F + \|V\|_F)$. Le gradient par rapport à $U_{i\bullet}$ (la i -ème ligne de U) se calcule facilement : $\nabla_{U_{i\bullet}} J = (U_{i\bullet} V - Y_{i\bullet}) V^\top + \lambda \mathbf{1}$. Ainsi en annulant la dérivée et en isolant $U_{i\bullet}$ de l'équation nous obtenons $U_{i\bullet} = (Y_{i\bullet} V^\top) (V V^\top + \lambda I)^{-1}$, qui sera la règle de mise à jour de $U_{i\bullet}$ dans notre algorithme. La règle de mise à jour de $V_{\bullet j}$ (la j -ième colonne de V) s'obtient de la même façon. On en déduit l'algorithme d'optimisation qui à chaque itération met à jour l'ensemble des vecteurs $U_{i\bullet}$ et ensuite l'ensemble des vecteurs $V_{\bullet j}$ jusqu'à convergence. Ce type d'algorithme a été introduit par [Young *et al.* 1976].

Des versions évoluées de cet algorithme ont été proposées dans la littérature comme les méthodes proposées par [Hu *et al.* 2008, Pan *et al.* 2008] qui sont optimisées pour la factorisation d'une matrice parcimonieuse et permet de gérer les données de très grandes tailles pour des problèmes d'ordonnement dans le cadre de la recommandation. Les auteurs choisissent, dans les deux contributions, un critère d'attache aux données de la forme $\mathcal{L}(Y, U, V) = \sum W_{ij} (Y_{ij} - U_{i\bullet} V_{\bullet j})^2$. [Hu *et al.* 2008] gère des données d'appréciations implicites en résolvant un problème de complétion, tandis que les données de [Pan *et al.* 2008] contiennent des appréciations positives et négatives et le problème est transformé en un problème de classement. Dans les deux cas les règles de mise à jour des facteurs sont les suivantes :

$$\begin{aligned} U_{i\bullet} &= Y_{i\bullet} \tilde{W}^{(i)} V^\top (V \tilde{W}^{(i)} V^\top + \lambda I)^{-1}, \\ V_{\bullet j} &= (U^\top \tilde{W}^{(j)} U + \lambda I)^{-1} U^\top \tilde{W}^{(j)} Y_{\bullet j}, \end{aligned} \quad (2.12)$$

avec $\tilde{W}^{(i)}$ et $\tilde{W}^{(j)}$ des matrices diagonales telles que $W_{ij} = \tilde{W}_{jj}^{(i)} = \tilde{W}_{ii}^{(j)}$. Seul le choix des valeurs de W varie d'un modèle à l'autre suivant la nature des données et du problème. En plus de gérer le type de problème lié aux données, le choix de W permet également de tirer avantage de la nature parcimonieuse des données. Cette façon de tirer avantage de la parcimonie est présentée dans le paragraphe 4.3.2 dans un problème précis.

À titre de remarque, il est important de noter qu'il n'existe pas nécessairement de solution analytique au problème et que l'existence d'une telle solution va dépendre du choix des critères d'attache aux données et de régularisation. Pour cette raison, il est très rare que le coût choisi ne soit pas un coût quadratique. Ce type d'algorithme est alors appelé moindres carrés alternés (ALS).

2.2.4 Algorithme Multiplicatif

Les algorithmes multiplicatifs pour la factorisation de matrice ont été introduits par [Lee & Seung 1999]. L'idée générale des algorithmes multiplicatifs est de progresser dans le processus d'op-

timisation en multipliant les paramètres à optimiser par un terme (qui tendra vers la valeur 1) à chaque itération, contrairement aux méthodes additives telles que les descentes de gradient ou les algorithmes de Newton où l'on avance dans le processus d'optimisation en ajoutant un terme à chaque étape. Ainsi lorsque l'on cherche à factoriser $Y \in \mathbb{R}^{n \times p}$ sous la forme UV (avec $U \in \mathbb{R}^{n \times d}$ et $V \in \mathbb{R}^{d \times p}$), les règles de mise à jour de U et V sont de la forme suivante :

$$\begin{aligned} U &\leftarrow U \otimes \frac{\nabla_U^- J(Y, U, V)}{\nabla_U^+ J(Y, U, V)}, \\ V &\leftarrow V \otimes \frac{\nabla_V^- J(Y, U, V)}{\nabla_V^+ J(Y, U, V)}, \end{aligned} \quad (2.13)$$

où $\nabla^- J$ (respectivement $\nabla^+ J$) désigne la partie négative (respectivement positive) du gradient de J , où $\otimes$ désigne le produit de Hadamard (multiplication terme à terme), et où la barre de fraction désigne le quotient de Hadamard (division terme à terme). Les problèmes qui utilisent le plus souvent ce genre d'algorithme pour leur résolution sont les problèmes de factorisation non-négative (NMF). Notons qu'il est possible d'ajouter des termes régularisants au coût suivant la nature du problème. Par exemple, [Hoyer 2002, Peharz & Pernkopf 2012] régularisent afin d'induire la parcimonie dans leur modèle.

Pour mieux comprendre le fonctionnement des algorithmes de type multiplicatif, prenons l'exemple suivant :

Exemple 2.1. Supposons que l'on se pose le problème d'optimisation suivant

$$\begin{aligned} \min_{U, V} J(Y, U, V) &= \frac{1}{2} \sum_{i,j} (Y_{ij} - U_i V_j)^2 + \lambda \|U\|_F^2 + \mu \|V\|_F^2, \\ \text{s.t. } U &\geq 0, V \geq 0. \end{aligned}$$

Les gradients de la fonction J par rapport à U et V sont :

$$\Leftrightarrow \begin{cases} \nabla_U J = & -(Y - UV)V^\top + 2\lambda U, \\ \nabla_V J = & -U^\top(Y - UV) + 2\mu V, \\ \nabla_U J = & -YV^\top + U(VV^\top + 2\lambda I_p), \\ \nabla_V J = & -U^\top Y + (U^\top U + 2\mu I_n)V. \end{cases}$$

On peut très clairement identifier, dans l'expression du gradient, une partie où les termes sont négatifs et une où les termes sont positifs. On définit alors $\nabla_U^- J = YV^\top$ et $\nabla_U^+ J = U(VV^\top + 2\lambda I_p)$. En faisant de même pour le gradient par rapport à V on obtient les règles de mise à jour de notre algorithme :

$$\begin{aligned} U &\leftarrow U \otimes \frac{YV^\top}{UVV^\top + 2\lambda I_p}, \\ V &\leftarrow V \otimes \frac{U^\top Y}{U^\top UV + 2\mu I_n}. \end{aligned}$$

L'algorithme consiste à itérer successivement ces règles de mise à jour jusqu'à convergence globale du modèle vers un minimum local. Et dans l'hypothèse où $Y \geq 0$, les contraintes de positivité sur U et V sont respectées.

Pour une matrice initiale Y telle que $Y_{ij} \geq 0$, ces algorithmes multiplicatifs garantissent la positivité des facteurs U et V . Comme toutes les méthodes de factorisation, elle est basée sur la minimisation d'un coût, et l'on peut trouver dans la littérature trois principaux coûts pour $L(Y, \hat{M})$. Le coût le plus usuel est le coût quadratique, mais il est également possible d'observer l'utilisation de divergences telles que celle de Kullback-Leibler (Lee & Seung 2001) et celle d'Itakura-Saito (Févotte *et al.* 2009).

La NMF est utilisée pour résoudre plusieurs classes de problèmes. On peut citer notamment l'approximation, la complétion, l'ordonnancement, le co-clustering. Par exemple [Févotte *et al.* 2009] fait de la décomposition de signaux musicaux où une factorisation d'une analyse temps-fréquences est effectuée afin d'identifier les notes jouées ou les voix aux différents instants du signal musical. Il s'agit là d'un problème de classification. Tout comme dans [Labiod & Nadif 2011b], où une tri-factorisation non négative est utilisée pour résoudre un problème de co-clustering.

Cette factorisation a été utilisée également lors du défi Netflix par l'équipe gagnante dans son mélangeur de modèle [Bell *et al.* 2009]. Dans le cas où la matrice Y contient des données manquantes, il existe des méthodes adaptées permettant la résolution de problèmes de complétion de matrices non négatives (Pessiot *et al.* 2006).

2.2.5 Approches non linéaires

Malgré les problèmes de calculs numériques liés aux méthodes à noyaux et le temps de calcul lié à la taille des données souvent prohibitive, certaines méthodes non linéaires utilisant une approche de type « machine à noyaux » ont été déjà abordées (comme par exemple par [Abernethy *et al.* 2009]).

Dans ce contexte, les utilisateurs et les articles sont vus comme des points dans un espace de Hilbert à noyaux reproduisants, et l'on mesure l'appétence de l'utilisateur i pour l'article j par le produit scalaire $f(u_i, a_j) = \langle u_i, Ga_j \rangle$, où G est un opérateur à deux noyaux. Dans le cas linéaire on a $\hat{M}_{ij} = Y_{i\bullet} G Y_{\bullet j}$ avec G une matrice de taille $p \times n$. Ainsi, lorsque l'on utilise le principe de minimisation du coût régularisé (équation 2.1) avec un terme de pénalisation de type Hilbert-Schmidt adapté, on obtient un théorème de représentation qui stipule que la solution s'écrit comme une combinaison linéaire des observations $G = \sum_{ij} \alpha_{ij} Y_{i\bullet}^\top Y_{\bullet j}^\top$.

Un choix astucieux dans la construction du noyau permet de traiter des problèmes de complétion de matrice et des problèmes analogues d'apprentissage multitâche et d'apprentissage par pair. On ne dispose pas encore aujourd'hui d'algorithme permettant de passer à l'échelle avec ce type d'approche.

2.2.6 Factorisation Tensorielle

Une matrice reflète une relation d'arité deux (par exemple entre un utilisateur et un produit). Un tenseur représente une relation d'arité supérieure à deux (par exemple une relation entre un utilisateur et un produit indexé dans le temps). Un tenseur est un tableau multidimensionnel et une matrice n'est qu'un cas particulier de tenseur (d'ordre deux). Tout comme les matrices, les tenseurs peuvent être factorisés et nous nous concentrons ici sur la factorisation de tenseur d'ordre trois (mais l'on peut très bien imaginer des tenseurs d'ordres supérieurs). Nous allons présenter dans cette section quelques unes des décompositions tensorielles les plus populaires de la littérature. Bien que nous ne les utilisons pas, nous trouvons ces approches intéressantes et les considérons pour de futurs travaux.

Le principe de décomposition tensorielle a été évoqué pour la première fois par [Hitchcock 1927]. La décomposition qui y est proposée s'est démocratisée lorsqu'elle a été reprise par [Carroll &

[Chang 1970] (sous le terme CANDECOMP pour *canonical decomposition*) et par [Harshman 1970] (sous le terme de PARAFAC pour *parallel factors*) et est appelée décomposition CP (*Canonical Polyadic*). En définissant $T \in \mathbb{R}^{n \times p \times m}$ un tenseur d'ordre trois, la décomposition CP approxime T de la façon suivante :

$$\hat{T}_{ijk} = \sum_{p=1}^d \lambda_i U_{ip} V_{jp} W_{kp}, \quad (2.14)$$

où d est le rang des facteurs U, V et W de la décomposition et les $\lambda_i \in \mathbb{R}$ pondèrent les différentes variables latentes de la décomposition. Dans cette décomposition on cherche à ce que U, V et W soient orthogonales et la décomposition est généralement unique. Il s'agit d'une généralisation à un ordre supérieur de la décomposition partielle en valeurs singulières. La figure 2.2 illustre les facteurs de la décomposition CP (on peut imaginer que les poids λ_i sont intégrés dans l'un des facteurs).

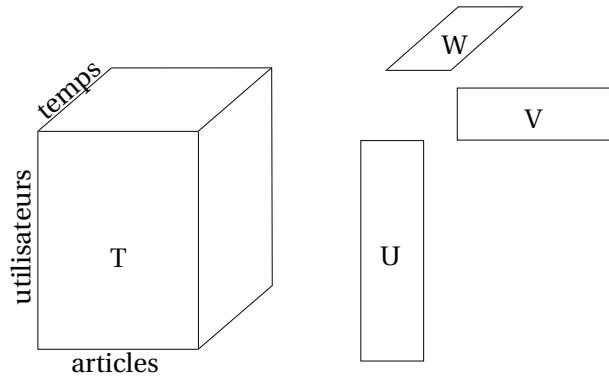


FIGURE 2.2 – Décomposition tensorielle canonique

La décomposition tensorielle de Tucker, introduite par [Tucker 1963], est une généralisation de la décomposition CP de rang d . Elle permet à chacune des matrices de facteurs latents de ne pas avoir nécessairement le même nombre de variables latentes. Pour cela, dans la décomposition apparaît un hypercube $C \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ qui permet cette différence du nombre de variables. La figure 2.3 illustre cette décomposition et la fonction de décision associée à cette décomposition est explicitée dans l'équation 2.15.

$$\hat{T}_{ijk} = \sum_{p=1}^{d_1} \sum_{q=1}^{d_2} \sum_{r=1}^{d_3} C_{pqr} U_{ip} V_{jq} W_{kr}. \quad (2.15)$$

La décomposition tensorielle de Tucker de type 2 (ou Tucker2) a été introduite par [Kroonenberg & De Leeuw 1980]. Dans cette décomposition W est un tenseur. Cette décomposition approxime T de la façon suivante :

$$\hat{T}_{ijk} = \sum_{p=1}^{d_1} \sum_{q=1}^{d_2} U_{ip} V_{jq} W_{pqk}. \quad (2.16)$$

On peut faire l'analogie avec la décomposition de Tucker si l'on choisit la matrice $W = I_m$, on aura $C \in \mathbb{R}^{d_1 \times d_2 \times m}$ qui correspond au W de la décomposition de Tucker2. La figure 2.4 illustre la décomposition de Tucker2.

Le choix de l'algorithme va dépendre (comme pour une factorisation classique) de la modélisation des données (en l'occurrence la décomposition choisie), du problème d'optimisation choisi (critère d'attache aux données, pénalité...) et des contraintes éventuelles. Nous recommandons la lec-

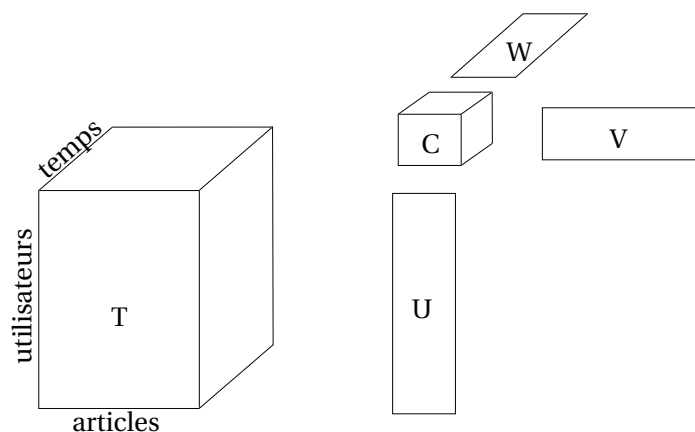


FIGURE 2.3 – Décomposition de Tucker

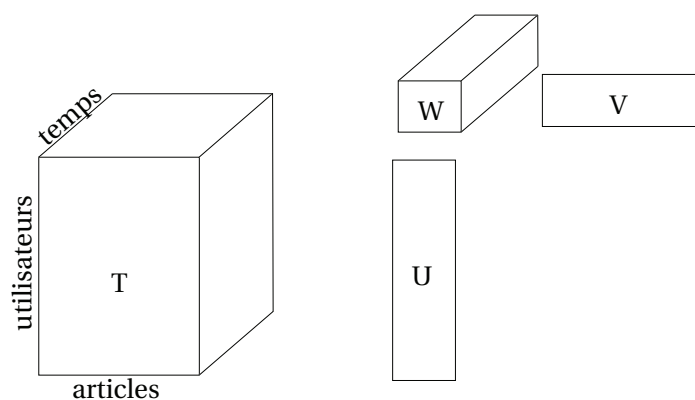


FIGURE 2.4 – Décomposition de Tucker2

ture de [Kolda & Bader 2009] ou le tutoriel d'ICML 2007⁸ où est dressé un état de l'art assez complet des décompositions tensorielles, des algorithmes qui leur sont associées et des applications possibles de ces méthodes. Les algorithmes les plus souvent utilisés dans le domaine de la recommandation sont la descente de gradient stochastique (Rendle *et al.* 2009, Karatzoglou *et al.* 2010, Shi *et al.* 2012) ou l'optimisation des moindres-carrés alternés (Rendle *et al.* 2011, Hidasi & Tikk 2012). Plusieurs librairies qui implémentent les principales décompositions avec différents algorithmes sont disponible en ligne. On peut citer notamment *Tensor Package*⁹ qui implémente plusieurs algorithmes pour résoudre la décomposition CP en Matlab. On peut également citer *Tensor Lab*¹⁰ qui implémente plusieurs décompositions sur des algorithmes Quasi-Newtoniens ou de type ALS en Matlab. Enfin, on peut citer *Factorization Machines Library*¹¹ qui n'est pas à proprement parler une librairie pour la factorisation tensorielle mais permet d'effectuer n'importe quel type de factorisation (cf. [Rendle 2012] pour plus de détails).

8. Mining Large Time-evolving Data Using Matrix and Tensor Tools : <http://www.cs.cmu.edu/~christos/TALKS/ICML-07-tutorial/>

9. www.gipsa-lab.grenoble-inp.fr/~pierre.comon/TensorPackage/tensorPackage.html

10. www.esat.kuleuven.be/sista/tensorlab/

11. www.libfm.org/

2.3 Les méthodes d'évaluation

Une fois que le problème est formulé explicitement, et que l'algorithme de résolution est choisi, il nous faut un processus de sélection et de validation de modèle, c'est-à-dire construire un ensemble de modèles à partir d'un ensemble de paramètres, les comparer les uns aux autres d'une façon adaptée, afin d'obtenir le modèle qui donne les meilleures performances au sens d'une métrique donnée.

Nous allons discuter dans ce paragraphe, des choix communément faits quant à l'ensemble des modèles, à la méthode d'évaluation et à la métrique de comparaison. Nous en discuterons plus particulièrement dans le cadre de la factorisation pour la recommandation.

2.3.1 Le choix des modèles

Dans notre domaine d'application, ce choix de modèles, qui se traduit par un choix d'ensemble de paramètres n'est rarement fait autrement que par *grid search*. C'est-à-dire que l'on choisit pour chaque hyperparamètre, de façon arbitraire (ou non si l'on a une connaissance a priori sur les données), un ensemble de valeurs pour lesquelles on souhaite construire un modèle.

En ce qui concerne le nombre de facteurs latents pour la factorisation, il est choisi de façon plus ou moins automatique. Il est possible d'effectuer de la sélection de modèle automatique comme nous le verrons dans le chapitre 5, tout comme il est possible qu'il soit choisi arbitrairement.

2.3.2 La méthode d'évaluation

Comme pour le choix des modèles, le choix de la méthode d'évaluation dans le domaine qui nous intéresse reste très souvent le même. Il s'agit de la validation croisée (*cross-validation* ou CV), et plus particulièrement de la k-fold CV. Celle-ci est décrite dans l'algorithme 4.

```

Entrées :  $Y_{app}, Y_{test}, k$ 
Découper  $Y_{app}$  en  $k$  sous-ensembles  $Y_{app}^{(k)}$ 
Pour  $p$  un tuple d'hyperparamètres fixé faire
  Pour  $i \in \{1, \dots, k\}$  faire
     $Y_{Atmp} \leftarrow \bigcup_{j \in \{1, \dots, k\}, j \neq i} Y_{app}^{(j)}$ 
     $Y_{Tmp} \leftarrow Y_{app}^{(i)}$ 
     $\hat{f}_p \leftarrow \argmin_f J(f_p(Y_{Atmp}))$ 
     $measure(i) \leftarrow metric(\hat{f}_p(Y_{Tmp}))$ 
  Fin Pour
   $measureVal(p) \leftarrow \frac{1}{k} \sum_i measure(i)$ 
Fin Pour
 $\hat{p} \leftarrow \argmin_p measureVal(p) \quad / \quad \argmax_p measureVal(p)$ 
 $\hat{f}_{\hat{p}} \leftarrow \argmin_f J(f_{\hat{p}}(Y_A))$ 
 $err_{app} \leftarrow metric(\hat{f}_{\hat{p}}(Y_A)) \quad / \quad perf_{app} \leftarrow metric(\hat{f}_{\hat{p}}(Y_A))$ 
 $err_{test} \leftarrow metric(\hat{f}_{\hat{p}}(Y_T)) \quad / \quad perf_{test} \leftarrow metric(\hat{f}_{\hat{p}}(Y_T))$ 

```

Algorithme 4 – Validation croisée

2.3.3 La métrique d'évaluation

Pour évaluer la qualité du modèle, il nous faut choisir une métrique d'évaluation. Nous présentons ici quelques unes des métriques généralement utilisées dans le domaine de la recommandation. Le choix de la métrique va dépendre du problème que l'on souhaite résoudre. En effet, dans le cas d'un problème d'approximation ou de complétion, on choisira traditionnellement une erreur quadratique telle que la RMSE (*Root-Mean-Square Error*) ou la MSE (*Mean-Squared Error*).

Dans le cas d'un problème d'ordonnancement (*ranking*), le choix des métriques est très diversifié mais ces métriques restent généralement issues de la recherche d'information (*Information Retrieval*). On peut par exemple rencontrer l'AUC (*Area Under the Curve*) définie pour un utilisateur i donné par :

$$AUC_i = \frac{\sum_{j \in I_i^+} \sum_{j' \in I_i^-} \mathbb{1}(\hat{Y}_{ij} > \hat{Y}_{ij'})}{|I_i^+| |I_i^-|}, \quad (2.17)$$

avec I_i^+ l'ensemble des articles pertinents pour un utilisateur i et I_i^- ceux non pertinents. Cette métrique est intéressante car facilement interprétable. Elle est utilisée notamment dans [Herlocker et al. 1999, Rendle et al. 2009].

On rencontre très fréquemment la précision, le rappel et la F-mesure (comme dans [Rendle & Lars 2010, Yang et al. 2011]). Ces métriques sont définies de la façon suivante :

$$\begin{aligned} Precision@K &= \frac{\# \text{ articles pertinents recommandés}}{K}, \\ Rappel@K &= \frac{\# \text{ articles pertinents recommandés}}{\# \text{ articles pertinents total}}, \\ F1@K &= \frac{2Precision@K.Rappel@K}{Precision@K + Rappel@K}, \end{aligned} \quad (2.18)$$

où K est le nombre d'articles recommandés. On rencontre également la MAP (*Mean Average Precision*) qui est une moyenne des $Precision@K$ (avec $K \in \{1 \dots n\}$) comme [Xu et al. 2008, Shi et al. 2010].

Une autre métrique intéressante est le nDCG (*normalized Discounted Cumulative Gain*), celle-ci permet de prendre en compte des niveaux de pertinence (à la différence des autres métriques d'ordonnancement où il n'y a qu'un seul degré de pertinence). Elle est utilisée notamment dans [Xu et al. 2008, Yang et al. 2011].

On peut rencontrer le rang moyen utilisé par exemple par [Hu et al. 2008, Fang & Si 2011], cette métrique est plutôt utilisée sur des données d'appréciations implicites. Notons que cette métrique est au rappel ce que la MAP est à la précision.

On peut trouver une présentation détaillée de ces métriques et de quelques autres dans [Herlocker et al. 2004, Pradel 2013].

2.4 Conclusion

De la nature des données (taille, parcimonie...) va dépendre beaucoup la nature du problème que l'on pose et le choix de l'algorithme de résolution. Nous avons vu différentes méthodes fréquemment utilisées dans le domaine d'application qui nous intéresse. Dans un cas général, l'utilisation d'algorithmes de descente gradient stochastique est très justifié mais il est recommandé, lorsque l'on se place dans des cas particuliers, de choisir des algorithmes plus adaptés au problème. Par exemple,

s'il existe, pour un facteur fixé, une solution analytique au problème et qu'il est possible de tirer avantage de la nature parcimonieuse des données, nous recommandons l'utilisation d'un algorithme des moindres carrés alternés (ALS) qui a un bon taux de convergence et une validation croisée rapide.

Dans les contributions présentées dans ce manuscrit, nous sommes confrontés à quelques unes des problématiques énoncées dans ce chapitre.

Dans les chapitres 3 et 4, nous sommes dépendants de la nature particulière des données. Dans le chapitre 3, nous cherchons les méthodes les plus adaptées à des problèmes d'ordonnancement par le biais d'approches contextuelles. Dans le chapitre 4, nous adaptons un algorithme des moindres carrés alternés à des données de nature particulière (données sociales) et de grande taille pour un problème d'ordonnancement.

Dans le chapitre 5, nous cherchons le modèle qui explique le mieux les données et nous essayons d'adapter cette méthode de sélection de modèle à des problèmes de grande taille.

Tout est une histoire de contexte

For me context is the key - from that comes the understanding of everything.

– Kenneth Noland

Sommaire

3.1 Les types de données contextuelles	34
3.1.1 Les variables utilisateurs et produits	34
3.1.2 Les données temporelles	34
3.1.3 Les données sociales	36
3.1.4 Informations contextuelles implicites	36
3.2 Les différents paradigmes de contextualisation	36
3.2.1 La contextualisation par prétraitement	36
3.2.2 La contextualisation par post-traitement	37
3.2.3 La modélisation contextuelle	37
3.3 Un cas d'étude précis : la vente d'outils	38
3.3.1 Présentation du problème	38
3.3.2 Présentation des approches contextuelles	38
3.3.3 Protocole expérimental	39
3.4 Discussion	42

La contextualisation dans les systèmes de recommandation consiste à prendre en compte des données complémentaires pour améliorer les performances de prédiction des modèles. Ce chapitre fait une brève présentation des approches de contextualisation existantes. Nous présentons les types de données contextuelles utilisées et les différentes façons de tirer avantage de cette contextualisation dans la construction du modèle. Nous présentons également une étude de cas réalisée sur des données réelles. Cette étude vise à mettre en exergue l'intérêt de contextualiser les données, notamment d'un point de vue temporel et grâce à une connaissance a priori des données.

Le principe général de la recommandation contextuelle consiste, par opposition au filtrage collaboratif dit « classique » où l'on ne prend en compte que les données de transactions (ou de vote), à ajouter de nouvelles dimensions prenant en compte des données complémentaires, ou en ayant une connaissance a priori sur ces données.

3.1 Les types de données contextuelles

Les données contextuelles par nature sont très variées. Les données les plus couramment rencontrées (et utilisées) dans les systèmes de recommandation sont les variables « utilisateurs » et les variables « produits »¹. Il existe également d'autres informations, aussi bien explicites qu'implicites qui peuvent être utilisées pour améliorer la recommandation (données temporelles, graphe social...). Nous allons maintenant détailler les données utilisées pour la contextualisation.

3.1.1 Les variables utilisateurs et produits

Lorsque l'on parle de variables utilisateurs, il s'agit généralement de données telles que l'âge, le sexe, l'adresse. Il est aussi possible d'utiliser d'autres données propres à l'utilisateur, comme par exemple la position géographique ou l'adresse IP.

Les variables produits diffèrent suivant la nature du produit. Par exemple lorsque les produits à recommander sont des films, il peut s'agir du réalisateur, du genre filmographique ou de la durée. L'intérêt d'utiliser ces variables pour améliorer la recommandation peut être illustré avec l'exemple suivant : les films « Dora l'exploratrice » auront sûrement plus de succès pour un public en bas âge, tandis que « Derrick » intéressera plus un public d'un certain âge. De la même façon, un cliché répandu suggère qu'un public féminin aura peut être plus de plaisir à regarder une comédie romantique qu'un public masculin, qui lui préférera des films d'actions. Ainsi en utilisant le genre cinématographique des films, l'âge et le sexe des utilisateurs comme information de contextualisation, il est très certainement possible d'améliorer la qualité de prédiction d'un système de recommandation de film.

En revanche, si l'on souhaite recommander des restaurants, on s'intéressera plutôt au type de nourriture proposé, au prix, à l'adresse ou encore à l'appréciation donnée par des guides culinaires. On pourra également utiliser la position actuelle de l'utilisateur.

La recommandation de publicité ciblée sur Internet peut être, quant à elle, améliorée en prenant en compte l'adresse IP de l'utilisateur par exemple.

La définition de variable utilisateur ou variable produit est à prendre au sens large, n'importe quelle information sur l'utilisateur ou sur le produit peut être utilisée comme information contextuelle, dès lors qu'elle est susceptible de permettre une meilleure identification du comportement, des goûts de l'utilisateur ou une meilleure catégorisation des produits.

3.1.2 Les données temporelles

L'utilisation d'un contexte temporel est née de l'hypothèse, souvent vérifiée, de l'inconstance des préférences des utilisateurs. En effet, ces préférences peuvent évoluer au cours du temps (Koren 2009, Milkman *et al.* 2009). Certains goûts peuvent naître ou devenir plus prononcés en avançant dans le temps, d'autres peuvent au contraire se résorber ou tout simplement disparaître. Le contexte

1. le terme produit est à prendre au sens large et peut désigner des produits de consommation, des films, des outils, des publicités ciblées, des amis, des services, des lieux, des lignes de code, des itinéraires...

temporel peut aussi être associé à des facteurs de saisonnalité. La figure 3.1 illustre cette évolution des

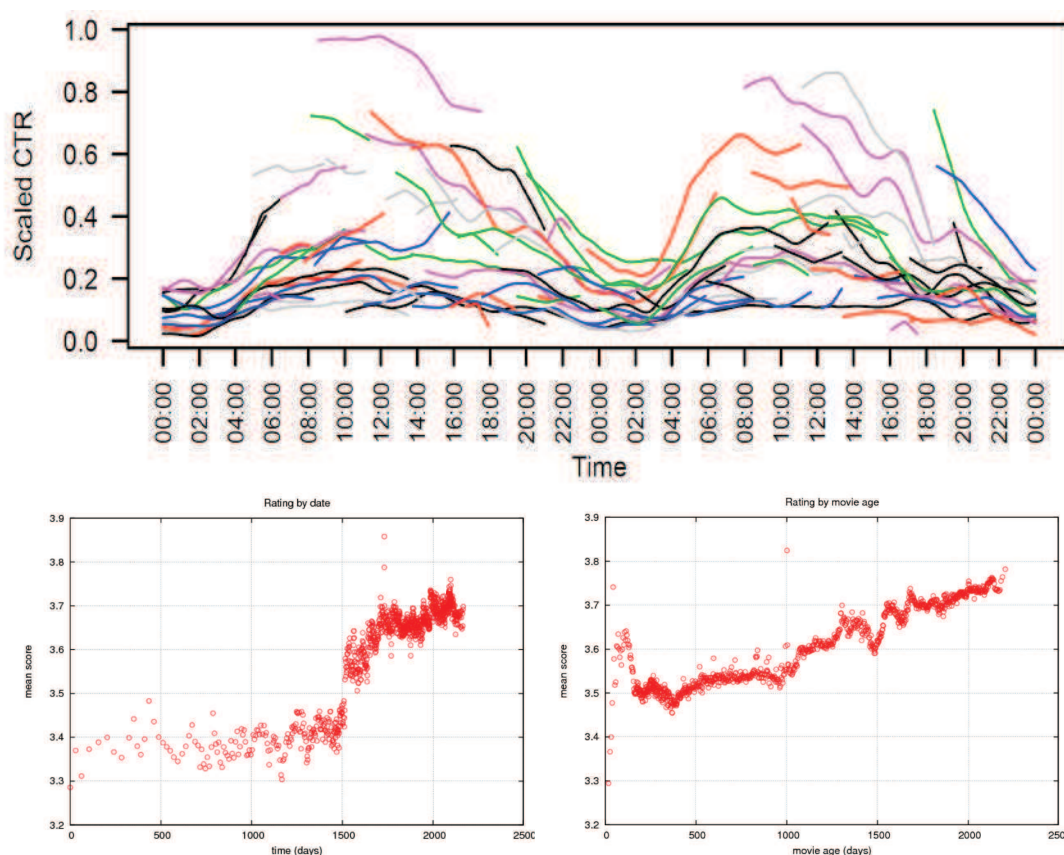


FIGURE 3.1 – Exemples d'évolution de comportement d'utilisateurs au cours de la journée (haut) et de notes au cours du temps (bas)².

comportements au cours du temps². Deux points importants en ressortent : une évolution certaine des comportements des utilisateurs au cours du temps, mais également une évolution de la façon dont sont perçus les articles au cours du temps (par exemple l'achat de maillots de bain est plus important à l'approche des vacances estivales que le reste de l'année). L'intérêt de prendre en compte le facteur temps dans la recommandation nous apparaît évident.

Dans la littérature, il est possible de rencontrer des systèmes de recommandation utilisant des données temporelles de différentes natures (historique de navigation sur un site web comme [Fu *et al.* 2000], historique de position géographique obtenu à l'aide d'un appareil GPS comme [Zheng *et al.* 2011] ou même un historique des achats ou des appréciations de l'utilisateur pour des produits comme [Richard *et al.* 2010]). Dans tous les cas il s'agit globalement d'un ensemble d'actions de l'utilisateur situées dans le temps. Ces actions de l'utilisateur dans le temps peuvent d'ailleurs être à l'encontre d'un même produit comme dans [Amatriain *et al.* 2009], où le système de recommandation tire avantage des données temporelles en invitant l'utilisateur à réexprimer une appréciation pour un produit, en prenant en compte l'éventuelle évolution de préférence au cours du temps afin d'améliorer les prédictions.

La majorité des travaux allant dans ce sens proposent de s'intéresser aux goûts de l'utilisateur sur

2. le graphique du haut est issu de <http://www.cs.rutgers.edu/cs/Media/YahooLectures2011/Agarwal/Agarwal-RecommenderSystems.pdf> et ceux du bas sont issus de [Koren 2009].

une fenêtre temporelle récente d'une longueur variable. C'est l'objet des travaux de [Ding *et al.* 2006] ou de [Zheng *et al.* 2011] par exemple.

3.1.3 Les données sociales

Dans le cadre de la recommandation dans les réseaux sociaux, l'utilisation des données sociales telles que le graphe d'amitié ou graphe de confiance est très commune. On effectue généralement la prédiction des préférences d'un utilisateur en s'intéressant aux préférences des personnes avec qui il est ami ou en qui il a confiance.

Nous reviendrons en détail sur la recommandation dans les réseaux sociaux dans le chapitre 4 car l'utilisation de ce type d'approche contextuelle en est l'objet principal.

3.1.4 Informations contextuelles implicites

Les données présentées précédemment peuvent permettre d'extraire des informations contextuelles de façon explicite. Ces données peuvent ne pas suffire à définir des contextes. Dans certains cas, le contexte doit être déduit à partir des données connues (et grâce éventuellement à une connaissance experte du domaine). On parlera alors d'information contextuelle implicite. Le cas d'étude présenté en section 3.3 de ce chapitre fait justement l'objet d'une contextualisation implicite qui a nécessité une connaissance a priori sur les produits. En effet si, un client achète une baignoire et du carrelage, on peut déduire que ce client s'est lancé dans un chantier de construction (ou de rénovation) de salle de bain, cette information implicite peut être utilisée pour une contextualisation éventuelle.

3.2 Les différents paradigmes de contextualisation

La contextualisation peut intervenir à différents endroits dans un système de recommandation. Trois grands types d'approche de contextualisation sont généralement définis suivant la place (dans le processus de construction du modèle) où cette contextualisation intervient. Ces approches sont les suivantes :

- la contextualisation par prétraitement (*contextual pre-filtering*),
- la contextualisation par post-traitement (*contextual post-filtering*),
- la modélisation contextuelle (*contextual modeling*).

La première approche consiste à ne sélectionner qu'un certain sous-ensemble des données pour la construction du modèle, ou à partitionner les données pour construire un ensemble de modèles. La deuxième, quant à elle, consiste à ne garder en sortie du modèle qu'un sous-ensemble des données. Enfin la troisième approche consiste à utiliser au cours de la construction du modèle les données contextuelles dont nous disposons.

Nous allons présenter brièvement ces trois paradigmes. Pour des informations complémentaires, nous invitons le lecteur à consulter la section 3 de [Adomavicius & Tuzhilin 2011].

3.2.1 La contextualisation par prétraitement

La contextualisation par prétraitement (*contextual pre-filtering*) consiste à sélectionner, pour le contexte dans lequel on se place, un certain sous-ensemble des données significatif pour ce contexte et d'effectuer l'apprentissage du modèle sur ce sous-ensemble. Ceci implique donc de construire un

modèle pour chaque contexte. Par exemple dans le cas où l'on veut recommander des films à des utilisateurs en fonction de leur âge. Chaque sous-ensemble sera l'ensemble des utilisateurs appartenant à une certaine tranche d'âge. On construira un modèle pour chaque tranche d'âge. Lors de l'évaluation du modèle pour un utilisateur donné, on prédit les préférences de l'utilisateur à partir du modèle correspondant au contexte dans lequel celui-ci se place. Ainsi pour prédire quels films sont susceptibles d'intéresser un utilisateur donné, on évaluera pour cet utilisateur le modèle correspondant à cette tranche d'âge.

Dans le cas où le nombre de contexte est très élevé, il peut s'avérer fastidieux de créer autant de modèles que de contextes. De plus, dans certains cas, cela nécessite la discrétisation de variables continues (comme l'extraction de tranches d'âge). La difficulté est de choisir correctement le bon nombre de classes et les bornes de chacune des classes. Ceci revient à ajouter au problème de recommandation, un problème de classification dont dépendent grandement les performances du système. Ce type d'obstacle éventuel est notamment pointé du doigt par [Rendle 2010] .

3.2.2 La contextualisation par post-traitement

Dans une approche de contextualisation par post-traitement (*contextual post-filtering*), le système de recommandation ne prend pas en compte les données contextuelles lors de l'apprentissage du modèle. Ce n'est qu'après l'évaluation du modèle pour un utilisateur donné que le contexte entre en jeu. Il peut intervenir de deux façons différentes. À partir de la liste des recommandations sortant de l'évaluation, le système peut filtrer en retirant de la liste un certain nombre de produits recommandés selon le contexte, ou le système peut pondérer chacun des produits de la liste suivant le contexte et réordonner la liste des produits en fonction des nouveaux scores obtenus. Notons que cette approche est généralement utilisée dans des problèmes d'ordonnancement.

Un exemple simple et régulièrement utilisé de ce type de procédure consiste à retirer de la liste de recommandation l'ensemble des produits déjà achetés par l'utilisateur. Les informations contextuelles utilisées sont les données de transactions elles mêmes et l'on fait la supposition que l'utilisateur ne sera pas intéressé par un produit qu'il a déjà acheté. Cette hypothèse est valide dans la plupart des cas. De plus, même si ce produit peut à nouveau intéresser l'utilisateur, il connaît son existence (puisque l'a acheté) et il devient inutile de le recommander. Un autre exemple de recommandation dans un paradigme de contextualisation par post-traitement pourrait être, lorsque l'on cherche à recommander des lieux (bars, restaurants, musées...), de ne suggérer parmi les lieux sélectionnés, que les lieux géographiquement proches de la position actuelle de l'utilisateur.

Cette approche ne souffre pas de la faiblesse de l'approche prétraitement étant donné qu'il n'y a qu'un modèle à construire. La difficulté de cette approche réside dans le choix (arbitraire) ou dans l'apprentissage des coefficients de pondération.

3.2.3 La modélisation contextuelle

L'approche de modélisation contextuelle (*contextual modeling*) consiste à intégrer directement dans l'apprentissage du modèle les informations contextuelles.

Pour prendre en compte les informations contextuelles, [Karatzoglou *et al.* 2010] ou [Wermser *et al.* 2011] proposent des méthodes de factorisation tensorielle. Dans ce tenseur, en plus des deux premières dimensions traditionnellement utilisées pour les utilisateurs et pour les produits, chaque nouvelle dimension correspond à un type de contexte et la taille de ces nouvelles dimensions correspond au nombre de contextes pour chaque type de contexte.

Pour un ensemble de données contextuelles variées (de nature quelconque), [Rendle 2010] présente un ensemble de méthodes de factorisation visant à tirer avantage du contexte selon cette approche de modélisation contextuelle.

Dans le cadre de la recommandation dans un contexte social, c'est généralement la modélisation contextuelle qui est utilisée et il est courant d'observer une modification de la fonction objectif à minimiser afin de prendre en compte les informations contextuelles sociales (Ma *et al.* 2008, Yang *et al.* 2011, Ma *et al.* 2011).

L'intérêt de cette approche est que la contextualisation fait partie intégrante du problème d'optimisation et ne génère pas de nouveaux problèmes en amont (classification des données en contextualisation par prétraitement par exemple) ou en aval (réordonnancement en contextualisation par post-traitement par exemple) du problème principal, comme c'est le cas dans les deux autres approches.

3.3 Un cas d'étude précis : la vente d'outils

Les travaux que nous allons présenter dans ce paragraphe proposent une solution de contextualisation pour un problème de recommandation précis. Dans un premier temps, nous présenterons le problème tel qu'il nous a été posé ainsi que les données dont nous disposons. Nous expliciterons ensuite les solutions de contextualisation imaginées pour résoudre le problème. Enfin nous analyserons les résultats obtenus.

Ces travaux ont été réalisés dans le cadre du projet CADI soutenu par l'ANR et ont fait l'objet d'une publication dans la conférence ACM SIGKDD 2011 (Pradel *et al.* 2011). Dans ces travaux, ma contribution concerne principalement l'application des méthodes de factorisation aux contextualisations proposées. L'article associé est disponible en annexe A.

3.3.1 Présentation du problème

Le problème de recommandation auquel nous nous intéressons est issu d'un besoin exprimé par la société La « Boîte à Outils » (BAO)³. Cette société est spécialisée dans la vente d'outillage et possède plusieurs magasins dans le sud-est de la France. Il a été remarqué que souvent, ses clients quittaient le magasin après leurs achats en ayant oublié certains produits dont ils avaient besoin pour leurs travaux, les amenant à faire un aller-retour supplémentaire (ou à aller chercher le produit chez la concurrence). BAO souhaiterait pouvoir recommander à ses clients les produits qu'ils auraient éventuellement oubliés de mettre dans leur caddie, après une identification du client et du contenu du caddie. L'objectif final était que, parmi 5 articles recommandés, au moins un (en moyenne) intéresse vraiment le client.

Pour mettre en place notre moteur de recommandation, nous disposons des données de transactions datées entre 2005 et 2008 et d'informations sur les articles proposés. Il s'agit donc ici de données issues de magasins réels (en opposition aux données issues de la vente en ligne comme on en trouve généralement lorsqu'il s'agit de recommandation).

3.3.2 Présentation des approches contextuelles

Le problème consiste à prendre en compte l'historique d'achat du client afin de lui recommander l'outillage le plus adapté à ses besoins. Malheureusement, nous nous sommes vite rendus compte

3. www.la-bao.fr/

qu'une approche classique qui vise à s'intéresser à l'ensemble de l'historique d'achat de l'utilisateur ne permettait pas d'obtenir des performances correctes. Ceci est du notamment à la nature des produits vendus par BAO. En effet, BAO vend des produits qui reflètent un besoin immédiat (et généralement temporaire) tandis que la plupart des sites de vente en ligne propose des produits reflétant les goûts de l'utilisateur (qui peuvent certes évoluer au cours du temps mais qui ont une plus grande pérenité). Un client qui va vouloir rénover sa salle de bain n'aura besoin du matériel nécessaire à de tels travaux qu'une seule fois. Une fois la rénovation terminée ce client n'en aura plus besoin, de plus les besoins temporaires relatifs à cette rénovation ne devrait pas avoir d'influence sur des besoins futurs (comme l'aménagement de son jardin par exemple). L'importance de l'utilisation d'une contextualisation temporelle semble évidente.

Pour ces raisons, nous proposons dans ces travaux deux approches contextuelles afin de prendre en compte la nature particulière des données dont nous disposons. La première est une approche temporelle basée sur l'historique d'achat des utilisateurs. Nous appellerons la seconde approche, approche par chantier. Cette approche est basée sur une connaissance experte a priori sur les produits et sur une temporalité des achats également.

3.3.2.1 Un contexte temporel

Pour cette approche nous nous plaçons dans un paradigme de contextualisation par prétraitement. Nous souhaitons mettre en évidence la volatilité des informations de transaction. Pour cela, l'idée générale est de ne prendre en compte, pour un jour donné, que les transactions des deux semaines précédentes pour prédire les transactions des deux semaines suivantes. On suppose ici que les besoins immédiats d'un client sont visibles sur une fenêtre temporelle restreinte de deux semaines. Cette approche de contextualisation est en fait un premier pas vers la seconde approche que nous proposons : l'approche par chantier, que nous allons maintenant présenter.

3.3.2.2 Un contexte de chantier

Pour cette approche également, nous nous plaçons dans un paradigme de contextualisation par prétraitement. Nous introduisons ici la notion de chantier. Un chantier est un projet de bricolage (construction, réparation, rénovation...) mené par le client. Ces chantiers peuvent aller de la réparation d'une salle de bain, de l'installation de l'électricité à la rénovation de la toiture. Ces chantiers nécessitent l'achat d'articles spécifiques en une quantité élevée sur une courte durée. Nous identifions comme chantier tout ensemble d'achats dépassant 200 euros sur une durée inférieure à deux semaines.

Grâce à une classification de type K-moyennes, 11 types de chantiers ont été identifiés. Ces chantiers ont été étiquetés par un expert du domaine. Ils sont présentés dans la table 2 de l'article page 96.

Dans ce cadre, la contextualisation par prétraitement consiste, lors de l'apprentissage, à construire autant de modèles que de contextes proposés, c'est-à-dire 11 modèles.

3.3.3 Protocole expérimental

De ces deux contextualisations, nous avons élaboré un ensemble de protocoles expérimentaux (que nous allons décrire dans cette section) afin de valider leur intérêt. Dans tout les cas, l'ensemble des transactions des années entre 2005 et 2007 sert à l'apprentissage et les transactions de l'année

2008 aux tests. En terme de volumétrie, cela correspond à environ 50000 clients pour plus de 3 millions de transactions entre 2005 et 2007 et à environ 30000 clients pour plus d'un million de transaction sur l'année 2008. De plus, ces protocoles ont été testés sur un ensemble de quatre méthodes : une approche par calcul de règles d'associations, une approche par calcul de similarité en article, une factorisation par décomposition en valeurs singulières et une factorisation non négative. Nous présentons les méthodes de factorisation dans le paragraphe 3.3.3.1, les autres méthodes sont décrites dans la section 4 de l'article.

3.3.3.1 Méthodes de factorisation choisies

Nous nous intéressons dans ce paragraphe aux méthodes de factorisation choisies pour résoudre notre problème, à savoir la décomposition en valeurs singulières et la factorisation non-négative.

Pour la décomposition en valeurs singulières, le problème d'optimisation est le suivant :

$$\begin{aligned} \min_{U,V} \|Y - UV\|_{Fro}^2, \\ \text{s.t. } \text{rang}(U) = \text{rang}(V) = k, \end{aligned} \quad (3.1)$$

et pour la factorisation non négative, nous posons le problème suivant :

$$\begin{aligned} \min_{U,V} \|Y - UV\|_{Fro}^2, \\ \text{s.t. } \text{rang}(U) = \text{rang}(V) = k, \\ U \in \mathbb{R}_+^{n \times k}, \\ V \in \mathbb{R}_+^{k \times p}, \end{aligned} \quad (3.2)$$

avec $Y \in \mathbb{R}^{n \times p}$ les données observées. Les méthodes de résolution de ces problèmes sont décrites dans le paragraphe 2.2.

Quelque soit le protocole expérimental choisi, les phases d'apprentissage et de test se déroulent de la même façon pour les deux méthodes de factorisation. L'ensemble d'apprentissage sert à établir les profils produits, on apprend les profils utilisateurs avec l'ensemble de test et le score d'appétence d'un utilisateur pour un produit est calculé par le produit scalaire entre le profil utilisateur appris au cours de la phase de test et le profil produit appris au cours de la phase d'apprentissage comme le montre l'algorithme 5.

Entrées : $Y_{\text{app}}, Y_{\text{test}}$

Sortie : $\hat{Y}$

$U_a, V_a \leftarrow \underset{U,V}{\text{argmin}} J(Y, UV)$ (Apprentissage)

$U_t \leftarrow \underset{U}{\text{argmin}} J(Y, UV_a)$ (Test)

$\hat{Y} \leftarrow U_t V_a$ (Prédiction)

Algorithme 5 – Protocole expérimental pour la factorisation

3.3.3.2 Évaluation

Pour chaque utilisateur, nous obtenons une liste de score d'appétence pour des produits. Les méthodes d'évaluation les plus adaptées dans ce cadre sont les méthodes d'ordonnement où l'on va trier les éléments de la liste et déterminer les performances de prédiction en utilisant des métriques adaptées. Nous avons choisi d'utiliser les métriques de précision, rappel et F-measure décrites dans le paragraphe 2.3.3. Pour ces métriques nous devons choisir le nombre de recommandation à présenter. BAO souhaite recommander à ses clients 5 produits, nous calculerons donc les précisions à 5, rappels à 5 et F-measure à 5.

3.3.3.3 Protocole de contextualisation temporelle

Pour valider l'intérêt de la contextualisation temporelle, nous décidons de comparer deux protocoles d'évaluation de modèles. Dans le premier nous prédisons, pour une date donnée, les achats futurs d'un utilisateur à partir de l'ensemble des achats précédemment effectués dans le courant de l'année 2008. Dans le second protocole nous prédisons, pour une date donnée, les achats des deux prochaines semaines à partir des achats effectués dans les deux semaines précédentes. Dans les deux cas nous apprenons notre modèle sur l'ensemble des transactions des années 2005 à 2007.

Les résultats expérimentaux de ce protocole sont présentés dans la table 1 de l'article page 95. Le fait le plus marquant de ces résultats est la différence d'influence de l'introduction d'un contexte temporel suivant le type de méthode choisi. En effet, le contexte semble avoir très peu, voire pas du tout, d'influence sur les performances des méthodes de factorisation tandis que les deux méthodes orientées produit y sont sensibles. Bien que les méthodes de factorisation semblent montrer une certaine stabilité face aux données (face au bruit de l'historique complet) dans un tel contexte, force est de constater que l'introduction d'une contextualisation temporelle accroît la qualité d'un système de recommandation, comme les travaux récents semblent le montrer.

Ce premier protocole expérimental est détaillé dans la section 5 de l'article page 95.

3.3.3.4 Protocole de contextualisation par chantier préliminaire

Nous construisons une nouvelle matrice d'apprentissage, dans laquelle chaque ligne est un chantier identifié dans l'ensemble d'apprentissage et où les colonnes représentent les articles. Nous apprenons notre modèle sur cette matrice chantier×article. Dans ce protocole, nous supposons ne pas connaître le chantier en cours des clients, ainsi pour la validation nous appliquons le modèle de la même façon que pour le protocole expérimental temporel. C'est-à-dire que, pour prédire les achats futurs (dans leur globalité ou sur deux semaines) d'un utilisateur, nous évaluons notre modèle sur l'historique passé (complet ou de deux semaines) de cet utilisateur.

Les résultats expérimentaux de ce protocole sont présentés dans les tables 3 et 4 de l'article page 96 et 97. Ces résultats présentent de grandes similarités avec ceux détaillés dans la section précédente. Le fait de cumuler le contexte temporel et contexte par chantier semble permettre une légère hausse des performances (principalement en terme de rappel) des modèles (excepté pour la SVD qui reste très peu sensible à cette contextualisation). Cependant on remarque que lorsque le contexte temporel n'est pas utilisé, les performances d'une contextualisation par chantier sont moins bonnes (et ce pour l'ensemble des méthodes) que lorsqu'aucun contexte n'est pris en compte. Ainsi, nous touchons du doigt le fait que toute approche contextuelle n'est pas nécessairement bonne et que la vraie difficulté du problème de contextualisation n'est pas nécessairement d'appliquer un contexte donné à un problème, mais plutôt de choisir le bon contexte discriminant pour un problème donné.

Ce deuxième protocole expérimental est détaillé dans la section 6.2 de l'article page 96.

3.3.3.5 Protocole de contextualisation par chantier évolué

Le dernier protocole expérimental concerne uniquement la contextualisation par chantier mais de façon plus poussée. Ce protocole correspond à la section 6.3 de l'article page 96.

Nous allons à nouveau comparer deux approches. La première approche ressemble assez à celle choisie dans le second protocole expérimental. La matrice d'apprentissage chantier×article est la même. La différence réside dans le choix des données d'évaluation du modèle. Au lieu d'effectuer la prédiction sur les achats des utilisateurs dans les deux semaines futures (à partir des achats des deux semaines précédentes), nous allons prédire les achats futurs d'un chantier donné à partir des achats déjà effectués sur ce chantier. La seconde approche consiste à construire une matrice par type de chantier différent pour l'ensemble d'apprentissage, ainsi nous apprendrons 11 modèles différents. Pour l'évaluation du modèle, nous appliquerons le modèle correspondant au chantier en cours pour lequel nous souhaitons effectuer la prédiction. Ce protocole expérimental nécessite la connaissance a priori du chantier en cours, cette information n'est pas compliqué à obtenir, il faut que le vendeur demande au client le chantier sur lequel il est en train de travailler.

Les résultats expérimentaux de ce protocole sont présentés dans la table 5 de l'article page 97. Les performances des deux approches parlent d'elles mêmes et plus particulièrement pour l'approche impliquant la création de 11 modèles différents. En effet les performances de l'ensemble des méthodes dépassent les 20% pour la précision à 5. De plus on remarque que les méthodes de factorisation, jusqu'alors peu sensibles au contexte, voient leurs performances réellement améliorées. Les performances de cette approche finale rendent acceptable l'utilisation d'un tel système de recommandation par BAO. Cependant, il est important de noter que seuls les clients en chantier sont concernés par cette contextualisation. En effet, seulement un tiers des transactions est représenté ici et la plupart des clients (qui ne sont pas en chantier) ne bénéficieront pas nécessairement d'une recommandation de même qualité que les clients en chantier.

3.4 Discussion

Les résultats obtenus sur les expérimentations impliquant le contexte temporel nous montrent que les méthodes de factorisation (*SVD* et *NMF*) semblent peu sensibles au bruit provoqué par les achats isolés sur l'ensemble de l'historique des clients. Pour ces approches et pour ces données l'utilisation d'un tel contexte ne semble pas justifiée. Cette contextualisation se justifie beaucoup plus pour les autres approches proposées à savoir les approches orientées produit et les règles d'associations.

La variation de performance entre le premier protocole expérimental et le second nous indique également qu'une contextualisation partielle par chantier n'est pas pertinente (en terme de performance) pour l'ensemble des méthodes (comme le montrent les résultats présentés dans les tables 1 et 3). L'utilisation de ce protocole pourrait se justifier dans la résolution de problèmes de recommandation d'articles peu souvent achetés (« longue traîne ») comme les résultats présentés dans la table 4 semblent le montrer.

Seule la contextualisation par chantier complète devient intéressante en terme de performance. En effet, la précision à 5 dépasse 20% pour l'ensemble des méthodes proposées, ce qui signifie que sur les 5 articles recommandés, entre 1 et 2 (en moyenne) intéressent effectivement le client.

Suite à ces résultats nous pouvons souligner plusieurs points. Tout d'abord la contextualisation permet d'améliorer les performances des moteurs de recommandation, comme le montrent la plupart des travaux récents dans le domaine. Ensuite, toute contextualisation n'est pas nécessairement intéressante, voire peut brider les performances des moteurs de recommandation. Cette tâche de contextualisation n'est donc pas aisée. Enfin l'ensemble des contextualisations proposées ici s'inscrit dans le même paradigme qui est celui de la contextualisation par prétraitement. Les différents travaux récents semblent montrer que la mise en place de contextes dans un paradigme de modélisation contextuelle donne généralement de meilleurs résultats qu'avec les deux autres types d'approche (principalement pour les méthodes de factorisation). Une étude plus poussée dans ce sens là pourrait s'avérer intéressante.

Recommandation dans les réseaux sociaux

Le choix des amis peut orienter ou désorienter toute une vie.

– Daniel Desbiens

Sommaire

4.1	Les Concepts	46
4.2	État de l'art sur la recommandation dans les réseaux sociaux	47
4.2.1	Notations	48
4.2.2	Ajout de contraintes au problème	48
4.2.3	Modification de la fonction de décision	50
4.3	D'une pierre deux coups : recommandation et apprentissage d'influence	50
4.3.1	L'influence des amis	50
4.3.2	Optimisation	51
4.3.3	Expérimentations	55
4.4	Discussion	63

Dans ce chapitre, nous présenterons le contexte social tel qu'il est généralement vu dans le domaine de la recommandation. Nous ferons ensuite un état de l'art des méthodes de recommandation de préférences dans les réseaux sociaux, en nous attardant particulièrement sur les méthodes de factorisation. Enfin, nous présenterons une approche qui permet d'apprendre simultanément les préférences des utilisateurs et le graphe de confiance associé au réseau social en tirant avantage des données d'appréciations et du graphe social, tout en gérant le volume considérable des données, ce qui peut être une difficulté dans ce type d'application. Ces travaux ont fait l'objet d'une publication dans la conférence ECML PKDD 2013 qui a reçu le prix du meilleur article étudiant (Delporte *et al.* 2013).

4.1 Les Concepts

Dans le cadre de la recommandation dans les réseaux sociaux, nous admettons de façon quasi-systématique un important principe sociologique. Ce principe, évoqué pour la première fois par [Lazarsfeld & Merton 1954], est celui d'Homophilie que l'on peut définir de la façon suivante :

Définition 4.1. *L'Homophilie est la tendance qu'ont les individus à se rapprocher des personnes avec qui ils ont des points communs.*

On peut dire qu'il y a une certaine corrélation entre le fait d'être connecté socialement, et celui d'avoir des affinités¹. Ces affinités peuvent être de différentes natures. Il peut s'agir du statut social ou professionnel, des idées politiques ou religieuses, ou simplement des goûts personnels. Dans notre cadre d'étude, nous nous intéresserons principalement aux goûts. Dans les réseaux sociaux, il est devenu courant d'admettre l'hypothèse suivante comme étant vraie : deux personnes amies dans un réseau social auront tendance à apprécier les mêmes choses et à avoir les mêmes centres d'intérêts. Ainsi il sera plus probable qu'un utilisateur apprécie un article si un ou plusieurs de ses amis ont déjà explicitement apprécié cet article.

Le second concept important dans les réseaux sociaux est celui de confiance. Plusieurs sens peuvent être accordés à ce terme, mais nous ne nous intéresserons qu'à celui généralement utilisé dans les réseaux sociaux à savoir la confiance dite orientée réputation (*reputation-based trust*). Voici donc la définition de ce que nous entendons par confiance :

Définition 4.2. *On dit de l'individu A qu'il a confiance en un individu B s'il est enclin à laisser B le guider dans ses actions (on peut également dire que B exerce une influence sur A).*

Cette confiance est représentée traditionnellement au travers d'un graphe où les nœuds sont des individus et les arcs dénotent la confiance entre deux individus. Cette confiance est généralement quantifiée et il y a ici plusieurs écoles : la première idée consiste à représenter la confiance de façon binaire en choisissant les valeurs sur l'ensemble $\{0, 1\}$ (Golbeck & Hendler 2006). Cette confiance peut également être représentée de façon discrète en choisissant les valeurs sur un ensemble $\{0, \dots, n\} \subset \mathbb{N}$ (Golbeck et al. 2003). Enfin cette représentation de la confiance peut être continue (sur l'intervalle $[0, 1]$ par exemple), où le niveau de confiance peut varier d'un couple d'individus à un autre (Richardson et al. 2003). Pour chacune de ces possibilités, 0 dénotera systématiquement l'absence de confiance d'un individu A pour un individu B (la non pertinence des goûts de B pour ceux de A). Généralement, le zéro est représenté dans le graphe par l'absence d'arc entre nœuds.

À cette relation de confiance est associé un certain nombre de propriétés (énumérées dans [Golbeck & Hendler 2006]). Pour la suite, il paraît important de citer certaines de ces propriétés :

- Une relation de confiance est *non symétrique*. En effet, le fait qu'un individu A fasse confiance à un individu B ne signifie pas que B fasse forcément confiance à A. De même que le niveau de confiance de A pour B n'est pas nécessairement le même que celui de B pour A (pour le cas discret multivalué ou le cas continu).
- Une relation de confiance est *personnelle*. A ne fera pas forcément confiance à C de la même façon que B à C. De plus A ne fera pas confiance de la même façon à B qu'à C.

À cette relation de confiance est parfois associée ce que l'on pourrait appeler la méfiance (*distrust*). Nous entendons par méfiance l'opposé de la confiance, à savoir la tendance d'un individu à avoir des goûts opposés à ceux d'un autre individu et ne dénotent surtout pas l'absence de

1. Affinité : Rapport de conformité, de ressemblance ; harmonie de goûts, de sentiments (entre personnes). Dictionnaire culturel de la langue française, le Robert, 2005.

confiance (qui elle est représentée par la valeur zéro et par l'absence d'arc entre nœuds dans le graphe de confiance). Cette méfiance est encodée de différentes façons dans la littérature. Par exemple dans [DuBois *et al.* 2011], les valeurs que peuvent prendre la confiance appartiennent à l'ensemble $\{-1, 0, 1\}$, où 1 dénote la confiance, -1 la méfiance et 0 l'absence de confiance, ainsi la méfiance est représentée sur le même graphe que la confiance dans ce cas. Dans [Guha *et al.* 2004], un second graphe est construit pour encoder la méfiance. Ainsi il y a deux graphes, un pour la confiance et un pour la méfiance. Ce qui est appelé *belief*, qui serait la confiance globale, est une combinaison de ces deux graphes. Les valeurs des arcs de ce graphe appartiennent à l'intervalle $[-1, 1]$ dans [Guha *et al.* 2004]. La figure 4.1 illustre un exemple de ce type de graphe.

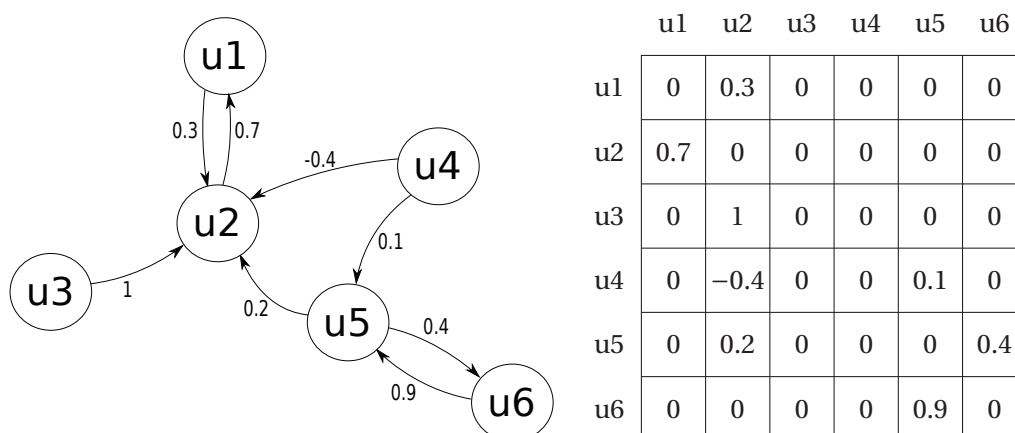


FIGURE 4.1 – Exemple d'un graphe de confiance où les arcs prennent leurs valeurs dans $[-1, 1]$ et sa matrice associée. On peut noter que u_3 a confiance en u_2 mais que l'avis de u_3 n'a pas d'influence sur celui de u_2 , tandis que u_1 et u_2 se font mutuellement confiance mais à des degrés différents. On peut voir également que u_4 se méfie de u_2 .

[Artz & Gil 2007] dresse un état de l'art exhaustif du sujet. Nous recommandons particulièrement la lecture de la section 4 qui concerne la confiance dans la recommandation sociale et notamment la création de graphe de confiance dans un tel environnement.

4.2 État de l'art sur la recommandation dans les réseaux sociaux

Depuis l'arrivée des réseaux sociaux en ligne (OSN) et la masse gigantesque de données qui en résulte, les systèmes de recommandation sont devenus un outil essentiel pour guider les utilisateurs dans leur navigation à travers le contenu de ces réseaux sociaux. La recommandation dans les réseaux sociaux se concentre sur trois aspects principalement :

- la recommandation de centres d'intérêt (film, musique, restaurant...),
- la recommandation de publicités ciblées (ce qui représente dans la plupart des cas le modèle économique du réseau social),
- la recommandation d'amis.

On considérera que la recommandation de centres d'intérêt et de publicités ciblées sont des problèmes similaires, voire équivalents, car il est paraît évident que le produit proposé dans une publicité correctement recommandée est un produit associé à un des centres d'intérêt de l'utilisateur pour qui la recommandation a été effectuée.

Bien qu'une partie non négligeable de la recommandation dans les réseaux sociaux s'intéresse à la prédiction de lien d'amitié, nous nous concentrerons principalement sur la recommandation d'articles dans les réseaux sociaux. Nous nous attarderons tout particulièrement sur le problème de l'amélioration de cette recommandation en se servant du graphe social.

4.2.1 Notations

Avant de rentrer dans les détails, posons quelques notations. Soient $Y \in \{0, 1\}^{n \times p}$ une matrice de préférences, et $\mathcal{Y}$ l'ensemble des couples (i, j) pour lesquels on sait si l'utilisateur i apprécie l'article j . Notons qu'un 0 dans Y ne dénote pas forcément le fait que l'article n'est pas apprécié, mais peut vouloir simplement dire que l'information n'est pas connue, mais cependant si $Y_{ij} = 0$ avec $(i, j) \in \mathcal{Y}$ alors cela signifiera que l'utilisateur i n'aime pas l'article j . Soit également $\mathcal{F}$ un graphe social avec n nœuds, où le nœud i est relié au nœud j si les utilisateurs i et j sont amis. On dira alors de façon formelle que $i \in \mathcal{F}_j$ et $j \in \mathcal{F}_i$, où $\mathcal{F}_i$ représente l'ensemble des amis de i . Définissons également la matrice S associée à ce graphe social définie par $S_{ij} = 1$ si $i \in \mathcal{F}_j$, 0 sinon.

Ainsi, en plus des données d'appréciation de la matrice Y et des données contextuelles (propres aux utilisateurs ou aux articles) dont nous disposons traditionnellement, nous avons le graphe social du réseau qui est une source d'information des plus intéressantes pour améliorer la recommandation d'article. Nous allons voir maintenant les différentes approches possibles pour utiliser cette information afin d'améliorer nos recommandations.

4.2.2 Ajout de contraintes au problème

De nombreux travaux récents sur les réseaux sociaux en ligne se servent des données dans le graphe social en ajoutant de nouvelles contraintes au problème. Ceci se traduit lors de l'écriture de la forme lagrangienne du problème par l'ajout d'un terme dans l'expression de la fonction objectif. Ces approches, visant à introduire une contextualisation sociale, s'incrivent toutes dans la classe d'approches définie dans le chapitre 3 par le terme modélisation contextuelle.

C'est le cas notamment de [Yang et al. 2011] qui ajoute un terme assimilable à un terme d'attache aux données, que l'on pourrait appeler le critère d'attache au graphe social.

$$\min_{f, h} \sum_{(i, j) \in \mathcal{Y}} l_1(Y_{ij}, f_{ij}) + \lambda_S \sum_i \sum_{k \in \mathcal{F}_i} l_2(S_{ik}, h_{ik}) + \Omega(f, h),$$

où l_1 et l_2 sont des coûts et Ω un ensemble de pénalités sur le modèle permettant entre autres sa régularisation. Les fonctions de décision f et h dans le modèle sont définies dans l'article comme ceci : $f_{ij} = U_i V_j + x_i^\top W y_j$ et $h_{i,i'} = U_i U_{i'}^\top + x_i^\top M x_{i'}$, où x_i et $x_{i'}$ sont des variables observables sur les utilisateurs i et i' , et où y_j un ensemble de variables observables sur l'article j , c'est-à-dire des données contextuelles dans les deux cas. L'optimisation se fait selon les paramètres U, V, W et M . Afin de mieux voir ce qui se passe au niveau de l'intégration des données sociales, plaçons nous dans un cadre sans information contextuelle (c'est-à-dire $x_i = 0 \forall i$ et $y_j = 0 \forall j$). Le problème peut se formuler simplement comme dans l'équation 4.1.

$$\min_{U, V} \sum_{(i, j) \in \mathcal{Y}} l_1(Y_{ij}, U_i V_j) + \lambda_S \sum_i \sum_{k \in \mathcal{F}_i} l_2(S_{ik}, U_i U_k^\top) + \lambda_\Omega \Omega(U, V). \quad (4.1)$$

On voit très clairement que les facteurs latents U doivent permettre l'attache du modèle à la fois aux données de préférences et aux données sociales. De plus ce modèle comporte deux fonctions de décisions distinctes, l'une (f) permettant la prédiction de préférences comme sur un modèle classique,

et l'autre (h) permettant de prédire d'éventuels liens d'amitié entre utilisateurs. Les auteurs testent leur modèle avec une liste de fonctions l différentes, à savoir un coût quadratique, un coût quadratique "fainéant" ($l(y, f) = \min(1, \max(0, 1 - yf)^2)$), un coût logistique, un coût de Huber défini par l'équation 4.2 ou encore un coût Ψ défini par l'équation 4.3.

$$\eta(y, f) = \begin{cases} \frac{1}{2} \max(0, 1 - yf)^2 & \text{si } yf > 0, \\ \frac{1}{2} - yf & \text{sinon,} \end{cases} \quad (4.2)$$

$$\Psi(y, f) = \begin{cases} \frac{1}{2} \max(0, 1 - yf)^2 & \text{si } yf > 0, \\ 1 - \frac{1}{2} \max(0, 1 + yf)^2 & \text{sinon.} \end{cases} \quad (4.3)$$

Les auteurs proposent de résoudre ce problème d'optimisation grâce à un algorithme de descente de gradient stochastique. Notons enfin, en remarque, que les auteurs ont fait le choix d'utiliser la même fonction de perte pour le critère d'attache aux données de préférences que pour le critère d'attache aux données sociales ($l_1 = l_2$). Nous pourrions cependant imaginer que cette attache se fasse différemment pour des types de données différentes et que donc la fonction de perte soit différente pour chaque critère.

Cette idée de changer la formulation du problème en ajoutant un terme à la fonction objectif, comme le fait [Yang et al. 2011], a été proposée plus tôt dans [Ma et al. 2008] où les auteurs utilisaient un coût quadratique pour l_1 et l_2 . Les travaux préliminaires de [Ma et al. 2008] ont permis à [Yang et al. 2011] de proposer une version améliorée du modèle : comme dit plus haut, celui-ci s'intéresse maintenant à la recommandation d'amis, et le modèle a été testé sur un panel complet de fonctions coût.

Cet ajout d'une contrainte au problème d'optimisation se retrouve également dans [Ma et al. 2011]. Les auteurs se sont intéressés à plusieurs façons de modéliser l'homophilie et proposent de forcer les variables latentes des utilisateurs à ressembler aux variables latentes de leurs amis. Ils se sont intéressés à trois problèmes d'optimisation allant dans ce sens. Le premier problème proposé vise à faire ressembler les variables latentes d'un utilisateur donné à la moyenne des variables latentes de ses amis :

$$\min_{U, V} \sum_{(i, j) \in \mathcal{Y}} l(Y_{ij}, U_i V_j) + \lambda_S \sum_{i=1}^n \|U_i - \frac{1}{|\mathcal{F}_i|} \sum_{k \in \mathcal{F}_i} U_k\|_F^2 + \lambda_\Omega \Omega(U, V). \quad (4.4)$$

Un deuxième problème vise à affiner le premier, en remplaçant la moyenne des variables latentes des amis par une somme pondérée de ces variables latentes :

$$\min_{U, V} \sum_{(i, j) \in \mathcal{Y}} l(Y_{ij}, U_i V_j) + \lambda_S \sum_{i=1}^n \|U_i - \frac{\sum_{k \in \mathcal{F}_i} \text{sim}(i, k) U_k}{\sum_{k \in \mathcal{F}_i} \text{sim}(i, k)}\|_F^2 + \lambda_\Omega \Omega(U, V), \quad (4.5)$$

où $\text{sim}(i, k)$ est une fonction de similarité entre l'utilisateur i et l'utilisateur k . Plusieurs choix pour cette fonction sont proposés dans [Ma et al. 2011] et cette similarité est calculée dans chaque cas à partir des informations présentes dans la matrice de préférences et non pas à partir de celles du graphe de confiance. La dernière formulation du problème est proche de celle de l'équation 4.5 et vise à pénaliser une distance entre les variables latentes des personnes amies. Cette distance est pondérée par la mesure de similarité $\text{sim}(i, k)$.

$$\min_{U, V} \sum_{(i, j) \in \mathcal{Y}} l(Y_{ij}, U_i V_j) + \lambda_S \sum_{i=1}^n \sum_{k \in \mathcal{F}_i} \text{sim}(i, k) \|U_i - U_k\|_F^2 + \lambda_\Omega \Omega(U, V). \quad (4.6)$$

En résolvant les problèmes 4.5 et 4.6, les variables latentes d'utilisateurs qui sont amis auront tendance à avoir des valeurs proches. Pour chacune de ces trois fonctions objectif, la fonction l est une

erreur quadratique. Le processus d'optimisation se fait comme dans [Yang *et al.* 2011] à travers un algorithme de descente de gradient stochastique.

On retrouve une approche très similaire à celle de l'équation 4.5 dans [Jamali & Ester 2010], à ceci près qu'ici le terme $sim(i, k) / \sum_{k' \in \mathcal{F}_i} sim(i, k')$ est remplacé par les valeurs S_{ik} du graphe de confiance qui ont été normalisées par ligne. Dans ces travaux aussi un algorithme de descente de gradient est utilisé.

4.2.3 Modification de la fonction de décision

Plutôt que d'ajouter une contrainte (ou plusieurs) au problème pour prendre en compte les informations sociales, certains choisissent de modifier la fonction de décision. Ainsi apparaîtra dans cette fonction un terme qui permettra la prise en compte du réseau social.

Un exemple de cette utilisation des données sociales est présenté dans [Ma *et al.* 2009]. Les auteurs modifient la fonction de décision (usuellement $F_{ij} = U_i V_j$) de la façon suivante :

$$F_{ij} = g \left(\mu U_i V_j + (1 - \mu) \sum_{k \in \mathcal{F}_i} S_{ik} U_k V_j \right), \quad (4.7)$$

où g est la fonction logistique (ce qui permet d'avoir $\forall i, j \quad F_{ij} \in [0, 1]$) et μ un paramètre d'équilibre entre l'influence des propres goûts d'un utilisateur et l'influence des goûts de ses amis sur la prédiction. Ce paramètre μ est fixé par validation croisée. La fonction objectif reste identique à celle d'un problème sans interactions sociales puisque celles-ci sont directement intégrées dans F :

$$\min_{U, V} \sum_{i, j} W_{ij} l(F_{ij}, Y_{ij}) + \lambda_{\Omega} \Omega(U, V), \quad (4.8)$$

où W est une matrice de pondération définie comme dans l'équation 2.5 et l un coût quadratique ici encore. Le processus d'optimisation est réalisé à travers un algorithme de descente de gradient. Ces travaux ont été importants pour la contribution proposée dans le paragraphe 4.3, nous reviendrons donc sur les avantages et inconvénients de cette méthode.

4.3 D'une pierre deux coups : recommandation et apprentissage d'influence

Les méthodes de factorisation pour la recommandation de produits dans la littérature partent toutes du postulat (parfois non explicite) que le graphe social à disposition est un graphe de confiance qui reflète une réelle influence entre les préférences des utilisateurs. Dans le cas où cette hypothèse n'est pas vraie, les performances du système risquent de drastiquement chuter.

Nos motivations ici sont, à partir de données d'appréciations implicites, de construire un modèle qui permet la recommandation d'articles et qui, quelque soit la nature du graphe social, s'adapte automatiquement en tirant pleinement avantage de celui ci lorsqu'il s'agit d'un graphe de confiance et en modérant son impact lorsqu'il ne reflète qu'une connexion sociale faible.

4.3.1 L'influence des amis

Un des défis de ce travail est d'inclure l'influence du graphe social dans le modèle de factorisation matricielle. Dans ce but, comme dans [Ma *et al.* 2009], nous proposons de modifier la fonction de décision usuelle pour inclure l'influence du réseau social. En s'inspirant de la fonction de décision

de l'équation 4.7, avec le désir de ne pas vouloir dépendre aussi fortement de S , nous construisons la fonction de décision suivante :

$$F_{ij} = U_i V_j + \sum_{k \in \mathcal{F}_i} \frac{\alpha_{ik}}{|\mathcal{F}_i|} U_k V_j, \quad (4.9)$$

où α_{ik} est un paramètre poids qui pondère le niveau d'influence d'un ami k sur un utilisateur i .

Nous modélisons ainsi les préférences d'un utilisateur par une combinaison de ses propres préférences et de celles de ses amis. Comme nous nous basons sur le principe d'homophilie dans le réseau social, il est raisonnable de supposer que certaines des préférences latentes d'un utilisateur ne sont pas exprimées dans les données mais peuvent en revanche être exprimées dans les préférences de ses amis. Nous supposons en effet qu'un utilisateur aura tendance à avoir des amis avec des goûts similaires.

De plus, la fonction de décision de l'équation 4.9 exprime le fait que l'utilisateur est « influencé » par son réseau d'amis et le poids signale le niveau d'influence de chaque ami sur l'utilisateur. Nous supposons pour cela que certains amis de l'utilisateur peuvent être plus « influents » que d'autres, en se basant sur la propriété qu'une relation de confiance est *personnelle*. En particulier dans les réseaux sociaux en ligne où les utilisateurs auront tendance à avoir un nombre assez conséquent d'amis, nous nous attendons à ce que ces utilisateurs n'aient des goûts similaires qu'avec seulement une fraction de leurs amis, d'où l'introduction de ce poids α qui va pondérer l'influence des amis.

En outre, il est à noter que cette influence n'est pas nécessairement symétrique étant donné qu'un utilisateur peut être « influencé » par un ami sans pour autant exercer sur lui une influence équivalente (d'après la propriété de *non symétrie*).

À partir de cette fonction de décision et de la fonction objectif de l'équation 2.1, nous posons une nouvelle fonction objectif dépendant des facteurs U et V , ainsi que des poids α_{ik} . Nous définissons la matrice A telle que $A_{ik} = \alpha_{ik}$, $\forall i, \forall k \in \mathcal{F}_i$, 0 ailleurs.

$$\min_{U, V, A} J = \sum_{(i, j) \in \mathcal{Y}} c_{ij} \left(U_i V_j + \sum_{k \in \mathcal{F}_i} \frac{\alpha_{ik} U_k V_j}{|\mathcal{F}_i|} - Y_{ij} \right)^2 + \Omega_{U, V, A}, \quad (4.10)$$

où $\Omega_{U, V, A} = \lambda_1 \|U\|_F^2 + \lambda_2 \|V\|_F^2 + \lambda_3 \|A\|_F^2$ est un terme régularisant et c_{ij} est une constante caractérisant le poids des données.

4.3.2 Optimisation

Bien que l'équation 4.10 ne soit pas convexe selon U , V et A , elle l'est disjointement selon chacun des trois paramètres à condition de fixer les deux autres. Nous proposons d'optimiser la fonction objectif de l'équation 4.10 en utilisant un processus relaxation dans l'idée de celui de Gauss-Seidel : nous fixons alternativement deux des trois paramètres (U , V ou A) et nous mettons à jour le troisième. Lorsque deux des trois paramètres sont fixés, le problème résultant est un problème simple et convexe de minimisation quadratique des moindres carrés et peut être résolu efficacement. Ainsi le processus d'optimisation consiste à mettre à jour de façon efficace alternativement à chaque itération la matrice utilisateur U , la matrice article V et la matrice poids A . Pour obtenir les bonnes règles de mise à jour pour chacun des trois paramètres (U_i , V_j et $\alpha_{ii'}$), nous devons calculer les gradients de la fonction objectif par rapport aux différents paramètres.

Comme nous sommes dans le cas de données d'appréciation implicite (*i.e.* nous ne pouvons décider si $Y_{ij} = 0$ signifie que l'utilisateur i n'aime pas l'article j ou qu'il ne le connaît pas), nous ne pouvons pas donner la même importance aux informations que nous savons vraies (comme les 1 dans la matrice Y), et aux informations dont nous ne connaissons pas le vrai sens (comme les 0 dans

la matrice Y). Nous discuterons plus loin de la meilleure façon de définir c_{ij} afin de pénaliser plus lourdement les données observées. Il est à noter qu'en opposition aux modèles de factorisation pour des données explicites, comme des données de vote (*rating*), où l'apprentissage est réalisé en parcourant uniquement les données observées, nous réalisons notre optimisation sur l'ensemble de la matrice Y , ceci inclut donc les entrées non observées qui peuvent être vues comme une forme faible d'appréciation négative (*negative feedback*).

4.3.2.1 Mise à jour de U

Pour trouver la règle de mise à jour d'un vecteur facteur pour un unique utilisateur i , U_i , nous calculons le gradient de la fonction objectif pour un utilisateur donné par rapport au facteur U_i de cet utilisateur et nous l'annulons. Nous pouvons ensuite analytiquement résoudre cette expression par rapport à U_i en trouvant le vecteur $\hat{U}_i$ tel que $\nabla_{U_i} J_i(\hat{U}_i) = 0$. Pour formuler la règle de mise à jour, il est pratique de l'écrire sous forme matricielle. Pour cela, nous définissons une matrice diagonale $C^i \in \mathbb{R}^{p \times p}$ telle que $C_{jj}^i = c_{ij}$ ($C_{jk}^i = 0$, si $j \neq k$). On peut noter également $C^i = \text{diag}(c_{i\cdot})$.

$$\begin{aligned}
0 &= \nabla_{U_i} J_i(\hat{U}_i), \\
0 &= \left[\nabla_{U_i} \sum_j c_{ij} (U_i V_j + \sum_{k \in \mathcal{F}_i} \frac{\alpha_{ik} U_k V_j}{|\mathcal{F}_i|} - Y_{ij})^2 + \lambda_1 \|U_i\|^2 \right]_{\hat{U}_i}, \\
0 &= 2(\hat{U}_i V + \frac{A_i U V}{|\mathcal{F}_i|} - Y_{i\cdot}) C^i V^\top + 2\lambda_1 \hat{U}_i \quad (\text{car } A_i U = \sum_{k \in \mathcal{F}_i} \alpha_{ik} U_k), \\
\hat{U}_i &= \left(Y_{i\cdot} C^i V^\top - \frac{A_i U V C^i V^\top}{|\mathcal{F}_i|} \right) (V C^i V^\top + \lambda_1 I_d)^{-1}.
\end{aligned} \tag{4.11}$$

Dans cette règle de mise à jour, le vrai problème n'est pas l'inversion de la matrice $d \times d$ mais le calcul de la matrice $V C^i V^\top$, qui est de complexité en $\mathcal{O}(p \times d^2)$ (puisque C^i est diagonale) et que nous allons réduire grâce à une astuce de calcul.

Ce produit est trop coûteux même pour de petites données car il faut le calculer pour chaque utilisateur. En reprenant l'astuce présentée par [Hu *et al.* 2008], nous remplaçons $V C^i V^\top$ par $V V^\top + V(C^i - I)V^\top$. Comme le produit $V V^\top$ est indépendant de l'utilisateur i , il peut être calculé une fois pour toute avant chaque itération (et non pas pour chaque utilisateur i), et en choisissant intelligemment c_{ij} , le produit $V(C^i - I)V^\top$ peut être calculé efficacement. L'idée est d'annuler un maximum de termes diagonaux de la matrice $C^i - I$. Fixer le poids comme dans l'équation 4.12 est un choix intéressant.

$$c_{ij} = 1 + \beta Y_{ij}, \tag{4.12}$$

où β est une constante donnant plus ou moins de poids aux données connues. En effet, les termes diagonaux de $C^i - I$ seront nuls pour chaque j où $Y_{ij} = 0$. Ainsi il ne sera pas nécessaire de calculer $V(C^i - I)V^\top$, mais seulement $V_{\mathcal{Y}_i}(C^i - I)V_{\mathcal{Y}_i}^\top$, où $\mathcal{Y}_i$ est l'ensemble des articles qu'un utilisateur i apprécie. Étant donné que la matrice Y est par nature parcimonieuse, nous avons $|\mathcal{Y}_i| \ll p$. Ceci conduit à une complexité en $\mathcal{O}(|\mathcal{Y}_i| \times d^2)$ linéaire par rapport au nombre d'articles que l'utilisateur i apprécie.

4.3.2.2 Mise à jour de V

Pour mettre à jour la matrice V , nous utilisons la matrice U' définie par $U'_i = U_i + \sum_{k \in \mathcal{F}_i} \frac{\alpha_{ik} U_k}{|\mathcal{F}_i|}$ pour chaque utilisateur i . En injectant U' dans la fonction coût, on obtient :

$$\mathcal{L}(U, V, A) = \sum_{i,j} c_{ij} (U'_i V_j - Y_{ij})^2.$$

Le calcul du gradient par rapport à V_j est alors direct et la règle de mise à jour peut s'écrire sous forme matricielle en définissant la matrice diagonale C^j par $C^j_{ii} = c_{ij}$ ². Ainsi la règle de mise à jour de V_j est la suivante :

$$\begin{aligned} 0 &= \nabla_{V_j} J(\hat{V}_j), \\ 0 &= \left[\nabla_{V_j} \sum_i c_{ij} (U'_i V_j - Y_{ij})^2 + \lambda_2 \|V_j\|^2 \right]_{\hat{V}_j}, \\ 0 &= 2(U'^T C^j (U' \hat{V}_j - Y_{\bullet j}) + \lambda_2 \hat{V}_j), \\ 0 &= U'^T C^j U' \hat{V}_j - U'^T C^j Y_{\bullet j} + \lambda_2 \hat{V}_j, \\ \hat{V}_j &= (U'^T C^j U' + \lambda_2 I_d)^{-1} U'^T C^j Y_{\bullet j}. \end{aligned} \quad (4.13)$$

Pour calculer le produit coûteux, nous proposons de réutiliser l'astuce décrite plus haut en réécrivant $U'^T C^j U' = U'^T U' + U'^T (C^j - I)_{\mathcal{Y}_j} U'_{\mathcal{Y}_j}$, où $\mathcal{Y}_j$ désigne l'ensemble des utilisateurs qui apprécient l'article j . Tout comme dans le paragraphe concernant la mise à jour de U , nous calculons $U'^T U'$ une fois avant chaque itération et non pour chaque article j , et la complexité du calcul de $U'^T (C^j - I)_{\mathcal{Y}_j} U'_{\mathcal{Y}_j}$ est en $\mathcal{O}(|\mathcal{Y}_j| \times d^2)$.

4.3.2.3 Mise à jour de A

Nous allons proposer dans cette partie deux approches permettant la mise à jour de la matrice A .

Une première approche consiste à travailler ligne par ligne, c'est-à-dire mettre à jour $A_{i\bullet}$ pour chaque utilisateur i . Comme nous ne cherchons pas à prédire de nouveaux liens d'amitié entre utilisateurs, mais simplement le niveau d'influence des amis d'un utilisateur i donné, la mise à jour de $A_{i\bullet}$ se ramène à celle de $A_{i\mathcal{F}_i}$. Nous annulons le gradient comme précédemment pour obtenir la règle de mise à jour suivante sous forme matricielle :

$$\hat{A}_{i\mathcal{F}_i} = (Y_{i\bullet} C^i V^T U_{\mathcal{F}_i}^T - U_i V C^i V^T U_{\mathcal{F}_i}^T) \left(\frac{U_{\mathcal{F}_i} V C^i V^T U_{\mathcal{F}_i}^T}{|\mathcal{F}_i|} + \lambda_3 I_{\mathcal{F}_i} \right)^{-1}. \quad (4.14)$$

Notons à nouveau que le coût de calcul du produit $U_i V C^i V^T U_{\mathcal{F}_i}^T$, peut être réduit si nous appliquons la même astuce que celle des règles de mise à jour de U_i et de V_j . Le principal obstacle en terme de complexité de cette approche se situe dans le calcul de l'inverse de la matrice qui est de taille $|\mathcal{F}_i| \times |\mathcal{F}_i|$, ce qui implique une complexité en $\mathcal{O}(|\mathcal{F}_i|^3)$, c'est-à-dire que la complexité est cubique par rapport au nombre d'amis de l'utilisateur i . Suivant le réseau social, si nous avons $d \ll |\mathcal{F}_i|$ pour un pourcentage significatif d'utilisateur, cette règle de mise à jour peut s'avérer coûteuse en terme de temps de calcul.

2. Notons que $C^j \in \mathbb{R}^{n \times n}$ tandis que $C^i \in \mathbb{R}^{p \times p}$

Une seconde approche pour mettre à jour A consiste à calculer les termes de A , non pas utilisateur par utilisateur mais terme à terme, c'est-à-dire mettre à jour $\alpha_{ii'}$ pour deux utilisateurs i et i' donnés. Nous calculons ici la dérivée partielle de la fonction objectif par rapport à $\alpha_{ii'}$ et nous l'annulons pour obtenir la règle de mise à jour suivante :

$$\hat{\alpha}_{ii'} = \left(Y_{i\bullet} - U_i V - \sum_{\substack{k \in \mathcal{F}_i \\ k \neq i'}} \frac{\alpha_{ik} U_k V}{|\mathcal{F}_i|} \right) C^i V^\top U_{i'}^\top \left(\frac{U_{i'} V C^i V^\top U_{i'}^\top}{|\mathcal{F}_i|} + \lambda_3 \right)^{-1} \quad (4.15)$$

Dans ce cas, nous avons seulement à inverser un scalaire au lieu d'une matrice. Nous pouvons également utiliser la même astuce que précédemment pour le produit apparemment coûteux $V C^i V^\top$. Celui-ci peut en effet être réécrit comme $V C^i V^\top = V V^\top + V_{\mathcal{Y}_i} (C^i - I)_{\mathcal{Y}_i} V_{\mathcal{Y}_i}^\top$, où $\mathcal{Y}_i$ est l'ensemble des articles appréciés par l'utilisateur i . Dans cette approche, l'obstacle computationnel majeur peut être le nombre d'itérations qui est exactement le nombre de relations dans le graphe social.

Pour comparer ces deux approches, nous devons considérer le calcul de l'ensemble des termes $A_{i\bullet}$ pour un utilisateur i donné. La complexité de l'équation 4.15 croît linéairement par rapport au nombre d'amis de i (en $\mathcal{O}(|\mathcal{F}_i| \times d^3)$), tandis que la complexité dans l'équation 4.14 est polynomiale par rapport au nombre d'amis de i (en $\mathcal{O}(|\mathcal{F}_i|^3)$). Suivant la nature des données, il faudra choisir la méthode la plus intéressante. Toutefois, il est plus que courant de rencontrer des réseaux sociaux où un grand nombre d'utilisateurs ont beaucoup d'amis, bien plus que le rang que l'on choisit pour le modèle ($|\mathcal{F}_i| \gg d$). Dans ce cas la règle de mise à jour de l'équation 4.15 sera clairement plus performante que celle de l'équation 4.14 en terme de complexité.

4.3.2.4 Algorithme

À partir de ces règles de mise à jour, nous posons l'algorithme 6 d'optimisation de la fonction objectif 4.10 selon U , V et A . Nous discuterons de l'initialisation des différents paramètres dans la partie expérimentation.

Entrées : Y , $\mathcal{F}$

Initialiser U, V et A

Répéter

Pour $i \in \mathcal{U}$ **faire**

 mettre à jour U_i comme spécifié dans l'équation (4.11)

Fin Pour

Pour $j \in \mathcal{V}$ **faire**

 mettre à jour V_j comme spécifié dans l'équation (4.13)

Fin Pour

Pour $i \in \mathcal{U}$ **faire**

Pour $k \in \mathcal{F}_i$ **faire**

 mettre à jour α_{ik} comme spécifié dans l'équation (4.15)

Fin Pour

Fin Pour

jusqu'à ce que (convergence globale)

Nous avons également proposé une version légèrement modifiée de l'algorithme 6. Au lieu de mettre à jour alternativement U , V et A , nous fixons A et optimisons alternativement U et V jusqu'à convergence. Une fois la convergence atteinte, nous mettons A à jour pour U et V fixés. Nous itérons ce processus jusqu'à une convergence globale. Ainsi obtenons-nous l'algorithme 7.

```

Entrées :  $Y, \mathcal{F}$ 
Initialiser  $U, V$  et  $A$ 
Répéter
  Répéter
    Pour  $i \in \mathcal{U}$  faire
      | mettre à jour  $U_i$  comme spécifié dans l'équation (4.11)
    Fin Pour
    Pour  $j \in \mathcal{V}$  faire
      | mettre à jour  $V_j$  comme spécifié dans l'équation (4.13)
    Fin Pour
  jusqu'à ce que ( $U$  et  $V$  aient convergé)
  Pour  $i \in \mathcal{U}$  faire
    Pour  $k \in \mathcal{F}_i$  faire
      | mettre à jour  $\alpha_{ik}$  comme spécifié dans l'équation (4.15)
    Fin Pour
  Fin Pour
jusqu'à ce que (convergence globale)

```

Algorithme 7 – Socially-Enabled Collaborative Filtering v2

Nous avons constaté que ce deuxième algorithme permettait une convergence plus rapide vers une solution, tout en assurant une meilleure stabilité.

4.3.2.5 Prédiction

Notons qu'une fois le modèle appris, les prédictions peuvent être réalisées en utilisant la fonction de décision de l'équation 4.9. Ceci nécessite qu'au moment de la prédiction l'ensemble du réseau social soit en mémoire et de nombreux accès mémoire risquent de ralentir le calcul du score. Les systèmes de recommandation commerciaux modernes n'ont généralement que quelques millisecondes pour fournir des recommandations. Pour accélérer cette opération nous pouvons précalculer les facteurs utilisateurs $U'_i = U_i + \sum_{k \in \mathcal{F}_i} \frac{\alpha_{ik}}{|\mathcal{F}_i|} U_k$. Le calcul du score devient alors $F_{ij} = U'_i V_j$, et le modèle est simplement composé de la matrice U' et la matrice V et non plus de U , V et A .

4.3.3 Expérimentations

Pour mettre en évidence l'intérêt de notre méthode nous avons déterminé un protocole d'évaluation et sélectionné un certain nombre de méthodes de l'état de l'art comme outil de comparaison. Pour valider notre modèle, nous avons utilisé deux ensembles de données réelles. Le premier provient d'un réseau social en ligne espagnol : *Tuenti*. Le second est l'ensemble de données *Epinions* disponibles sur Internet.

4.3.3.1 Les Problèmes

Tuenti Nous utilisons les données du service de lieux du réseau social en ligne *Tuenti*³, qui est appelé *TuentiSitios*. *Tuenti* est le leader espagnol des réseaux sociaux en terme de trafic. Plus de 80% des espagnols âgés de 14 à 27 ans utilisent le réseau. Depuis sa mise en place en 2006, le service n'a été accessible que par invitation d'une personne déjà membre, et celui-ci compte désormais plus de 13 millions d'utilisateurs pour plus d'un milliard de pages vues quotidiennement.

Depuis le lancement, une composante principale a été la connexion entre les utilisateurs, leur identité et leur localisation.

En 2010, *Tuenti* a enrichi l'expérience des utilisateurs autour de la géolocalisation et des relations géographiques. Un module a été ajouté à la plateforme *Tuenti*, dans laquelle les utilisateurs pouvaient dire à leur amis où ils étaient et quels endroits ils affectionnaient particulièrement. Ces lieux étaient alors ajoutés à leur profil. Ce service a contribué au changement progressif du réseau social qui s'est de plus en plus ancré à des positions géographiques spécifiques. Afin d'améliorer ce service, l'entreprise souhaite recommander aux utilisateurs des lieux qu'ils seraient susceptibles d'apprécier en tirant avantage de leur historique personnel et de celui de leurs amis.

Epinions Le site *Epinions* est une plateforme sociale sur laquelle les utilisateurs peuvent échanger leurs opinions sur différents types d'articles. Les utilisateurs peuvent voter pour des articles et sont connectés socialement entre eux à travers un graphe de confiance. L'idée ici aussi est de recommander aux utilisateurs un certain nombre d'articles à partir des informations de vote et du graphe social.

4.3.3.2 Les données

Pour tester notre modèle, nous avons utilisé comme matrice Y la matrice d'interaction lieu-utilisateur de *Tuenti*, qui contient tous les lieux que les utilisateurs ont ajoutés à leur profil. Nous avons également utilisé le réseau social $\mathcal{F}$ de *Tuenti*, c'est-à-dire la matrice contenant l'ensemble des amis des utilisateurs. Les données contiennent environ 10 millions d'utilisateurs et approximativement 100 000 lieux. Chacune des matrices est très parcimonieuse, étant donné que chaque utilisateur a apprécié en moyenne 4 lieux et a 20 amis sur son profil. Notons que le stockage de la matrice Y prend 2 Go et que celui du graphe social avoisine 22Go. Il s'agit d'un ensemble de données massives (« *big data* ») à échelle industrielle.

Nous testons également notre modèle sur un jeu de données libre disponible sur l'Internet qui est *Epinions*⁴. Les données *Epinions* contiennent, quant à elles, 50000 utilisateurs pour 140000 articles. Sur ce jeu de données aussi les matrices sont parcimonieuses : un utilisateur est connecté socialement à 10 autres utilisateurs en moyenne et a émis une opinion sur environ 13 articles. Le graphe social a été construit suivant la confiance que les utilisateurs s'accordent entre eux. Il s'agit donc d'un graphe de confiance. Contrairement à *Tuenti*, les données d'*Epinions* sont des données de vote avec des valeurs comprises entre 1 et 5. En remplaçant ces votes par la valeur 1, nous transformons les données en appréciations implicites en posant le problème comme dans le défi KDD 2007 *who-rated-what*?⁵ Ainsi nous ne cherchons plus à prédire l'appréciation d'un utilisateur pour un article mais à prédire si un utilisateur donnera une appréciation pour un article.

La principale différence entre nos deux jeux de données réside dans la nature du graphe social. Dans celui de *Epinions* est clairement reflétée la confiance ou l'influence des utilisateurs entre eux,

3. www.tuenti.com

4. www.trustlet.org/wiki/Epinions_datasets

5. www.cs.uic.edu/~liub/KDD-cup-2007/proceedings.html

tandis que dans celui de *Tuenti* la notion de relation entre utilisateur est bien plus large, elle peut aller de meilleur ami à simple connaissance, ce qui ne traduit que rarement la notion de confiance.

Il y a un intérêt à utiliser ces deux jeux de données. Grâce à la base de données *Tuenti* nous pouvons tester le passage à l'échelle de notre approche, ce qui est un point crucial dans le domaine de la recommandation. Grâce à la base de données *Epinions*, qui est publique, nous pouvons nous comparer avec l'état de l'art de façon plus transparente et mettre en avant la généralité de notre approche.

4.3.3.3 Protocole d'évaluation

Pour la procédure d'évaluation, nous avons adopté une stratégie similaire à [Cremonesi et al. 2010]. Nous avons découpé les données en deux parties, un ensemble d'apprentissage pour apprendre notre modèle et un ensemble de test pour l'évaluation. L'ensemble de test contient les 25 derniers pour cent des lieux (ou articles) ajoutés sur les profils des utilisateurs et l'ensemble d'apprentissage contient les lieux précédents. Pour chaque utilisateur, nous avons tiré aléatoirement plusieurs entrées $Y_{ij} = 0$ et avons supposé ces lieux comme étant effectivement non appréciés par l'utilisateur i .

Nous avons appris le modèle pour pouvoir calculer F_{ij} pour chaque utilisateur i et lieu j dans l'ensemble de test. Nous avons ensuite ordonné les lieux pour chaque utilisateur suivant leurs scores, du plus pertinent au moins pertinent. La meilleure façon d'évaluer notre modèle est d'utiliser une métrique d'ordonnement (de *ranking*). Une métrique d'ordonnement populaire pour ce type de données est la métrique *Mean Average Precision* (MAP) qui est particulièrement adaptée à l'ordonnement de recommandation parce qu'elle accorde de l'importance au fait d'avoir en tête de liste des articles pertinents. Dans notre cas, la MAP peut être écrite comme dans l'équation 4.16.

$$\text{MAP} = \frac{1}{n} \sum_{i=1}^n \frac{\sum_{k=1}^p P(k) Y_{ik}}{|\mathcal{Y}|} \quad (4.16)$$

où $P(k)$ désigne la précision à k (définie comme dans l'équation 2.18).

Nous avons également calculé la métrique RANK décrite dans [Hu et al. 2008] pour évaluer les performances des différents modèles. Contrairement à la MAP, plus la valeur du RANK est petite, meilleures sont les performances. Cette métrique possède les mêmes avantages que la MAP dans un problème d'ordonnement de recommandation. Comme nous n'avons pas de données de *rating*, la métrique RANK peut s'écrire, dans notre cas, comme dans l'équation 4.17.

$$\text{RANK} = \frac{\sum_{i,j} Y_{ij} \text{rank}_{ij}}{|\mathcal{Y}|} \quad (4.17)$$

où rank_{ij} est la position d'ordonnement normalisée (c'est-à-dire les valeurs des positions vont de $\frac{1}{p}$ à 1) de l'article j pour un utilisateur i donné.

Enfin la sélection des hyperparamètres (λ_1 , λ_2 , λ_3 et β) s'est faite à travers une validation croisée sur une grille de recherche des paramètres (*grid search*).

4.3.3.4 Méthodes en comparaison

Pour évaluer les performances relatives de notre méthode, nous la comparons à plusieurs méthodes de l'état de l'art. La première de ces méthodes à laquelle nous la comparons est une méthode

de factorisation décrite dans [Hu *et al.* 2008]. Cette méthode est profilée pour des données d'appréciation implicite mais ne prend pas en compte le graphe social. Elle optimise également une fonction objectif en mettant à jour alternativement les matrices des utilisateurs et des articles avec une procédure des moindres-carrés alternés (ALS). Nous pourrions ainsi juger, à l'aide de la comparaison avec cette méthode, à quel point l'utilisation du réseau social améliore les performances de recommandation. Dans la suite, nous noterons cette méthode *iMF* (pour *implicit feedback Matrix Factorization*).

Nous avons vu dans le paragraphe 4.2 que les travaux de [Yang *et al.* 2011] tirent avantage du réseau social à l'aide des informations contextuelles pour réaliser leur recommandation. Les facteurs utilisateurs appris servent à prédire à la fois les articles susceptibles d'intéresser l'utilisateur et les utilisateurs susceptibles d'être ses amis. Comme nous nous concentrons sur l'aspect social, nous n'utilisons pas de données contextuelles autres que le graphe social. En adaptant leur fonction objectif à notre environnement, nous arrivons à l'optimisation de la fonction décrite dans l'équation 4.1. La méthode a été testée avec différentes fonctions coût, nous avons choisi celle qui donnait les meilleurs résultats sur leur problème : un coût logistique. Nous avons utilisé une norme de Frobenius pour le terme régularisant. Comme dans [Yang *et al.* 2011], un algorithme de descente de gradient stochastique a été utilisé pour optimiser la fonction objectif. Pour la suite, nous nommerons cette méthode *LLA* (du nom de l'article : *Like Like Alike*).

La troisième méthode avec laquelle nous avons comparé notre méthode est introduite dans [Ma *et al.* 2011]. Elle prend en compte les données sociales en pénalisant une distance quadratique entre les amis dans la fonction objectif. Trois approches sont proposées pour pénaliser cette distance entre amis. Nous avons choisi celle qui leur a donné les meilleures performances, à savoir celle explicitée dans l'équation 4.6, où le score de similarité $sim(i, i')$ entre un utilisateur i et un utilisateur i' est calculé en utilisant les similarités d'espaces vectoriels. Ici aussi une descente de gradient stochastique est utilisée pour optimiser la fonction objectif. Dans la suite de l'article, nous appellerons cette méthode *RSR* (du nom de l'article *Recommender systems with Social Regularization*).

La dernière méthode qui tire avantage du graphe social avec laquelle nous comparons notre méthode est celle décrite dans [Ma *et al.* 2009]. Un coût similaire à celui présenté dans cet article est minimisé, mais les travaux se concentrent sur des données de type d'appréciation explicite (données de vote) et n'optimise pas la matrice d'influence A . Ils utilisent une matrice de confiance précalculée et fixée pour l'optimisation et apprennent leur modèle en optimisant les facteurs U et V résultat de la factorisation d'une matrice de vote. Comme nous travaillons sur des données d'appréciation implicite, nous calibrons leur méthode au problème de données d'appréciation implicite, en fixant d'une part la matrice A au début du processus d'optimisation et en utilisant d'autre part la pondération de la fonction coût (avec l'utilisation des coefficients c_{ij}) pour rendre possible l'apprentissage sur des appréciations implicites. Nous noterons cette méthode *Trust Ensemble* dans la suite de l'article.

Nous avons aussi comparé notre modèle à une méthode basique : le prédicteur moyen, qui recommande les lieux les plus populaires aux utilisateurs.

Notre méthode est notée *SECoFi* dans la suite du chapitre.

4.3.3.5 Résultats d'expériences

Résultats sur les données *Tuenti* Nous initialisons aléatoirement U et V selon une loi uniforme entre 0 et 1. Pour l'initialisation du poids des amis α_{ij} , nous avons déterminé empiriquement que les meilleures performances sont atteintes en initialisant à $\alpha_{ij} = 1$. Nous avons également estimé que la valeur optimale pour le paramètre β (utilisé dans le coefficient c_{ij}) est $\beta = 30$, en accord avec les métriques MAP et RANK (cf. figure 4.2). Nous avons utilisé cette valeur de β pour toutes les expériences.

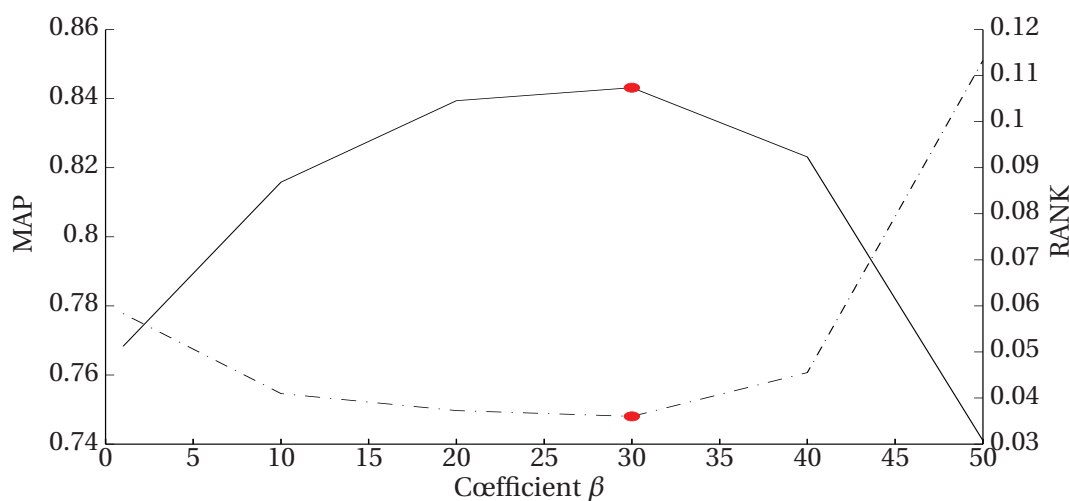


FIGURE 4.2 – Performances de notre modèle en terme de MAP et RANK par rapport à la valeur du coefficient β sur les données *Tuenti*. La valeur optimale de β est 30.

Nous avons validé les performances de notre modèle pour un ensemble de valeurs du nombre de facteurs d (1, 5, 10, 15 et 20). Nous avons répété l'expérience plusieurs fois (10) pour chaque méthode et reporté la valeur moyenne des calculs ainsi que les intervalles de confiance à 95%. Ainsi nous avons testé les différentes méthodes avec plusieurs valeurs de d et tracé la courbe des résultats sur la figure 4.3.

Nous observons que même pour un petit nombre de facteurs, notre méthode est plus performante que les méthodes alternatives utilisant les informations sociales (*LLA* et *RSR*) en terme de MAP et de RANK (plus de 17% d'amélioration pour la MAP et plus de 14% pour le RANK). De plus notre méthode est statistiquement meilleure que *iMF* en terme de MAP, et pour de grandes valeurs de d notre méthode devient statistiquement équivalente à *iMF* en terme de RANK. Notre méthode est également plus performante en terme de MAP et de RANK que la méthode *Trust Ensemble*. Étonnamment *iMF* obtient de meilleurs résultats que les méthodes *LLA* et *RSR*. La raison de ceci peut être le haut niveau de parcimonie des données qui privilégie les méthodes prenant en compte l'ensemble des données non observées.

Les performances relatives entre les méthodes ne dépendent pas fortement du nombre de variables latentes utilisées, à part pour la méthode *Trust Ensemble* qui était statistiquement équivalente à notre méthode pour de faibles dimensions, mais nous voyons indubitablement la différence pour de plus grandes dimensions. En effet, les performances de *SECoFi* sont meilleures que celles de *Trust Ensemble* aussi bien en terme de MAP que de RANK pour un nombre de facteurs $d \geq 10$. Notons enfin que les performances de *SECoFi* sont meilleures que toutes les autres méthodes pour toutes les valeurs de d testées (en terme de MAP).

Nous avons jusqu'à maintenant confirmé que les performances relatives de notre modèle ne dépendent pas de d pour la plupart des méthodes alternatives et nous avons également observé que les performances relatives de notre méthode par rapport à *Trust Ensemble* sont améliorées pour les plus grandes valeurs de d . De plus, nous avons observé que les optimaux des paramètres de régularisation λ_i de *SECoFi* étaient tout le temps les mêmes, indépendamment de la valeur de d . Ceci signifie que la sélection de modèle peut être réalisée d'une façon beaucoup plus directe car une fois les optimaux trouvés pour une valeur donnée de d , nous pouvons fixer les λ_i à ces valeurs optimales et optimiser

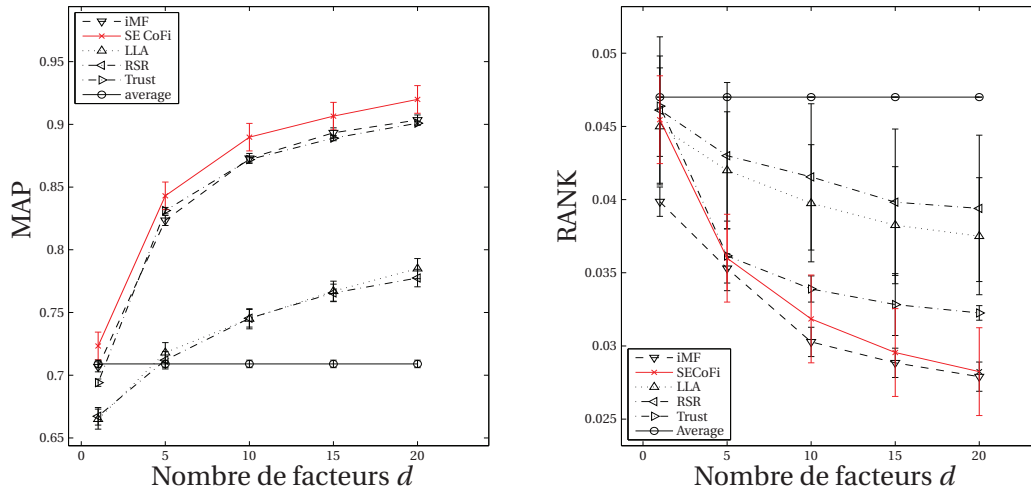


FIGURE 4.3 – Performances des différentes méthodes sur les données *Tuenti* en terme de MAP et de RANK par rapport au nombre de facteurs d .

pour une valeur de d quelconque. Ce n'est pas le cas pour les méthodes de type gradient stochastique (SGD).

Les performances de notre méthode justifient l'utilisation du graphe social, particulièrement en terme de MAP étant donné que nous améliorons un peu les performances par rapport à *iMF* qui ne prend pas en compte les données sociales. De plus, il semble que les méthodes basées sur une optimisation alternée des moindres carrés fournissent de meilleurs résultats que celles utilisant la SGD. Notons que les méthodes basées sur la SGD sous-échantillonnent les données non observées pour éviter de biaiser l'estimateur.

Les résultats suggèrent qu'avec des valeurs plus grandes de d , les performances s'améliorent encore. Cependant l'espace mémoire requis pour de plus grandes valeurs de d (quelque soit la méthode) commence à constituer un obstacle réel, bien que les temps de calcul reste encore acceptables.

Résultats sur les données *Epinions* Afin de valider le fait qu'un apprentissage des valeurs α au cours du processus d'optimisation garantit de meilleures performances, nous avons réalisé les mêmes tests sur le jeu de données *Epinions*. Les résultats de ces tests sont visibles sur la figure 4.4. Nous arrivons, de façon encore plus marquée, à la même conclusion que celle tirée des données *Tuenti*, à savoir qu'apprendre les pondérations d'amitié au cours du processus d'optimisation mène à de meilleures performances que simplement utiliser le graphe de confiance disponible. De plus les performances relatives de *SECoFi* sont vraiment bonnes, même pour un faible nombre de facteurs d . On peut noter en remarque que *iMF*, qui ne prend pas en compte les données sociales, donne de nettement moins bons résultats sur ce jeu de données que sur *Tuenti*, tandis que les autres méthodes gardent un niveau de performance correct.

Ces résultats pointent du doigt le fait que la nature du graphe social a une importance considérable sur les performances des modèles. En effet sur un jeu de données où le graphe social ne représente pas de lien d'influence fort, les performances des modèles tirant avantage des données sociales chutent (à part notre modèle), tandis qu'elles dépassent les performances du modèle ne tirant pas avantage des informations contextuelles sociales sur un ensemble où le graphe social est un graphe de confiance.

En plus des performances du modèle, nous nous sommes intéressés aux valeurs de l'influence des

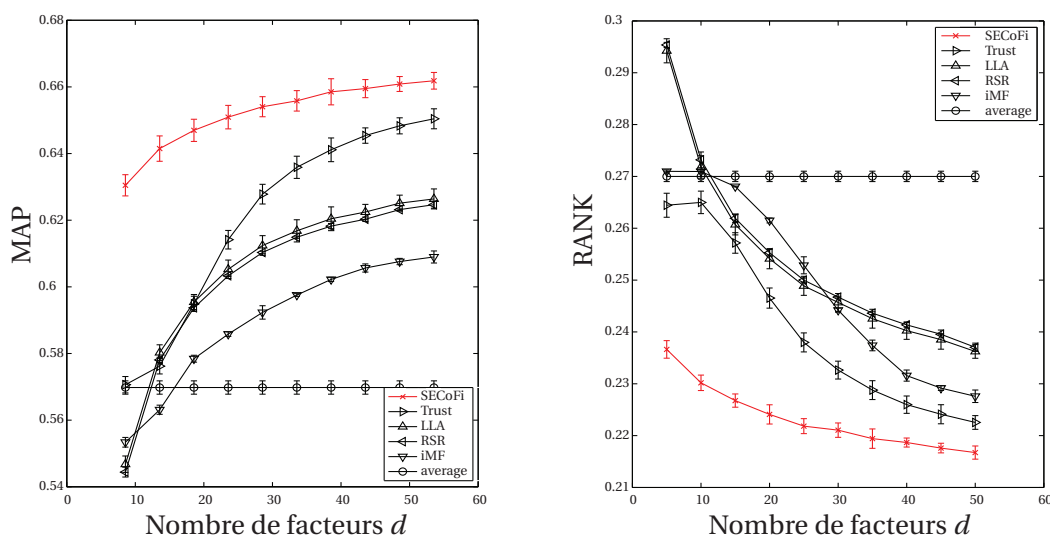


FIGURE 4.4 – Performances des différentes méthodes sur les données *Epinions* en terme de MAP et de RANK par rapport au nombre de facteurs d .

utilisateurs entre eux (les coefficients de la matrice A). Comme nous n'avons pas contraint les valeurs de α , nous voulions voir comment celles-ci se comportaient, et plus particulièrement au niveau des valeurs négatives. Une valeur négative de α peut être vue comme la mesure d'une méfiance ou d'une influence négative entre un utilisateur et un ami, c'est-à-dire à quel point un utilisateur ne va pas aimer un article que son ami aime ou à quel point cet utilisateur va aimer ce que son ami n'aime pas (cf. paragraphe 4.1).

Nous avons tracé la distribution des valeurs de α sur la figure 4.5 pour les données *Tuenti*. Nous observons que la distribution est bimodale. Comme on peut le voir dans la figure 4.5, très peu de valeurs sont négatives entre $-0,1$ et 0 , la plupart (environ 80%) sont entre 0 et $0,1$. Ainsi, lorsque des valeurs négatives apparaissent dans la matrice A , celles-ci sont si proches de zéro qu'elles pourraient être considérées comme étant zéro, ce qui connote l'absence d'influence d'un utilisateur sur un de ses amis. Nous pouvons conclure que le principe d'Homophilie est confirmé pour notre modèle. En effet, les personnes qui s'apprécient auront tendance à avoir des goûts communs (et dans le pire des cas des goûts très faiblement corrélés, mais des goûts anticorrélés absolument pas significatifs).

Nous nous sommes ensuite intéressés à la distribution des valeurs de α pour les données *Epinions*. Nous l'avons tracée sur la figure 4.6. La répartition des valeurs est également bimodale, mais nous remarquons une différence frappante : la proportion de valeurs autour de 0 est nettement inférieure à celle des valeurs de α pour les données *Tuenti* (30% contre 80%), et de fait, bien évidemment la proportion de valeurs autour de 1 est nettement supérieure. Ceci est un reflet de la nature même des données sociales de chaque jeu de données. En effet, le réseau social des données *Epinions* est basé sur la confiance que les utilisateurs s'accordent les uns aux autres, tandis que les relations sociales dans *Tuenti* sont définies de façon beaucoup plus large et peuvent aussi bien représenter une amitié très forte qu'une simple connaissance. Ainsi peu de relation dans *Tuenti* reflète cette notion de confiance/influence présente dans *Epinions*. Ceci est une raison de cette différence de distribution des valeurs de α entre les deux ensembles de données. Ceci explique également les bons résultats obtenus par les méthodes tirant avantage des données sociales sur *Epinions*. Notons que la variance

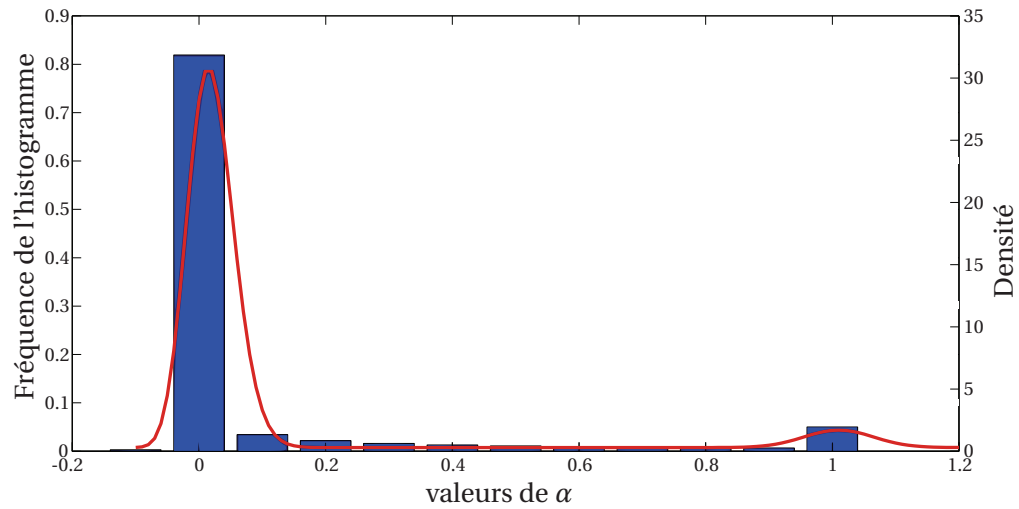


FIGURE 4.5 – Distribution des valeurs de α pour les données *Tuenti*. La distribution est bimodale avec une grande majorité de valeurs positives.

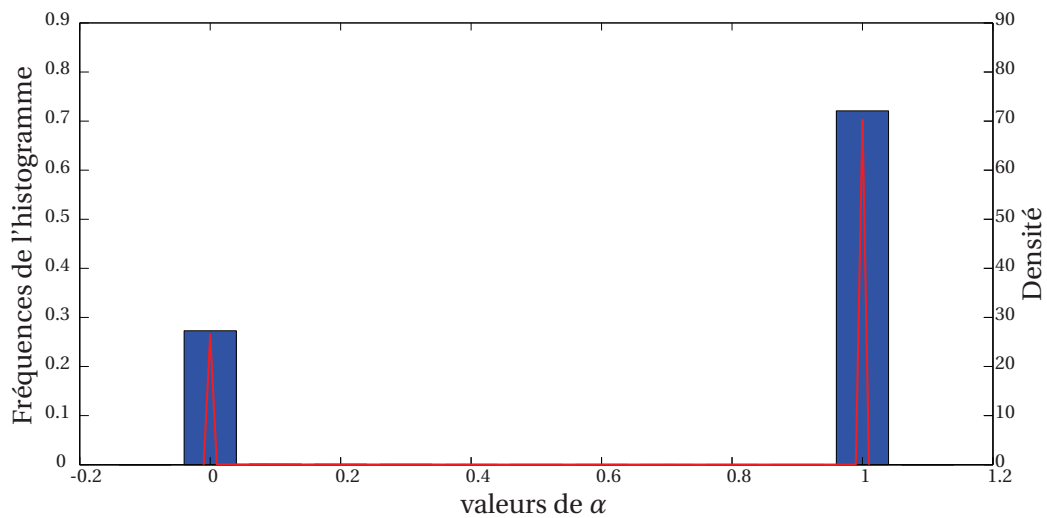


FIGURE 4.6 – Distribution des valeurs de α pour les données *Epinions*.

élevée des performances des méthodes tirant avantage des données sociales sur *Tuenti* (comme le montre la figure 4.3) laissent suggérer cette différence de qualité des graphes sociaux.

Toutes ces observations nous mènent à la conclusion que le modèle *SECoFi* est moins dépendant de la nature du graphe social dont nous disposons. *IMF* donne d'excellents résultats sur les données *Tuenti*, alors que cette méthode ne tire pas du tout avantage des données sociales. Les autres méthodes, à savoir *RSR*, *LLA* ou encore *Trust Ensemble*, tirent avantage des données sociales mais ne sont pas adaptatives, ce qui les pénalise fortement à cause de la définition trop faible de relation d'amitié dans ce contexte. Ces méthodes ont besoin de la préexistence d'un graphe de confiance, où les relations d'amitié respectent fortement le principe d'homophilie, pour conserver un niveau de performance correct. Ce n'est pas le cas de *SECoFi*, qui arrive à ne garder que l'information du réseau social utile pour la recommandation. Cette sélection d'information utile se fait naturellement au cours du processus d'optimisation par une annulation des valeurs de α considérées comme non importantes. Les résultats expérimentaux indiquent que cette sélection se réalise correctement.

4.4 Discussion

Nous avons présenté une méthode qui minimise une fonction objectif en tirant avantage des données du graphe social, afin de réaliser une recommandation personnalisée sur des données d'appréciation implicite. La méthode d'optimisation du modèle est efficace et nous l'avons testée sur un ensemble de données réelles de grande taille, grâce à une complexité linéaire par rapport au nombre de couples utilisateur-article. Nous avons montré que notre façon d'utiliser le graphe social permettait d'améliorer les performances d'autres méthodes de l'état de l'art.

De plus, cette méthode présente des aspects innovants que nous n'avons pas rencontrés dans l'état de l'art. Premièrement, notre méthode permet simultanément de prédire les préférences des utilisateurs pour des articles et de fournir une quantification de la confiance des utilisateurs pour leurs amis. Ensuite, les performances de recommandation de notre modèle ne dépendent que très peu de la nature des données sociales dont nous disposons, que ce soit un graphe de confiance (comme pour *Epinions*) ou un simple graphe d'amitié (comme pour *Tuenti*). La plupart des méthodes tirant avantage des données sociales nécessitent l'utilisation d'un graphe de confiance pour maintenir un niveau de performance correct. Ce n'est pas le cas de notre méthode qui, en estimant ce graphe de confiance au cours du processus d'optimisation, sélectionne l'information importante dans le graphe social et garantit ainsi des performances d'ordonnancement meilleures que celles proposées par les méthodes de l'état de l'art.

Dans de futurs travaux, nous souhaitons, à court terme, comparer les performances des méthodes de l'état de l'art sur l'ensemble de données *Epinions* en prenant en entrée du processus d'optimisation le graphe de confiance initial d'une part, et le graphe de confiance issu de notre méthode. Si nous constatons une amélioration des performances de ces méthodes, nous pourrions confirmer notre hypothèse d'amélioration du graphe social.

À plus long terme, nous envisageons d'améliorer notre méthode en utilisant des données contextuelles complémentaires disponibles sur les lieux (et/ou sur les utilisateurs). En l'occurrence, nous connaissons les coordonnées GPS et le type de lieux dont il s'agit (restaurant, musée...). En utilisant ces informations, nous pourrions construire un graphe mettant en relation les lieux (qui pourrait être vu comme une matrice d'« amitié » entre lieux) et l'intégrer au problème pour améliorer les performances.

Nous envisageons également une approche tensorielle du problème. En effet, nous imaginons

ici une autre façon d'aborder notre problème de recommandation. La factorisation tensorielle est de plus en plus utilisée pour des problèmes de recommandation contextuelle. L'idée serait de définir un tenseur T dans lequel les dimensions représenteraient les utilisateurs, les lieux et les amis (nous pourrions même envisager d'ajouter d'autres dimensions pour d'autres données contextuelles dont nous voudrions tirer avantage). Une idée de définition de tenseur serait la suivante :

$$T_{ijk} = \begin{cases} Y_{ij} & \text{si } k = i, \\ \frac{Y_{jk}S_{ik}}{|\mathcal{F}_i|} & \text{si } k \in \mathcal{F}_i, \\ 0 & \text{sinon.} \end{cases} \quad (4.18)$$

Dans ce tenseur, la matrice T_{iji} correspond exactement à la matrice Y_{ij} . En choisissant une décomposition de Tucker2, on a $\hat{T}_{ijk} = \sum_{p=1}^{d_1} \sum_{q=1}^{d_2} U_{ip} V_{jq} W_{pqk}$ et le problème de factorisation pourrait se formaliser de la façon suivante :

$$\min_{U,V,W} \sum_{i=1}^n \sum_{j=1}^p \sum_{k=1}^n c_{ijk} (T_{ijk} - \hat{T}_{ijk})^2 + \Omega(U, V, W) \quad (4.19)$$

où c est un poids qui joue le même rôle que dans la forme non tensorielle.

Sélection de modèle pour la factorisation

*Truth is stranger than fiction, but it is because
Fiction is obliged to stick to possibilities ; Truth isn't.*
– Mark Twain

Sommaire

5.1	La sélection de modèle	66
5.1.1	Un cadre de sélection de modèle	66
5.1.2	Estimation du risque	66
5.1.3	Choix du critère, de la pénalité et du coût	67
5.2	Sélection de modèle dans la factorisation	67
5.2.1	État de l'art	68
5.2.2	Formalisation du problème	68
5.2.3	Pénalité et estimateur du risque	69
5.2.4	Estimateur sans biais du risque	70
5.2.5	Calcul de la divergence	70
5.2.6	Limites du modèle actuel	71
5.3	Adaptation et élargissement de la méthode	71
5.3.1	Amélioration relative à la fonction de seuillage	71
5.3.2	Passage à l'échelle	76
5.3.3	Estimation de la variance	81
5.4	Discussion	84

Lorsqu'il s'agit de factorisation, un problème récurrent et qui a été peu adressé jusqu'à maintenant est le choix du nombre de facteurs latents à prendre en compte. Dans ce chapitre, nous nous intéresserons au problème de sélection de modèle pour extraire automatiquement le nombre de facteurs latents. Après un état de l'art des méthodes actuellement utilisées, nous verrons comment faire de la sélection de modèle par minimisation d'estimateur de risque pour la factorisation. Enfin nous proposerons un moyen d'adapter cette sélection de modèle à des problèmes de grande taille, comme c'est généralement le cas dans la recommandation.

5.1 La sélection de modèle

La sélection de modèle consiste à évaluer un certain nombre de modèles et les comparer afin de décider lequel représente le mieux le modèle sous-jacent aux données. Nous invitons le lecteur à consulter le paragraphe 2.1 de [Boisbunon 2013] pour une présentation détaillée du problème de sélection de modèle.

5.1.1 Un cadre de sélection de modèle

Nous allons présenter le cadre duquel nous sommes partis pour les travaux de recherche présentés dans ce chapitre. D'autres approches pour la sélection de modèle existent, comme des approches bayésiennes où l'on va par exemple chercher à optimiser le critère d'information de Bayes (BIC) introduit par [Schwarz *et al.* 1978].

Quelque soit le problème de sélection de modèle, on part systématiquement d'un a priori sur le modèle des données. Dans le cadre de la régression, on supposera toujours que pour $X \in \mathbb{R}^{n \times p}$ les données observées et $y \in \mathbb{R}^n$ les labels observés, que le modèle sous-jacent est de la forme :

$$y = r(X) + \epsilon \mathbb{1}, \quad \epsilon \sim \mathcal{N}(0, \sigma^2),$$

où $\mathbb{1}$ est un vecteur de taille p ne contenant que des 1. Nous nous concentrons ici sur le cas gaussien (où ϵ est un bruit gaussien), [Boisbunon 2013] s'intéresse également au cas non gaussien en généralisant le problème au cas des distributions à symétrie sphérique. Si l'on se place dans un contexte de régression linéaire, r sera généralement de la forme $r(X) = X\beta$ avec $\beta \in \mathbb{R}^p$. On cherche ici la fonction r qui explique le mieux les données.

Le principe général de la sélection de modèle est de restreindre le problème en fixant un ensemble de modèles (fonctions f) possibles $\mathcal{F} = \{f_1, \dots, f_m\}$ (notons que $\mathcal{F}$ peut très bien être infini). Le but est de trouver le bon modèle r^* qui minimise un critère donné. Ce critère est le risque de prédiction et se formule ainsi :

$$\mathcal{R}(f) = \mathbb{E}[L(r(X), f(X))], \quad (5.1)$$

où L est une fonction coût (par exemple $L(y, f) = \sum_{i=1}^n (y_i - f_i)^2$). Le principal obstacle de ce problème de minimisation est que l'on ne connaît pas r , nous ne pouvons donc pas trouver de cette façon la fonction r^* qui minimise le risque. L'idée est alors d'estimer ce risque.

5.1.2 Estimation du risque

Nous voulons estimer le risque de prédiction sur notre modèle. Le critère le plus évident est le risque empirique :

$$\mathcal{R}_{\text{emp}}(f) = \frac{1}{n} \sum_{j=1}^n l(y_j, f(x_j)).$$

Dans un cas de régression linéaire avec un coût quadratique pour l , le problème se résoud et $\hat{\beta} = \hat{\beta}_{LS} = (X^\top X)^{-1} X^\top y$. Cependant ce problème peut avoir une infinité de solutions et ces solutions peuvent avoir tendance à surapprendre les données.

Pour éviter ce surapprentissage, on préférera généralement choisir de minimiser un critère de la forme :

$$\min_{f \in \mathcal{F}} J(f) = \mathcal{R}_{\text{emp}}(f) + \Omega(f), \quad (5.2)$$

où Ω est un terme de pénalisation sur f et qui est une mesure de complexité ou un degré de liberté (df) sur f . λ est un paramètre qui joue ici un rôle d'équilibrage entre qualité d'apprentissage et simplicité du modèle. On considère que J est un critère qui estime correctement le risque de prédiction et on sélectionnera le modèle qui minimise ce critère.

Cependant, la taille de $\mathcal{F}$ peut être très grande, voire infinie, et il peut s'avérer difficile de trouver la fonction f qui minimise 5.2. Pour faciliter ceci, on partitionne généralement $\mathcal{F}$ en sous ensembles inclus. Par exemple dans le cas du lasso, ces sous ensembles sont de la forme : $\mathcal{F}_\lambda = \{f = b^\top x \mid \|b\|_1 \leq \lambda\}$, et le problème se réécrit ainsi :

$$\min_{\lambda} \min_{f \in \mathcal{F}_\lambda} \mathcal{R}_{\text{emp}}(f) + \Omega(\lambda). \quad (5.3)$$

On cherchera donc, dans un premier temps à trouver la famille $\{\hat{f}_\lambda\}$ pour $\lambda \in \mathbb{R}$ qui minimise le problème :

$$\min_{f \in \mathcal{F}_\lambda} \mathcal{R}_{\text{emp}}(f),$$

pour un λ donné. On cherchera ensuite le $\hat{\lambda}$ qui minimise

$$\min_{\lambda} \hat{f}_\lambda + \Omega(\lambda).$$

Ainsi nous obtenons $\hat{f}_{\hat{\lambda}}$ qui est un estimateur de r .

5.1.3 Choix du critère, de la pénalité et du coût

De cette forme de critère L ont été dérivés dans la littérature un grand nombre de critères d'estimation du risque. À titre d'exemple, nous pouvons citer les critères les plus utilisés tels qu'ils peuvent apparaître pour un choix de coût quadratique, comme l'estimateur sans biais du risque L_0 , le critère de Mallows C_p , le critère d'information d'Akaike AIC .

$$\begin{aligned} L_0(\hat{f}) &= \|y - \hat{f}(X)\|^2 + (2 \text{div}_y(\hat{f}(X)) - n) \hat{\sigma}^2, \\ C_p(\hat{f}) &= \frac{\|y - \hat{f}(X)\|^2}{\hat{\sigma}^2} + 2 \text{div}_y(\hat{f}(X)) - n, \\ AIC(\hat{f}) &= \frac{\|y - \hat{f}(X)\|^2}{\hat{\sigma}^2} + 2 \text{div}_y(\hat{f}(X)), \end{aligned} \quad (5.4)$$

où $\hat{\sigma}^2$ est un estimateur sans biais de la variance. Notons que ces critères trois sont équivalents dans ce cadre (Boisbunon 2013).

Les problèmes qui découle de cette approche sont multiples : comment partitionner $\mathcal{F}$? comment choisir $\Omega(\lambda)$? et comment optimiser l'ensemble du problème ? Le paragraphe 2.4 de [Boisbunon 2013] donne des éléments de réponse à ces questions.

5.2 Sélection de modèle dans la factorisation

Dans ce paragraphe, nous dresserons un état de l'art de la sélection de modèle dans le cadre de la factorisation matricielle. Nous proposerons ensuite une approche de sélection de modèle pour la factorisation de type minimisation de risque que nous formaliserons. Enfin, nous présenterons les limites d'une telle méthode dans un cadre qui nous intéresse, à savoir le domaine de la recommandation.

5.2.1 État de l'art

Un certain nombre de méthodes proposent de sélectionner le modèle en faisant de la détection de rupture sur les valeurs propres à partir d'une métrique donnée. On les dénote généralement par méthodes du « coude », on cherche en effet à détecter un changement d'état des valeurs singulières qui graphiquement une allure de coude. [Cattell 1966] propose une métrique basée sur la variation des valeurs singulières de la matrice. La métrique proposée par [Jolliffe 2002] est assez similaire, il s'agit d'un pourcentage cumulé des variations des valeurs singulières. On peut trouver d'autres métriques analogues comme dans [He et al. 2009] pour de la factorisation tensorielle. Le principal défaut de ces méthodes réside dans le fait qu'il faut fixer un seuil pour la détection de rupture, ce qui en fait des méthodes de sélection de modèle non automatiques.

D'autres approches s'intéressent à l'estimation du nombre de variables à sélectionner par la minimisation d'un critère de risque. [Ulfarsson & Solo 2008] et [Candès et al. 2012] estiment tous deux le risque sans biais de Stein (SURE). [Candès et al. 2012] utilise une fonction de seuillage doux ($f_\lambda(x) = \max(0, x - l)$) pour approximer le modèle, tandis que [Ulfarsson & Solo 2008] utilise la fonction suivante : $f_\lambda(x) = \max(0, \frac{x-l}{l})$. Un critère à minimiser dérivé du BIC (*Bayesian Information Criterion*) est proposé dans [Seghouane & Cichocki 2007], il s'agit de l'ICPPA (*Information Criterion for Principal Component analysis*)¹. Le modèle est approché à l'aide d'une fonction de seuillage doux. Une approche bayésienne est également proposée par [Minka 2000] qui utilise une approximation de Laplace pour estimer le rang de la matrice. [Hansen et al. 1999] estime, quant à lui, le risque à l'aide d'un critère similaire à l'AIC (*Akaike Information Criterion*). On peut également citer [Bouveyron et al. 2011] qui minimise l'estimateur de risque du maximum de vraisemblance.

Certaines approches sont plus orientées méthodologie et visent à sélectionner le bon modèle par validation croisée par exemple, comme [Eastment & Krzanowski 1982] ou [Dias & Krzanowski 2003]. On peut également trouver des approches de types Monte-Carlo, pour notamment approximer la distribution à posteriori à l'aide d'une chaîne de Markov (Hoff 2007), ou déterminer des critères d'arrêts dans le calcul successif des composantes principales d'une matrice (Peres-Neto et al. 2005). D'autres approches visent à trouver le rang de la matrice en modifiant le problème initial (comme par exemple [Labiod & Nadif 2011a] qui maximise la modularité de la matrice).

Plusieurs méthodes sont donc proposées mais aucune ne gèrent vraiment de façon automatique la sélection de modèle dans des problèmes de grande taille. Notre contribution, présentée dans la suite du chapitre, présente une proposition de réponse à une telle problématique.

5.2.2 Formalisation du problème

Soit $Y \in \mathbb{R}^{n \times p}$ une matrice de données que l'on cherche à factoriser. Sans perte de généralité, nous supposons dans la suite que $p \leq n$. Nous posons le modèle suivant

$$Y = M + R, \quad (5.5)$$

avec M une matrice que l'on cherche à estimer et où R est une matrice de bruit dont chaque composante suit une loi normale centrée $R_{ij} \sim \mathcal{N}(0, \sigma^2)$ pour $i = 1 \dots n$ et $j = 1 \dots p$. Soit $Y = U\Sigma V^\top$ la décomposition en valeurs singulières de la matrice Y . Nous considérons une famille d'estimateurs $\widehat{M}^{(\lambda)} = f_\lambda(Y) = U f_\lambda(\Sigma) V^\top$ obtenue à partir d'une famille de fonctions spectrales f_λ (comme par exemple $f_\lambda(x) = \max(0, x - \lambda)$). Le problème de sélection de modèle consiste ici à chercher le meilleur λ .

1. Pourquoi ICPPA et pas ICPCA ? Nous n'avons pas la réponse.

Soit λ^* l'oracle défini comme dans l'équation 5.6,

$$\lambda^* = \arg \min_{\lambda} \|M - \widehat{M}^{(\lambda)}\|_F^2, \quad (5.6)$$

et soit $\hat{\lambda}$ notre estimateur.

$$\hat{\lambda} = \arg \min_{\lambda} \|Y - \widehat{M}^{(\lambda)}\|^2 + \text{pen}(\lambda, Y, \widehat{M}^{(\lambda)}). \quad (5.7)$$

L'idée générale est de trouver une fonction pen pour que l'erreur sur $\hat{\lambda}$ soit la plus proche possible de celle sur λ^* . Ce qui peut se formuler :

$$\forall \alpha, \exists \eta, \quad \Pr(\|M - \widehat{M}^{(\hat{\lambda})}\|^2 - \|M - \widehat{M}^{(\lambda^*)}\|^2 \geq \eta) \leq \alpha. \quad (5.8)$$

5.2.3 Pénalité et estimateur du risque

Développons l'expression de l'oracle.

$$\begin{aligned} \|M - \widehat{M}^{(\lambda)}\|^2 &= \|M - Y\|^2 + \|\widehat{M}^{(\lambda)} - Y\|^2 - 2\langle M - Y, \widehat{M}^{(\lambda)} - Y \rangle_F \\ &= \|M - Y\|^2 + \|\widehat{M}^{(\lambda)} - Y\|^2 - 2\langle M - Y, \widehat{M}^{(\lambda)} - M \rangle_F - 2\|M - Y\|^2 \\ &= -\|M - Y\|^2 + \|\widehat{M}^{(\lambda)} - Y\|^2 + 2\langle Y - M, \widehat{M}^{(\lambda)} - M \rangle_F. \end{aligned} \quad (5.9)$$

Ce qui nous donne :

$$\lambda^* = \arg \min_{\lambda} \|Y - \widehat{M}^{(\lambda)}\|^2 + 2\langle R, \widehat{M}^{(\lambda)} - M \rangle_F. \quad (5.10)$$

Maintenant, nous allons chercher à borner le terme de gauche de l'inéquation 5.8 :

$$\begin{aligned} &\Pr\left(\|M - \widehat{M}^{(\hat{\lambda})}\|^2 - \|M - \widehat{M}^{(\lambda^*)}\|^2 \geq \eta\right) \\ &\leq \Pr\left(\|Y - \widehat{M}^{(\hat{\lambda})}\|^2 + 2\langle R, \widehat{M}^{(\hat{\lambda})} - M \rangle_F - \|Y - \widehat{M}^{(\lambda^*)}\|^2 - 2\langle R, \widehat{M}^{(\lambda^*)} - M \rangle_F \geq \eta\right) \\ &\leq \Pr\left(-\text{pen}(\hat{\lambda}) + 2\langle R, \widehat{M}^{(\hat{\lambda})} - M \rangle_F + \text{pen}(\lambda^*) - 2\langle R, \widehat{M}^{(\lambda^*)} - M \rangle_F \geq \eta\right) \\ &\leq \sum_{\lambda=1}^{\min(n,p)} \Pr\left(|\text{pen}(\lambda) - 2\langle R, \widehat{M}^{(\lambda)} - M \rangle_F| \geq \frac{\eta}{2}\right) \\ &\leq \sum_{\lambda=1}^{\min(n,p)} \frac{4}{\eta^2} \mathbb{E}\left[\left(\text{pen}(\lambda) - 2\langle R, \widehat{M}^{(\lambda)} - M \rangle_F\right)^2\right], \end{aligned} \quad (5.11)$$

par l'inégalité de Markov. Pour un α fixé, on choisit

$$\eta = \frac{2}{\sqrt{\alpha}} \mathbb{E}\left[\left(\text{pen}(\lambda, Y, U_{\lambda}, V_{\lambda}) - 2\langle R, \widehat{M}^{(\lambda)} - M \rangle_F\right)^2\right]^{1/2}.$$

Plus cette borne est petite, meilleure est la garantie de généralisation. Il nous faut donc minimiser $\mathbb{E}\left[\left(\text{pen}(\lambda) - 2\langle R, \widehat{M}^{(\lambda)} - M \rangle_F\right)^2\right]$ qui peut être interprétée comme le risque statistique associé à l'estimateur $\text{pen}(\lambda)$ du paramètre $2\langle R, \widehat{M}^{(\lambda)} - M \rangle_F$. Cette borne n'est pas optimale mais elle permet de donner une justification au choix du critère.

5.2.4 Estimateur sans biais du risque

L'heuristique de Stein suggère de choisir une pénalité pen telle que

$$\begin{aligned}\mathbb{E}(pen) &= \mathbb{E}\left(2\langle R, \widehat{M}^{(\lambda)} - M \rangle_F\right) \\ &= \mathbb{E}\left(2\langle R, \widehat{M}^{(\lambda)} \rangle_F\right) \quad (\text{puisque } R \text{ est centrée}).\end{aligned}\tag{5.12}$$

Avant d'aller plus loin dans l'expression de la pénalité, nous avons besoin du théorème 5.1.

Théorème 5.1. *Identité de Stein. Soit $X \in \mathbb{R}^{n \times p}$ telle que $X_{ij} \sim \mathcal{N}(\mu_{ij}, \sigma^2)$ et $f : \mathbb{R}^{n \times p} \rightarrow \mathbb{R}^{n \times p}$. Si f est une fonction dérivable en X , alors on a :*

$$\mathbb{E}[\langle X - \mu, f(X) \rangle_F] = \sigma^2 \mathbb{E}[\text{div}_X(f(X))].\tag{5.13}$$

La pénalité proposée par Stein est $pen = 2\hat{\sigma}^2 \text{div}_Y(\widehat{M}^{(\lambda)})$, où $\hat{\sigma}^2$ est un estimateur sans biais de la variance et indépendant des $\widehat{M}^{(\lambda)}$. Vérifions si cette pénalité respecte l'heuristique de Stein :

$$\begin{aligned}\mathbb{E}\left(2\hat{\sigma}^2 \text{div}_Y(\widehat{M}^{(\lambda)})\right) &= 2\sigma^2 \mathbb{E}\left(\text{div}_Y(\widehat{M}^{(\lambda)})\right) \\ &= 2\mathbb{E}\left(\langle Y - M, \widehat{M}^{(\lambda)} \rangle_F\right) \quad (\text{d'après le théorème 5.1}) \\ &= 2\mathbb{E}\left(\langle R, \widehat{M}^{(\lambda)} \rangle_F\right).\end{aligned}\tag{5.14}$$

Ainsi nous avons le terme que nous cherchons et un terme corrigeant le biais de coût quadratique. L'estimateur que nous obtenons est appelé estimateur sans biais du coût et correspond exactement au *SURE* (*Stein Unbiased Risk Estimator*) adapté par [Candès et al. 2012].

$$\text{SURE}(\widehat{M}^{(\lambda)}, Y) = \|Y - \widehat{M}^{(\lambda)}\|^2 + (2\text{div}_Y(\widehat{M}^{(\lambda)}) - np)\hat{\sigma}^2.\tag{5.15}$$

5.2.5 Calcul de la divergence

Le calcul de $\text{div}_Y(f(Y))$ la divergence de la fonction spectrale $f(Y)$ par rapport à la matrice Y a été donné initialement dans [Stein 1973], précisé dans [Konno 1992] et redémontré sous une autre forme dans [Candès et al. 2012]. La fonction spectrale f est définie par :

$$f(Y) = \sum_{i=1}^p f_i(\sigma_i) U_i V_i^\top \triangleq U f(\Sigma) V^\top,\tag{5.16}$$

où les fonctions f_i forment une famille de fonctions différentiables telles que $f(\Sigma) = \text{diag}(f_1(\sigma_1), \dots, f_p(\sigma_p))$, avec $Y = U\Sigma V^\top$ la forme dite réduite de la décomposition en valeurs singulières de Y et $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_p)$ la matrice diagonale des valeurs singulières de Y . Notons que $U \in \mathcal{L}(p, n)$, et nous définissons $\tilde{U} \in \mathcal{L}(n, n)$, où $\tilde{U} = [U \quad Q]$ tel que $U^\top Q = 0$ et $Q^\top Q = I_{n-p}$. Dans ce cadre nous pouvons énoncer le résultat suivant :

Théorème 5.2. *Si Y est une matrice simple et de plein rang, σ_i ($i \in \{1 \dots p\}$) les valeurs singulières de Y et f une fonction spectrale continue et différentiable en Y , alors :*

$$\text{div}_Y(f(Y)) = \sum_{i=1}^p \left(f'_i(\sigma_i) + (n-p) \frac{f_i(\sigma_i)}{\sigma_i} \right) + 2 \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p \frac{\sigma_i f_i(\sigma_i)}{\sigma_i^2 - \sigma_j^2}\tag{5.17}$$

La preuve du théorème 5.2 reprise d'après [Candès et al. 2012] est détaillée en annexe B.1. De plus, l'annexe B.2 présente, à travers un exemple simple, la notion de différentielle et de divergence dans le cadre matriciel.

5.2.6 Limites du modèle actuel

Nous proposons dans ce paragraphe quelques voies d'amélioration du modèle proposé dans [Candès *et al.* 2012] en intervenant sur différents points. Pour cela, nous présentons les limites du modèle actuel et proposons de palier à ces difficultés.

Il est fait état, dans [Candès *et al.* 2012], de l'utilisation d'une fonction de seuillage doux (*soft thresholding*) comme définie dans l'équation 5.18.

$$SOFT(x) = \begin{cases} 0 & \text{si } x \leq \lambda, \\ (x - \lambda) & \text{sinon.} \end{cases} \quad (5.18)$$

Nous nous sommes intéressés aux performances du modèle pour d'autres fonctions de seuillage et montrons dans le paragraphe 5.3.1 que certaines peuvent être plus intéressantes.

Le modèle de Candès présente un défaut important, à savoir qu'il ne passe pas l'échelle. En effet, le calcul de la divergence impose la connaissance de l'ensemble des valeurs singulières de la matrice Y , ce qui est inconcevable en grandes dimensions. Nous proposons ici un moyen d'adapter ce modèle afin qu'il soit utilisable sur des données de grande taille, comme il est souvent possible d'en rencontrer dans le domaine de la recommandation. C'est l'objet du paragraphe 5.3.2.

Enfin, un problème connu est celui de l'estimation de la variance du bruit. En effet, la plupart du temps on estime cette variance à l'aide d'un estimateur sans biais de la variance. Nous proposons dans le paragraphe 5.3.3 une heuristique qui permet d'obtenir une meilleure estimation de la variance.

5.3 Adaptation et élargissement de la méthode

Dans la section 5.2.6, nous avons vu les limites du modèle initialement proposé. Pour chacune des difficultés citées, nous proposerons une voie possible d'amélioration pouvant éventuellement palier à ces obstacles qui peuvent être rencontrés.

Pour chacun des obstacles, nous présenterons la méthode proposée et nous vérifierons ensuite sa validité expérimentalement.

5.3.1 Amélioration relative à la fonction de seuillage

Dans [Candès *et al.* 2012], les auteurs utilisent un estimateur du risque sans biais de Stein (*SURE*) défini comme dans l'équation 5.15 et utilisent systématiquement dans leur modèle une fonction de seuillage f spectrale de type seuillage doux (définie comme dans l'équation 5.18). En restant dans le cas d'un estimateur de risque sans biais, nous proposons d'utiliser d'autres fonctions de seuillage et d'évaluer expérimentalement les performances pour ces différentes fonctions. En effet, il a été montré que, dans un cadre de régression linéaire classique (où le modèle est un vecteur), une fonction de type seuillage doux n'était pas forcément la plus adaptée et que certaines fonctions plus sophistiquées permettaient d'obtenir de meilleures performances (Boisbunon 2013). Nous allons vérifier que ce comportement est observable également dans un cadre de factorisation matricielle.

5.3.1.1 Fonctions de seuillage proposées

Afin que le calcul de la divergence de f soit possible, il faut que f soit continue et dérivable sur Y . C'est le cas du *MCP* (*Minimax Concave Penalty*) introduit par [Bruce & Gao 1996], du *SCAD*

(*Smoothly Clipped Absolute Deviation*) introduit par [Fan & Li 2001] ou encore de l'*adaptive lasso* (*ADALASSO*) introduit par [Zou 2006]. $\forall x \in \mathbb{R}^+$, ces trois fonctions de seuillage sont définies de la façon suivante :

$$MCP(x) = \begin{cases} 0 & \text{si } x \leq \lambda \\ \frac{\gamma}{\gamma-1}(x - \lambda) & \text{si } \lambda < x \leq \gamma\lambda \\ x & \text{si } \gamma\lambda < x, \end{cases} \quad (5.19)$$

$$SCAD(x) = \begin{cases} \max(0, x - \lambda) & \text{si } x \leq 2\lambda \\ \frac{a}{a-2}(x - \lambda) & \text{si } 2\lambda < x \leq a\lambda \\ x & \text{si } a\lambda < x, \end{cases} \quad (5.20)$$

$$ADALASSO(x) = \begin{cases} 0 & \text{si } x^{1+q} \leq \lambda \\ x - \frac{\lambda}{x^q} & \text{sinon.} \end{cases} \quad (5.21)$$

Notons que $MCP(Y)$ est dérivable sur Y , si $\forall i, \sigma_i \neq \lambda$ et $\sigma_i \neq \gamma\lambda$, $SCAD(Y)$ est dérivable sur Y , si $\forall i, \sigma_i \neq \lambda$ et $\sigma_i \neq 2\lambda$ et $\sigma_i \neq a\lambda$ et $ADALASSO(Y)$ est dérivable sur Y , si $\forall i, \sigma_i \neq \lambda$. On supposera que ces conditions de dérivabilité sont remplies, ce que l'on peut obtenir en s'assurant de ne jamais évaluer un modèle pour un λ qui violerait ces contraintes.

Chacune de ses fonctions de seuillage est associée à une pénalité. Celles-ci sont représentées dans la figure 5.1. Ce ne sont bien évidemment pas les seuls fonctions de seuillage, d'autres auraient pu être envisagées, comme par exemple l'*adaptive elastic-net*.

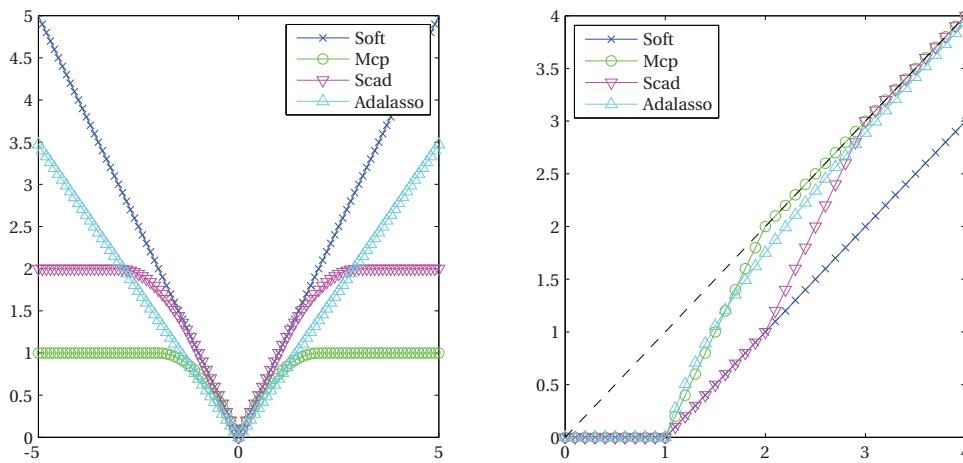


FIGURE 5.1 – Représentation des différentes fonctions de seuillage (à droite) ainsi que leur pénalité associée (à gauche) pour $\lambda = 1$, $a = 3$, $\gamma = 2$ et $q = 2$.

5.3.1.2 Protocole expérimental

Le protocole expérimental élaboré ici est inspiré de celui proposé par [Candès *et al.* 2012]. Soit $Y \in \mathcal{L}(p, n)$ une matrice que nous cherchons à factoriser. Cette matrice est construite suivant le modèle de l'équation 5.5 et $p = 200$ et $n = 500$.

Nous testerons les différents modèles pour un ensemble de matrices M générés aléatoirement suivant un loi normale centrée avec un ensemble de rangs différents, à savoir $\text{rang}(M) \in$

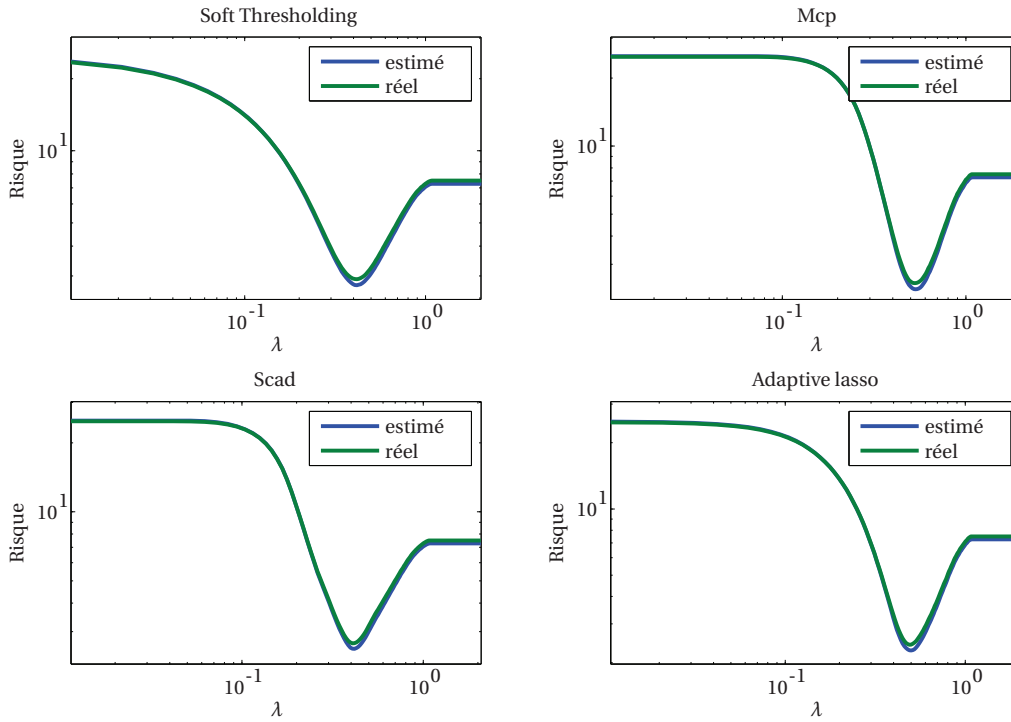


FIGURE 5.2 – Risque estimé et Risque réel en fonction de λ pour les différentes fonctions de seuillage proposées ($SNR = 0,2$, $\text{rang}(M) = 20$)

$\{\frac{p}{10}; \frac{p}{5}; \frac{3p}{10}; \frac{2p}{5}; \frac{p}{2}\}$, ainsi que pour un ensemble de matrices R de variances différentes, de telle sorte que la marge du bruit (*signal noise ratio*) soit $SNR \in \{\frac{1}{10}; \frac{1}{5}; \frac{1}{2}; 1; 2\}$. Nous définissons cette marge de bruit de la même façon que [Candès *et al.* 2012], à savoir $SNR = \frac{1}{\sigma\sqrt{np}}$. Comme dans [Candès *et al.* 2012], nous supposons que σ^2 (la variance de R) est connue pour le calcul de l'estimateur de risque. Nous nous concentrons dans cette partie sur les performances de différentes fonctions de seuillage pour un même estimateur de risque. Le problème de l'estimation de la variance est abordé dans le paragraphe 5.3.3.

Nous recherchons le λ optimal pour chacun des modèles à l'aide d'un *grid search* en répétant l'expérience 100 fois. Ce λ optimal minimise le risque estimé et est censé correspondre au λ^* « oracle » pour lequel le nombre de $f(\sigma_i) > 0$ est égal au rang de la matrice M .

5.3.1.3 Quelques résultats expérimentaux

Dans un premier temps, nous comparons l'estimateur de risque avec le risque réel pour chacune des quatre fonctions de seuillage. La figure 5.2 illustre graphiquement l'écart entre le risque réel et le risque estimé. Nous constatons que l'estimation de risque est bonne, les minima sont atteints pour un même λ et la différence entre les risques estimés et réels reste faible, quelque soit la fenêtre choisie.

Comparons maintenant les risques pour les différentes fonctions proposées. La figure 5.3 présente graphiquement les résultats pour un choix de paramètres fixés (à savoir $SNR = 0,2$ et $\text{rang}(M) = 20$). Le tableau 5.1 présente, quant à lui, les résultats obtenus sur les tests pour l'ensemble des valeurs de paramètres proposés. Pour chaque valeur de la marge de bruit, et pour chaque valeur

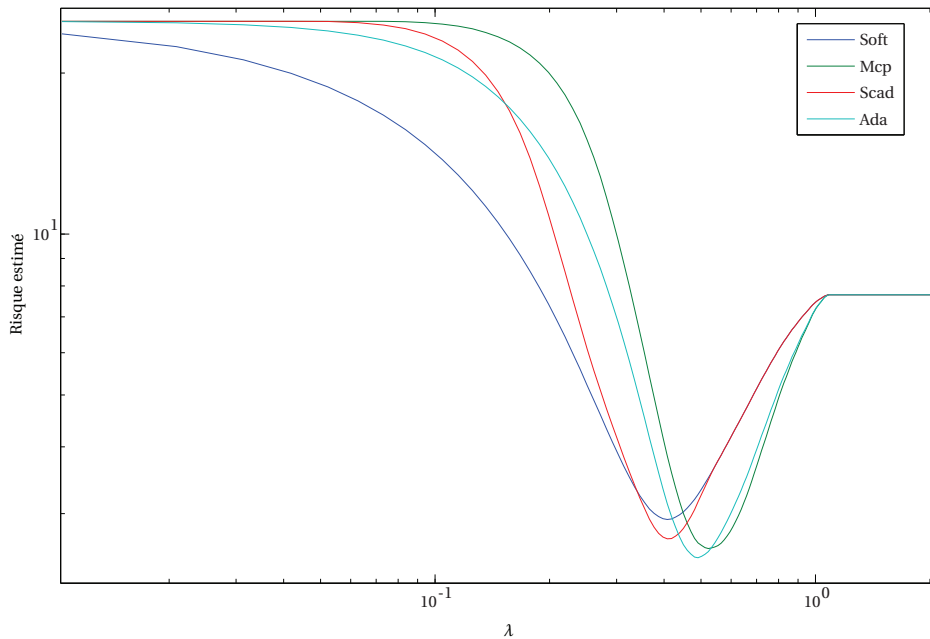


FIGURE 5.3 – Comparaison du risque estimé en fonction de λ pour les différentes fonctions de seuillage proposées (SNR = 0,2 et rang(M) = 20)

de rang de M , il présente le risque estimé optimal moyen de chaque méthode ainsi que le nombre moyen de caractéristiques sélectionnées pour cet optimal (ce qui correspond au nombre de termes $f(\sigma_i) > 0$).

On constate tout d'abord que le SOFT et le SCAD ont tendance à obtenir un risque relatif plus faible lorsque le bruit augmente. En effet pour une marge de bruit de $\frac{1}{10}$, et un rang supérieur à 60, le SOFT obtient le risque le plus petit (notons que le SCAD obtient les mêmes performances dans ces conditions). Cependant, on remarque que le modèle sélectionné par le soft n'est pas très bon, celui-ci sélectionne systématiquement beaucoup trop de variables, et ce quelque soit la marge bruit ou le rang de la matrice. Tandis que le risque du MCP et de l'ADALASSO sont dans tous les autres cas inférieur au SOFT et très souvent inférieur au SCAD (on note que le risque pour l'ADALASSO reste légèrement inférieur au MCP). Mais là où le SOFT et le SCAD avaient tendance à surestimer le nombre de variables à sélectionner, l'ADALASSO et le MCP sont beaucoup plus proches de la réalité (avec le MCP qui sous-estime lorsque le signal est bruité et l'ADALASSO qui surestime mais moins que les deux autres fonctions).

En conclusion, si l'on cherche la méthode qui nous donnera le risque minimal dans la plupart des cas, on choisira d'utiliser une fonction de type ADALASSO. Si l'on cherche la méthode qui nous donnera la meilleure approximation du rang du modèle sous-jacent, on choisira d'utiliser une fonction de type MCP.

Dans un problème de factorisation (de grande taille), la fonction de seuillage traditionnellement utilisée est la fonction de seuillage dur (*hard thresholding*) et la sélection de modèle (le choix du nombre de facteurs latents pour la factorisation) est choisie généralement graphiquement grâce à la méthode du coude. Cette méthode d'estimation du nombre de facteurs est assez correcte lorsque le bruit est faible, mais sur de vraies données de grande dimension, le coude n'est pas toujours visible et

rang	SNR	0,1		0,2		0,5		1		2	
		Risque	k	Risque	k	Risque	k	Risque	k	Risque	k
20	SOFT	5.819 ± 0.094	37.59 ± 0.56	2.898 ± 0.021	61.28 ± 0.51	0.7396 ± 0.0030	80.68 ± 0.18	0.2198 ± 0.00072	94.95 ± 0.19	0.062331 ± 0.00019	70.75 ± 0.17
	MCP	5.628 ± 0.1	19.54 ± 0.3	2.482 ± 0.022	20.19 ± 0.31	0.5221 ± 0.0031	22.97 ± 0.15	0.1361 ± 0.00083	20.34 ± 0.097	0.03423 ± 0.0002	20 ± 0
	SCAD	5.819 ± 0.094	37.59 ± 0.56	2.668 ± 0.021	61.56 ± 0.46	0.5471 ± 0.0031	52.62 ± 1	0.1407 ± 0.0008	48.52 ± 0.19	0.0352 ± 0.0002	31.31 ± 0.16
	ADALASSO	5.641 ± 0.099	21.03 ± 0.36	2.411 ± 0.022	29.73 ± 0.33	0.5049 ± 0.0031	29.39 ± 0.19	0.1341 ± 0.00083	30.77 ± 0.18	0.03428 ± 0.0002	31.31 ± 0.16
40	SOFT	11.12 ± 0.081	58.18 ± 0.5	5.336 ± 0.02	91.45 ± 0.42	1.318 ± 0.0027	114.8 ± 0.77	0.3848 ± 0.00068	128 ± 0.18	0.1048 ± 0.00017	128.3 ± 0.16
	MCP	11.15 ± 0.085	30.38 ± 0.28	5.002 ± 0.022	35.91 ± 0.29	1.028 ± 0.0032	43.58 ± 0.15	0.2657 ± 0.00077	45.24 ± 0.16	0.06648 ± 0.0002	40 ± 0
	SCAD	11.12 ± 0.081	58.18 ± 0.5	4.997 ± 0.02	80.32 ± 0.31	1.052 ± 0.0029	78.16 ± 0.19	0.2725 ± 0.00075	66.17 ± 1.8	0.06751 ± 0.0002	45.89 ± 0.16
	ADALASSO	11.08 ± 0.085	34 ± 0.29	4.703 ± 0.022	49.32 ± 0.35	0.9845 ± 0.0031	52.25 ± 0.73	0.2612 ± 0.00076	45.52 ± 0.45	0.06606 ± 0.0002	45.89 ± 0.16
60	SOFT	15.93 ± 0.079	74.82 ± 0.44	7.474 ± 0.015	115.4 ± 0.37	1.796 ± 0.0022	144.7 ± 0.16	0.519 ± 0.00055	158.5 ± 0.18	0.1426 ± 0.00013	182 ± 0.16
	MCP	16.43 ± 0.085	39.68 ± 0.28	7.488 ± 0.017	51.22 ± 0.27	1.5 ± 0.0029	63.83 ± 0.16	0.3863 ± 0.00068	61.31 ± 0.14	0.09656 ± 0.00017	60 ± 0
	SCAD	15.93 ± 0.079	74.82 ± 0.44	7.247 ± 0.016	92.77 ± 0.37	1.516 ± 0.0027	98.07 ± 0.18	0.3924 ± 0.00067	90.3 ± 0.14	0.09755 ± 0.00017	90.08 ± 1.2
	ADALASSO	16.18 ± 0.085	45.35 ± 0.36	6.856 ± 0.017	66.59 ± 0.3	1.426 ± 0.0028	74.83 ± 0.83	0.3781 ± 0.00068	73.11 ± 0.15	0.09613 ± 0.00018	61.74 ± 0.12
80	SOFT	20.64 ± 0.083	89.64 ± 0.45	9.44 ± 0.017	134.7 ± 0.49	2.214 ± 0.0019	162.8 ± 0.61	0.6331 ± 0.00038	185.2 ± 0.15	0.1686 ± 0.00009.9	185.7 ± 0.18
	MCP	21.82 ± 0.093	48.11 ± 0.27	9.987 ± 0.02	66.66 ± 0.27	1.949 ± 0.0027	84.06 ± 0.15	0.5001 ± 0.0006	81.4 ± 0.56	0.1245 ± 0.00015	80 ± 0
	SCAD	20.64 ± 0.083	89.64 ± 0.45	9.525 ± 0.018	105.9 ± 0.41	1.957 ± 0.0025	117.8 ± 0.17	0.5066 ± 0.00055	113.9 ± 1.8	0.125 ± 0.00014	102.3 ± 0.18
	ADALASSO	21.31 ± 0.09	55.49 ± 0.33	8.955 ± 0.02	83.44 ± 0.3	1.846 ± 0.0027	94.24 ± 0.8	0.4897 ± 0.00059	87.41 ± 0.58	0.1243 ± 0.00015	80.03 ± 0.034
100	SOFT	25.15 ± 0.07	103.3 ± 0.39	11.23 ± 0.014	151.1 ± 0.19	2.574 ± 0.0016	187.1 ± 0.31	0.727 ± 0.00039	189 ± 0.16	0.1941 ± 0.00009.9	189.2 ± 0.17
	MCP	27.24 ± 0.083	55.42 ± 0.27	12.43 ± 0.019	82.68 ± 0.25	2.369 ± 0.0023	103.8 ± 0.18	0.6039 ± 0.00057	102 ± 0.13	0.1507 ± 0.00015	100 ± 0
	SCAD	25.16 ± 0.071	102.7 ± 0.35	11.78 ± 0.017	118.5 ± 0.25	2.368 ± 0.0021	136.7 ± 0.18	0.6089 ± 0.00055	130.1 ± 0.18	0.1509 ± 0.00014	114.5 ± 0.18
	ADALASSO	26.37 ± 0.079	65.54 ± 0.36	10.95 ± 0.018	99.22 ± 0.31	2.238 ± 0.0023	111.6 ± 0.57	0.5906 ± 0.00056	113.6 ± 0.14	0.15 ± 0.00015	114.5 ± 0.18

TABLE 5.1 – Estimation du risque optimal par rapport à la marge du bruit et au rang de M pour les différentes fonctions de seuillage proposées.

l'on ne pourra pas estimer correctement le nombre de variables représentatives du modèle. Notons également que cette méthode n'est pas automatique. De plus le *hard thresholding* ne donne pas une bonne estimation du risque, notamment parce qu'il est discontinu et non différentiable. En utilisant une fonction de seuillage de type MCP, le rang réel du modèle semble être plutôt bien approximé (quelque soit le niveau de bruit) à travers le processus de minimisation de l'estimateur du risque. Dans l'éventualité où une telle méthode passerait l'échelle, celle-ci fournirait de meilleures estimations du modèle sous-jacent que les méthodes actuellement utilisées (qui ne sont pas des méthodes de sélection de modèle à proprement parler). Nous nous intéressons donc au passage à l'échelle de ces méthodes de sélection de modèle pour la factorisation dans le paragraphe suivant.

5.3.2 Passage à l'échelle

Le calcul de la divergence de f en Y est très intéressante car elle se résume à une somme de termes dépendants des valeurs singulières de la matrice Y . Cependant ce calcul nécessite la connaissance de l'ensemble des valeurs singulières de Y . Lorsque l'on travaille sur des matrices de taille 500×500 (en traitement de l'image par exemple), le calcul de la décomposition en valeurs singulières complète ne pose pas trop de problèmes. Malheureusement, dans des problèmes de recommandation, la dimension de Y peut rapidement dépasser la centaine de milliers et le calcul de l'ensemble des valeurs est impossible.

Il nous apparaît donc intéressant de proposer une approximation de cette divergence ne nécessitant pas la connaissance de toutes les valeurs singulières afin que ce modèle puisse être utilisé dans un domaine d'application qui nous intéresse, où la dimensionnalité peut s'avérer être un obstacle.

5.3.2.1 Approximation de la divergence

Nous n'allons pas, à proprement parler, approcher le calcul de la divergence. Nous prenons ici le parti d'estimer les valeurs singulières que nous ne pouvons nous permettre de calculer. Ainsi, soit par extrapolation, soit par régression, nous proposons d'estimer, à partir d'un certain nombre de valeurs singulières connues, le reste du spectre d'une matrice. Nous proposons deux méthodes d'approximation. La première est une extrapolation de la distribution des valeurs propres. La seconde consiste en une interpolation sur le spectre des valeurs singulières.

Extrapolation selon Marchenko-Pastur Il existe une grande littérature concernant les loi de distribution des valeurs propres de matrices aléatoires (Mehta 2004, Bai & Silverstein 2010), nous nous sommes intéressés à une loi en particulier qui est celle de Marchenko-Pastur. L'idée ici est d'approximer la loi de distribution de l'ensemble connu des valeurs propres d'une matrice et ensuite d'extrapoler cette loi à l'ensemble du spectre des valeurs propres afin de pouvoir estimer les valeurs non connues. Nous considérons dorénavant les valeurs singulières σ_i de la matrice X comme étant des variables aléatoires, réalisation d'une loi de probabilité. Pour le choix de cette loi, nous nous basons sur les travaux de [Marchenko & Pastur 1967] et [Wachter 1978] desquels ressort le théorème 5.3.

Théorème 5.3. Soit $X \in \mathcal{L}(p, n)$ une matrice aléatoire centrée. Les valeurs propres λ_i de la matrice $\frac{1}{n} X^\top X$ suivent (si on les considère comme variables aléatoires) une loi de Marchenko-Pastur de paramètre $c > 0$ dont la fonction de densité de probabilité vaut :

$$P_c(x) = \begin{cases} \frac{1}{2\pi xc} \sqrt{(x-a)(b-x)} & \text{si } a \leq x \leq b, \\ 0 & \text{sinon.} \end{cases}$$

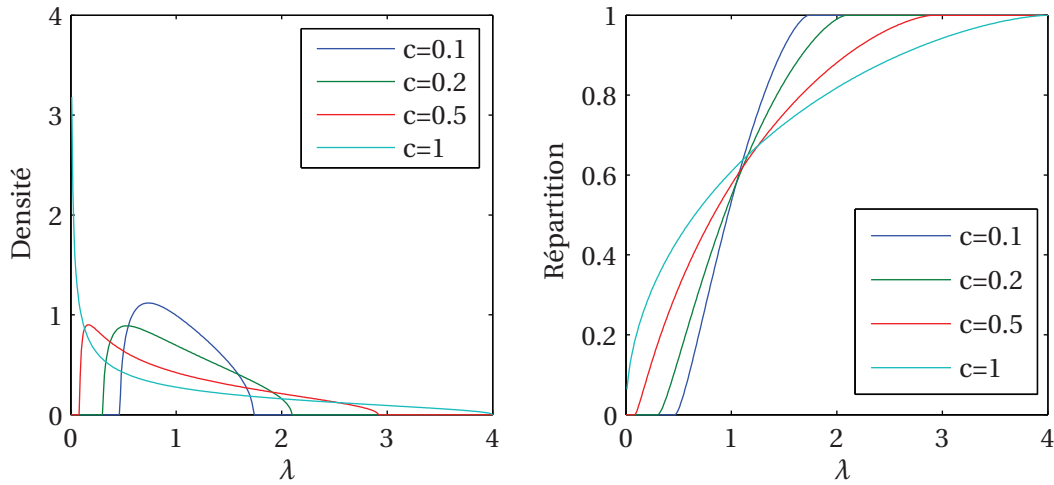


FIGURE 5.4 – Exemples de fonctions de densité de probabilité et de fonctions de répartition de la loi de Marchenko-Pastur pour différentes valeurs du paramètre c .

où $a = (1 - \sqrt{c})^2$ et $b = (1 + \sqrt{c})^2$.

La figure 5.4 représente les fonctions de densité de probabilité et de répartition de la loi de Marchenko-Pastur pour différents paramètres c .

À partir des valeurs singulières σ_i de la matrice X , nous pouvons obtenir les valeurs propres λ_i de la matrice $\frac{1}{n}X^\top X$, grâce à la formule suivante : $\lambda_i = \frac{\sigma_i^2}{n}$. On suppose donc connaître l'ensemble des valeurs λ_i pour $i \in \{1, \dots, i_{\max}\}$.

Afin d'approximer la loi que suivent ces λ_i , nous allons chercher la fonction de répartition d'une loi de Marchenko-Pastur qui soit la plus proche possible de la fonction de répartition empirique (calculée à partir des variables λ_i connues).

La fonction de répartition d'une loi de Marchenko-Pastur s'écrit :

$$F_c(x) = \frac{1}{2\pi c} \left(\sqrt{-x^2 + (a+b)x - ab} - \frac{a+b}{2} \left[\arcsin\left(\frac{-2x+a+b}{b-a}\right) + \frac{\pi}{2} \right] - \sqrt{ab} \left[\arcsin\left(\frac{(a+b)x-2ab}{x(b-a)}\right) - \frac{\pi}{2} \right] \right) \quad (5.22)$$

Nous détaillons le calcul de l'expression de la fonction de répartition d'une loi de Marchenko-Pastur en annexe B.3.

Nous estimons la fonction de répartition empirique $\bar{F}$ à partir des valeurs propres λ_i connues. En accord avec le théorème 5.3, nous cherchons le paramètre c qui minimise le coût suivant :

$$\min_c \sum_{i=1}^{i_{\max}} (\bar{F}(\lambda_i) - F_c(\lambda_i))^2 \quad (5.23)$$

Une fois le $\hat{c}$ optimal trouvé en minimisant le coût 5.23 (à l'aide d'un grid search sur un ensemble de valeurs de c), nous obtenons une loi dont la fonction de répartition $F_{\hat{c}}$ qui approxime $\bar{F}$ sur l'intervalle $[\lambda_{i_{\max}}, \lambda_i]$. Nous extrapolons alors cette répartition à l'ensemble du spectre. La figure 5.5 illustre un exemple d'approximation de la fonction de répartition des λ_i .

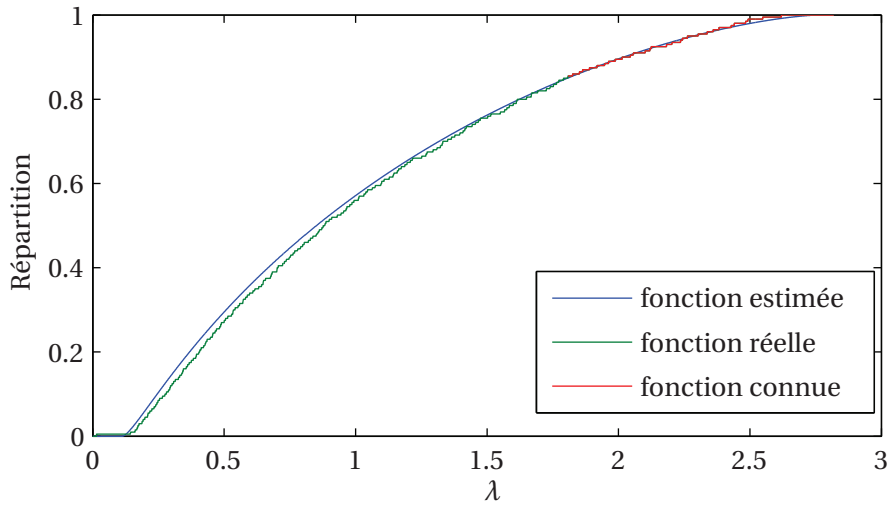


FIGURE 5.5 – Exemple d’approximation de la répartition des valeurs propres d’une matrice à partir d’un ensemble de valeurs propres connues (la courbe rouge représente la fonction de répartition empirique des valeurs propres connues, la courbe bleue représente la fonction de répartition de la loi de Marchenko-Pastur estimée à partir de la fonction empirique. Enfin, à titre de comparaison, la courbe verte représente la fonction de répartition empirique de l’ensemble du spectre des valeurs propres).

Une fois cette fonction de répartition $F_{\hat{\epsilon}}$ connue, nous proposons le modèle suivant pour approximer la divergence :

1. Nous tirons aléatoirement selon la loi de probabilité de répartition $F_{\hat{\epsilon}}$ autant de valeurs que de valeurs propres inconnues ($p - i_{max}$). Ce tirage aléatoire ne s’effectue que pour des valeurs inférieures à $\lambda_{i_{max}}$ (c’est-à-dire sur la partie extrapolée de la fonction de répartition).
2. Nous calculons la divergence pour ce tirage et pour une fonction de seuillage donnée.
3. Nous répétons 1 et 2 cent fois.

La divergence ainsi calculée est une statistique suivant une loi de probabilité. Nous calculons sa moyenne empirique pour obtenir une approximation de la divergence du modèle. La figure 5.6 représente un exemple d’ensemble de tirages aléatoires de valeurs propres.

Interpolation linéaire Nous mettons en place cette méthode d’approximation de la divergence à titre de *baseline*. L’idée ici est particulièrement simple, il s’agit d’effectuer une interpolation linéaire du spectre des valeurs singulières entre le point $(i_{max}; \sigma_{i_{max}})$ et le point $(p; \sigma_p)$, où i_{max} est l’indice de la dernière plus grande valeur singulière connue. Obtenir la plus petite valeur singulière (σ_p) est une tâche simple et rapide.

5.3.2.2 Protocole expérimental

Nous allons évaluer expérimentalement l’intérêt de ces méthodes. Même si nous aimerions pouvoir tester ces méthodes sur des problèmes de grande taille, nous ne pourrions pas comparer leur performance à la réalité, car comme précisé plus haut il nous serait impossible de calculer l’ensemble des valeurs singulières de la matrice X .

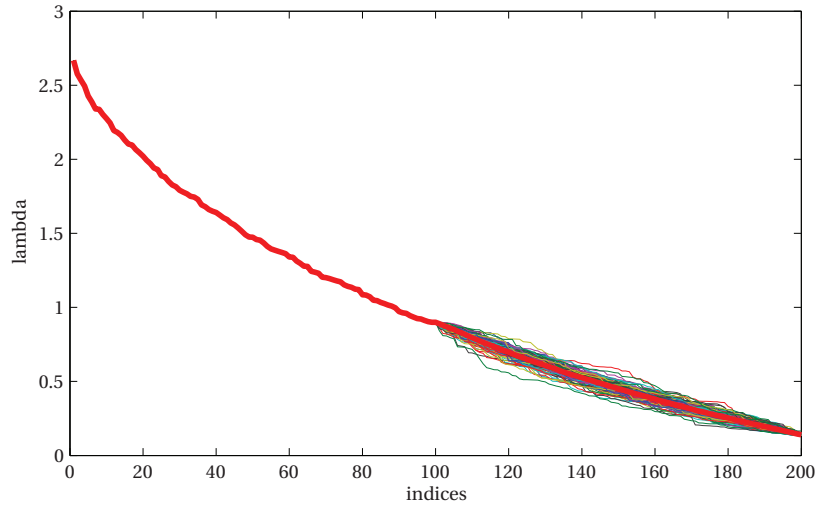


FIGURE 5.6 – Exemple d'ensemble de tirage de valeurs propres

Nous allons devoir tester ces méthodes sur des dimensions raisonnables pour pouvoir les comparer à la réalité. Prenons donc à nouveau $n = 500$ et $p = 200$.

Nous testerons les méthodes pour un ensemble de matrices M générées aléatoirement suivant une loi normale centrée, pour des matrices de bruit R de variances différentes (avec une marge de bruit variant entre 0, 1 et 2). Nous supposons ici aussi que la variance du bruit est connue, comme expliqué précédemment nous nous intéresserons à l'estimation de la variance dans le paragraphe 5.3.3.

Pour chacun des modèles pour lesquels nous testons nos méthodes d'approximation, nous calculons la divergence pour une fonction de seuillage donnée (à savoir la MCP) et pour un ensemble de valeurs de $\lambda \in \mathcal{L}_\lambda$ (le seuil de la fonction de seuillage). Nous calculons ensuite l'erreur quadratique moyenne (MSE) entre ses divergences approximées et les vraies divergences, obtenue par la formule :

$$MSE = \frac{1}{|\mathcal{L}_\lambda|} \sum_{\lambda \in \mathcal{L}_\lambda} (div_Y(f_\lambda(\Sigma)) - div_Y(f_\lambda(\hat{\Sigma})))^2$$

où Σ le spectre réel des valeurs singulières et $\hat{\Sigma}$ désigne l'approximation du spectre des valeurs singulières (soit par interpolation linéaire, soit par extrapolation selon Marchenko-Pastur). Nous répétons l'expérience 100 fois.

5.3.2.3 Résultats expérimentaux

La figure 5.7 représente les divergences en fonction de λ pour les différentes approximations et pour deux valeurs de marge de bruit ($snr = 0, 1$ ou $snr = 2$). Le bruit semble avoir une influence sur la qualité d'approximation de chacune des méthodes proposées. En effet pour un bruit élevé, l'interpolation linéaire du spectre des valeurs singulières semble permettre une bonne approximation de la divergence, tandis que dans une situation peu bruitée, c'est plutôt l'extrapolation selon Marchenko-Pastur qui permet une approximation de la divergence proche de la réalité.

Le figure 5.8 représente l'erreur quadratique moyenne entre nos méthodes d'approximation et la divergence réelle sur les 100 expériences réalisées. La tendance de chacune des méthodes est clai-

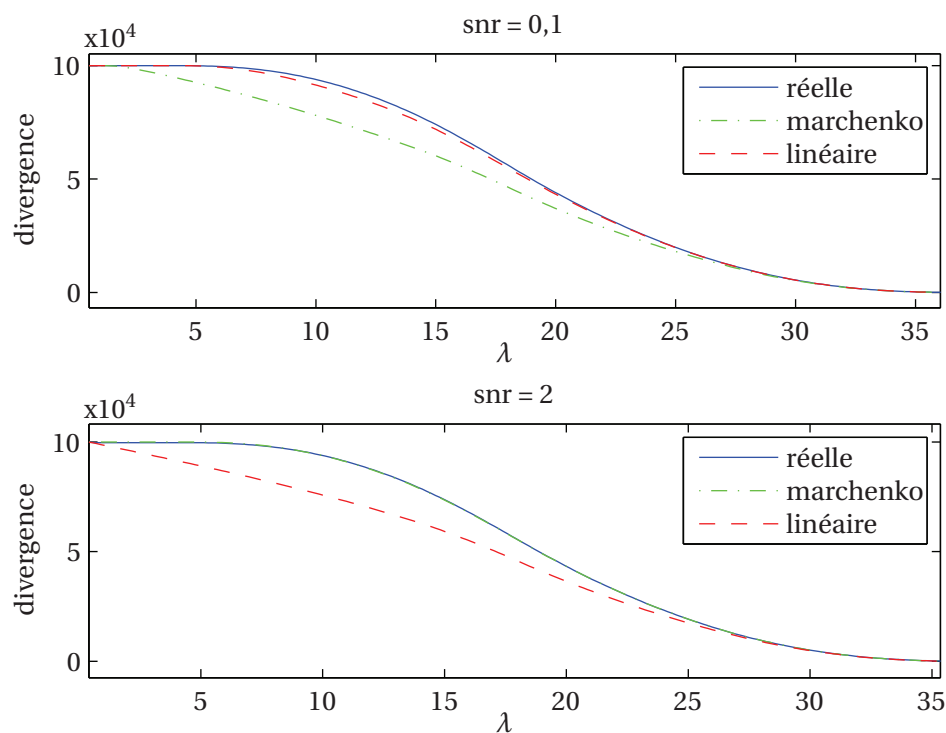
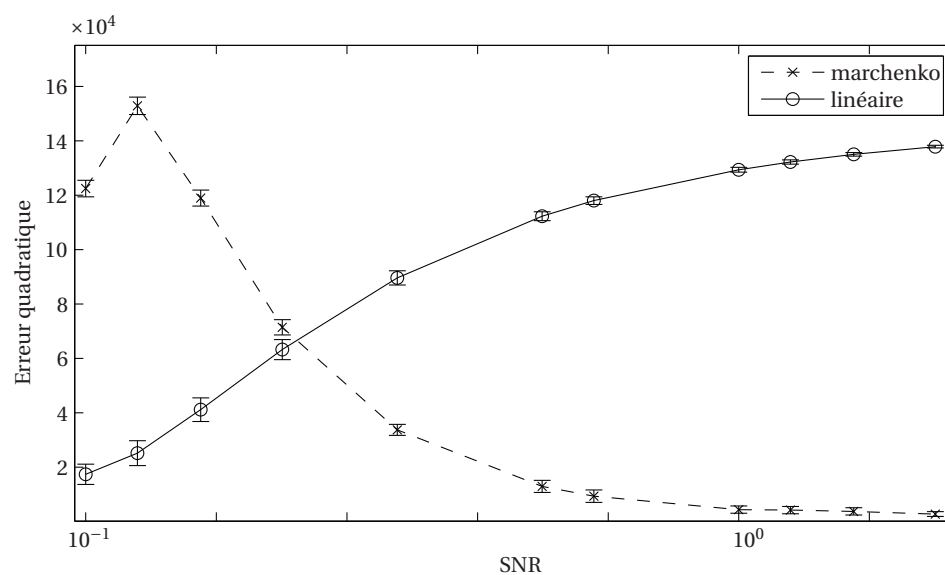
FIGURE 5.7 – Divergences (réelle et approximées) en fonction du paramètre λ 

FIGURE 5.8 – Erreur quadratique moyenne d'approximation de la divergence pour les deux types d'approches proposées en fonction de la marge de bruit.

rement visible : pour une faible marge de bruit, l'interpolation linéaire fournit la meilleure approximation de la divergence, et pour une marge de bruit élevée, c'est l'approximation par Marchenko-Pastur qui permet la meilleure approximation. Dans les deux cas de figure cette différence est significative. Ainsi le choix de la méthode d'approximation la plus adaptée dépendra du problème, et d'une connaissance a priori (éventuellement) sur la marge de bruit du modèle.

On peut cependant noter un important défaut du modèle proposé. En effet, l'approximation selon Marchenko nécessite un centrage des données qui entraîne la perte de la parcimonie des données, ce qui peut être très pénalisant dans un problème de grande taille. L'approche d'interpolation linéaire n'a pas cet inconvénient et donne de bons résultats sur des problèmes bruités. Bien qu'étant une approche plus simpliste, elle semble plus adaptée aux problèmes qui nous intéressent.

5.3.3 Estimation de la variance

Dans un problème idéal, la variance de la statistique du modèle est connue et il est simple de calculer l'estimateur du risque (qui dépend de cette variance). Malheureusement, cette variance n'est que très rarement connue et il faudra donc l'estimer.

Ce qui est traditionnellement fait est, dans un premier temps, de supposer que l'on connaît la variance du bruit et l'on minimise le risque avec cette variance. Une fois le λ_{opt} qui minimise ce risque trouvé, on calcule empiriquement la variance pour le modèle $\hat{M}^{(\lambda_{opt})}$. Cette variance empirique est $\hat{\sigma}^2 = \frac{\|Y - \hat{M}^{(\lambda_{opt})}\|^2}{np}$ si l'on ne souhaite pas nécessairement avoir un estimateur sans biais de la variance. Si l'on souhaite un estimateur sans biais, on choisira plutôt $\hat{\sigma}^2 = \frac{\|Y - \hat{M}^{(\lambda_{opt})}\|^2}{np - \widehat{df}}$, où $\widehat{df}$ désigne le degré de liberté (discuté dans [Ye 1998]), ce qui dans nous cas correspond à la divergence $div_Y(f)$.

Le problème d'un tel modèle est que la variance initialement choisie n'est pas nécessairement celle qui est la bonne, puisqu'on ne fait que supposer la connaître.

Heuristique 5.1. Soit $\tilde{\sigma}^2$ une valeur de variance fixée. Si la valeur de la variance calculée en sortie du modèle est égale (à un δ près) à $\tilde{\sigma}^2$, on dira que cette variance est une bonne approximation de la variance du bruit du modèle.

À partir de cette heuristique nous formalisons le problème suivant :

$$\min_{\tilde{\sigma}} |\tilde{\sigma}^2 - \hat{\sigma}^2| \quad (5.24)$$

Nous cherchons la variance $\tilde{\sigma}^2$ fixée en entrée du système qui soit la plus proche possible de la variance $\hat{\sigma}^2$ estimée en sortie du système.

Nous ne prétendons pas que ceci donnera une bonne approximation de la valeur de la variance, nous voulons simplement vérifier si cette heuristique peut être validée expérimentalement.

5.3.3.1 Protocole expérimental

Nous allons essayer d'évaluer expérimentalement la validité de l'heuristique 5.1. Nous nous plaçons à nouveau dans un cadre restreint en fixant les dimensions de la matrice M à $n = 500$ et $p = 200$. Nous fixons également $rang(M) = 20$.

L'idée est de tester si l'on trouve une variance $\tilde{\sigma}^2$, minimisant le problème 5.24, suffisamment proche de la vraie variance σ^2 , pour un ensemble de matrices Y données ayant des marges de bruit différentes ($SNR \in \{\frac{1}{10}, \frac{1}{5}, \frac{1}{2}, 1, 2\}$). Nous allons donc comparer la variance obtenue à la variance réelle et également vérifier si le modèle sélectionné avec cette estimation de la variance est proche du vrai

SNR	σ^2	$\min \tilde{\sigma}^2 - \hat{\sigma}^2 $	$\hat{\sigma}^2$	$\text{rang}(\widehat{M})$
0,1	1.10^{-3}	$2,60.10^{-6} \pm 1,50.10^{-6}$	$1,02.10^{-3} \pm 6,63.10^{-06}$	$17,49 \pm 1,69$
0,2	$2,5.10^{-4}$	$7,52.10^{-7} \pm 4,17.10^{-7}$	$2,56.10^{-4} \pm 1,73.10^{-06}$	$18,83 \pm 1,50$
0,5	4.10^{-5}	$1,13.10^{-7} \pm 6,49.10^{-8}$	$4,04.10^{-05} \pm 2,56.10^{-07}$	$22,37 \pm 1,55$
1,0	1.10^{-5}	$2,63.10^{-8} \pm 1,62.10^{-8}$	$1,00.10^{-05} \pm 6,77.10^{-08}$	$21,96 \pm 2,26$
2,0	$2,5.10^{-6}$	$7,25.10^{-9} \pm 4,53.10^{-9}$	$2,50.10^{-06} \pm 1,67.10^{-08}$	20 ± 0

TABLE 5.2 – Résultats sur l'estimation de la variance (avec des intervalles de confiance à 95%).

modèle. Nous utilisons une fonction de seuillage *MCP* qui nous donne la meilleure estimation du nombre de paramètre à sélectionner.

5.3.3.2 Quelques résultats

La table 5.2 présente les résultats obtenus. On peut noter que la variance obtenue grâce à l'heuristique proposée est une bonne estimation de la variance. De plus, le modèle sélectionné avec cette estimation de la variance est très similaire à ce qu'on pouvait obtenir lorsque la variance était connue (cf. table 5.1).

Cependant, nous souhaiterions pointer du doigt un inconvénient de cette méthode, ou du moins montrer qu'il faut être vigilant quant à son utilisation. La figure 5.9 illustre ce point que nous voulons souligner. Cette figure représente les courbes des $\tilde{\sigma}^2$, $\hat{\sigma}^2$ et de l'erreur $|\tilde{\sigma}^2 - \hat{\sigma}^2|$ en fonction de la variance en entrée $\tilde{\sigma}^2$. Nous pouvons constater graphiquement que les courbes se croisent deux fois et qu'il y a donc 2 minima locaux. De plus les courbes convergent en 0. Nous avons donc (dans ce cas) trois minima. Le minimum proche de 10^{-3} (qui est la vraie valeur de la variance) est celui qu'il nous faudrait sélectionner, mais nous n'avons pas de moyen pour le moment pour déterminer quel minimum est le bon (même si l'on choisit d'écarter d'office le minimum en 0). Nous avons néanmoins remarqué (graphiquement) que le minimum le plus grand semble être à chaque fois celui le plus proche de la variance réelle. Ceci ne représente en rien une règle pour le choix du minimum, mais elle semble convenir dans le problème expérimental que nous nous sommes fixés.

Afin de vérifier si l'ensemble des voies d'amélioration proposées peuvent fonctionner les unes avec les autres, nous avons estimé le risque pour les différents cas de figure possibles :

- estimation de risque avec divergence et variance connue,
- estimation de risque avec estimation de la divergence et variance connue,
- estimation de risque avec estimation de la variance et divergence connue,
- estimation de risque avec estimation de la divergence et de la variance.

La figure 5.10 représente les courbes de risque estimé pour ces différents cas pour un exemple donné. On remarque qu'il y a un léger biais dans l'estimation du risque lorsque l'on estime la variance tandis que l'approximation de la divergence n'influe que peu sur l'estimation du risque. Des variations de la valeur de la variance semble avoir plus d'impact sur le risque que des variations de la valeur de la divergence. On note cependant que quelque soit le cas de figure, le minimum semble être atteint à peu près pour un même λ , ce qui signifie que même si l'on ne connaît ni la variance ni l'ensemble du spectre des valeurs singulières il est toujours possible de sélectionner le bon modèle.

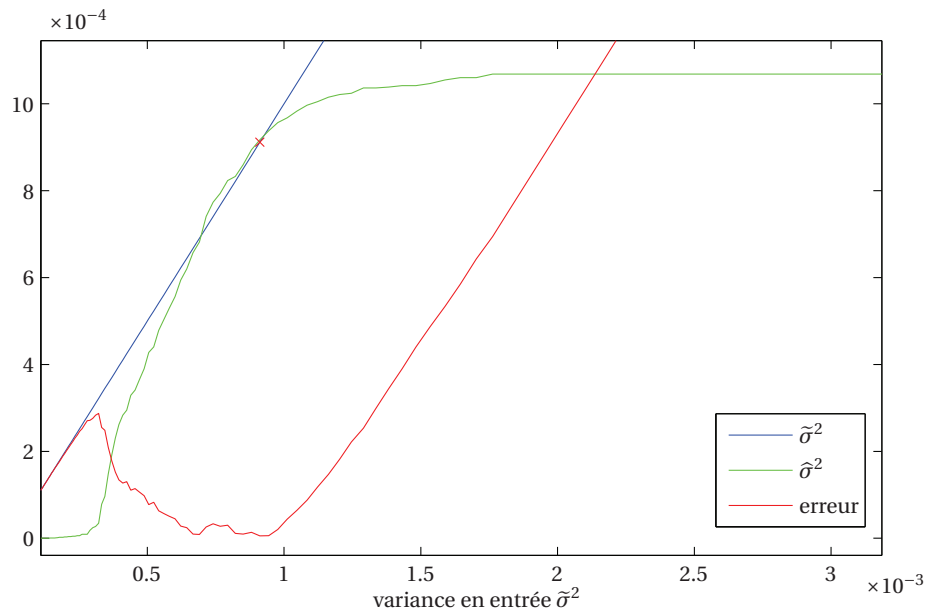


FIGURE 5.9 – Variance estimée $\hat{\sigma}^2$ en sortie du modèle en fonction de la variance $\tilde{\sigma}^2$ en entrée ($SNR = 0,1$)

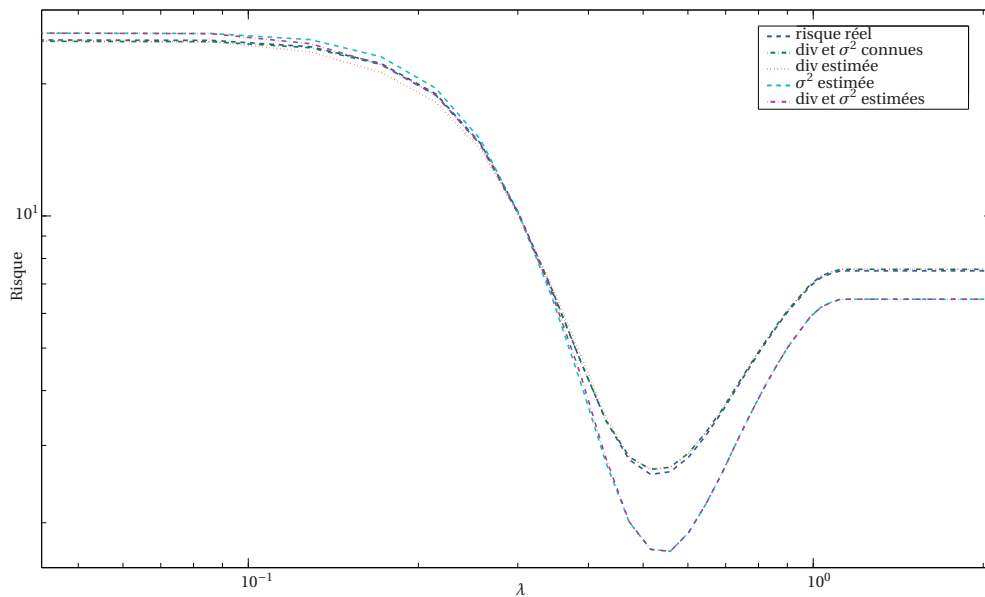


FIGURE 5.10 – Comparaison d'estimations de risque en fonction de λ pour différents cas de figure suivant si l'on estime ou non la divergence et la variance ($SNR = 0,2$ et $\text{rang}(M) = 20$)

5.4 Discussion

Le but de l'étude présentée dans ce chapitre n'était pas d'aboutir à une méthode universelle de sélection de modèle automatique pour la factorisation dans le domaine de la recommandation. Nous proposons simplement ici un moyen d'adapter la sélection de modèle à une problématique de recommandation, ce qui a été peu proposé auparavant. De plus, nous n'adaptions la sélection de modèle que pour un problème précis, à savoir le problème de décomposition partielle en valeurs singulières.

Même si la méthode de factorisation que nous utilisons pour la sélection de modèle peut être jugée comme n'étant pas des plus performantes, ce qui peut être en revanche intéressant serait de se servir du rang du modèle sélectionné pour l'intégrer dans une méthode de factorisation jugée plus performante que la décomposition en valeurs singulières ne l'est dans le domaine de la recommandation. Nous pourrions en effet nous en servir pour connaître le rang de la matrice à factoriser en prétraitement de la factorisation.

L'idée serait d'utiliser une fonction de seuillage sûrement de type MCP pour la sélection de modèle étant donné que, pour cette fonction, le nombre de variables latentes sélectionnées semble se rapprocher le plus du rang réel de la matrice M , tout en utilisant notre approximation de la divergence et de la variance (si ces approximations sont nécessaires pour le problème donné), afin d'estimer la rang $\hat{k}$ de la matrice M (directement corrélé à $\hat{\lambda}$ et d'utiliser ce $\hat{k}$ pour le nombre de variables latentes calculées pour la méthode de factorisation qui convient au problème posé).

Nous aimerions également, à partir d'un problème de factorisation tensorielle, faire de la sélection de modèle. Soit $Y \in \mathbb{R}^{n \times p \times m}$, on pose le modèle suivant : $Y = M + \epsilon$. on cherche la fonction spectrale f sur tenseurs qui estime le modèle ($f(Y) = M$). Soit $Y_{ijk} = \sum_{p=1}^d \lambda_l U_{ip} V_{jp} W_{kp}$ la décomposition tensorielle CP. On a $f(Y_{ijk}) = \sum_{p=1}^d f(\lambda_l) U_{ip} V_{jp} W_{kp}$. En partant de la formalisation du problème d'ordre deux, nous voudrions adapter la sélection de modèle proposée par [Candès *et al.* 2012] au problème d'ordre trois dont nous venons d'explicitier le modèle, et éventuellement le généraliser à un problème d'ordre o quelconque.

Conclusion

La science consiste à faire ce qu'on fait en sachant et en disant que c'est tout ce qu'on peut faire, en énonçant les limites de la validité de ce qu'on fait.

– Pierre Bourdieu

Dans le vaste domaine de la factorisation matricielle dans la recommandation, nous avons présenté dans ce manuscrit certains aspects qui nous semblaient importants et nous avons tenté d'apporter notre contribution à ces aspects. Nous allons dans ce chapitre faire un bilan critique des différentes contributions proposées et présenter quelques perspectives quant aux travaux futurs envisagés.

Critiques et limites des contributions

Contextualisons avec modération

Dans le cas d'étude présenté dans le chapitre 3, nous avons pointé du doigt l'intérêt évident d'une contextualisation. Nous avons également vu que cette tâche n'est pas nécessairement aisée et que l'on peut se retrouver avec des performances de prédiction amoindries si l'on se place dans un contexte peu pertinent pour le problème. Cette conclusion est également observable à travers les résultats d'expérience sur les données *Tuenti* du chapitre 4. En effet, on constate que certaines méthodes tirant avantage des informations du réseau social ont des performances inférieures à la méthode ne prenant pas en compte ces données sociales, ce qui n'est pas le cas sur les données *Epinions*. Ainsi, suivant le problème (et la méthode choisie pour le résoudre) et les données dont nous disposons, un choix de contextualisation peut être (ou ne pas être) pertinent.

Il n'existe malheureusement pas de méthode pour trouver automatiquement la contextualisation la mieux adaptée aux données et au problème posé. Nous ne pouvons qu'émettre des hypothèses en proposant d'éventuelles contextualisations et tenter de les valider expérimentalement ou nous pouvons faire appel aux experts du domaine d'application et leur connaissance a priori. Il n'est pas facile non plus de savoir quelle approche et quelle méthode de résolution du problème sera la plus adaptée, nous avons simplement remarqué que les méthodes de factorisation semblaient avoir une meilleure stabilité et être beaucoup moins dépendantes du bruit des données que les autres méthodes proposées.

Ayons confiance et recommandons

Nous avons montré expérimentalement, à travers deux jeux de données, que l'approche proposée dans le chapitre 4 était intéressante en terme de performance de prédiction. En effet, apprendre

les variables latentes de la factorisation simultanément au graphe de confiance semble être une approche pertinente du problème de prédiction de préférences dans un cadre social. L'approche proposée est moins dépendante de la qualité du graphe social disponible puisqu'elle l'adapte et le transforme en un graphe qui reflète une réelle confiance entre les utilisateurs. Notons toutefois que le gain de performance apporté par cette approche, par rapport à une méthode qui ne prend pas en compte les données sociales (comme [Hu *et al.* 2008]), ne justifie peut être pas nécessairement le coût computationnel supplémentaire dans le cas où le graphe social n'est que très peu informatif (comme c'est le cas dans le réseau *Tuenti*). La justification de ce coût computationnel semble nettement plus visible lorsque l'on dispose d'un graphe de confiance (qui sera d'ailleurs affiner au cours du processus d'optimisation).

Ne dénigrons pas non plus trop les méthodes auxquelles nous nous sommes comparés. Certes sur le problème posé notre approche semble sans équivoque la plus intéressante, notons cependant que nous ne répondons par exemple pas au problème de recommandation d'amis, tandis que l'approche [Yang *et al.* 2011] permet de résoudre ce problème. De plus leur formulation du problème permet la prise en compte d'informations contextuelles supplémentaires et pas nécessairement liées au réseau social.

Factorisons peu, factorisons bien

Nous proposons, dans le chapitre 5, plusieurs voies d'améliorations possibles à la sélection de modèle dans une décomposition en valeurs singulières par minimisation de risque.

Une première contribution propose une adaptation aux problèmes de grande taille (comme les problèmes de factorisation dans la recommandation) d'une méthode permettant de résoudre un problème de sélection de modèle par minimisation de risque. Nous avons montré expérimentalement, pour un problème donné, qu'il était possible d'obtenir une meilleure estimation du rang du modèle en choisissant la bonne fonction de seuillage. En l'occurrence la MCP semble être celle qui sélectionne le plus souvent le meilleur rang, malgré une estimation du risque parfois un peu moins bonne que pour d'autres fonctions de seuillage.

Nous avons également proposé deux approches permettant d'estimer la divergence de la fonction de seuillage lorsque le calcul de celle-ci peut être difficile voire impossible en grande dimension. Nous n'avons malheureusement pas pu déterminer laquelle des deux approches était la plus pertinente. Il apparaît évident que la pertinence de ces approches dépendra du problème et des données. Nous avons constaté que l'extrapolation de Marchenko-Pastur permettait une bonne approximation dans un cas peu bruité, tandis que l'interpolation linéaire se comportait mieux sur un problème plus bruité. N'ayant pas nécessairement de connaissance a priori sur ce bruit, il ne nous est pas vraiment possible de déterminer laquelle des méthodes il faudra utiliser face à un problème donné. Notons enfin que, malgré tout, l'approximation de la divergence semble permettre d'estimer correctement le rang du modèle et qu'il est donc possible, grâce à celle-ci, d'avoir une méthode de sélection de modèle par minimisation de risque qui passe à l'échelle.

Nous proposons une méthode d'approximation de la variance du modèle à travers une heuristique. Cette approximation peut s'avérer intéressante comme les résultats expérimentaux le montrent. Malheureusement, comme nous l'avons souligné, le coût que nous minimisons possèdent plusieurs minima locaux et nous n'avons pas moyen de savoir quel minimum correspond à la meilleure approximation du modèle. Ceci rend son utilisation délicate et nous recommandons d'être prudent. Remarquons qu'une estimation de la variance (même bonne) semble faire varier (et même biaiser) l'estimation du risque, mais sans pour autant modifier grandement l'estimation du rang du

modèle, ce qui est plutôt intéressant.

Perspectives

Plus en profondeur dans la contextualisation

Les approches contextuelles proposées dans le chapitre 3 ne s'inscrivent que dans un paradigme de contextualisation prétraitement. Les approches de modélisation contextuelle semblent plus intéressantes et l'idée serait de repartir du problème posé dans le cas d'étude et de le reformaliser afin d'intégrer directement le contexte dans le modèle. Des méthodes telles que les machines à factorisation (Rendle *et al.* 2011) permettent une formulation simple, rapide et uniformisée des problèmes de factorisation avec information contextuelle. Nous pourrions ainsi comparer différentes nouvelles propositions de contextualisation entre elles et également les comparer avec celles proposées dans le cas d'étude. Ceci nous permettrait notamment de tenter de valider l'hypothèse que la modélisation contextuelle est plus intéressante que la contextualisation par prétraitement.

Nous avons vu les limites du modèle proposé dans le chapitre 4. L'idée principale pour de futurs travaux serait de permettre l'intégration de données contextuelles supplémentaires (position géographique des lieux, types de lieux, informations utilisateurs...). Une approche tensorielle (ou par machines à factorisation) semble très intéressante pour prendre en compte de nouveaux contextes. Nous envisageons donc de reformuler le problème sous une forme plus adaptée (comme la factorisation tensorielle) pour élargir la contextualisation.

Élargissement ou généralisation des problèmes

L'approche présentée dans le chapitre 4 ne permet pas la recommandation d'amis, or ceci est une fonctionnalité quasi-obligatoire d'un réseau social en ligne. En reformulant le problème nous voudrions construire un modèle qui permet aussi bien la recommandation d'amis que la recommandation d'articles (à l'instar de [Yang *et al.* 2011]).

Dans le chapitre 5, nous supposons que le bruit du modèle suit une loi normale centrée de variance inconnue. Nous avons vu que l'estimation de cette variance peut être compliquée. Nous envisageons de formaliser le problème d'estimation de risque pour un bruit gaussien à variance inconnue pour notre problème de décomposition (comme cela est présenté pour un problème de régression linéaire dans [Boisbunon 2013]).

De plus, supposer que le bruit du modèle suit une loi normale n'est pas nécessairement judicieux. En effet, le bruit sur des données réelles ne suit pas forcément (et même très rarement) cette loi. Nous souhaitons reformaliser le problème pour un bruit suivant une loi à symétrie sphérique, ou même, pour gagner en généralisation, suivant une loi elliptique.

Par ailleurs, nous utilisons une estimation sans biais du risque (le *SURE*) or celui-ci n'est pas optimal. Nous envisageons d'appliquer une correction à cet estimateur afin d'obtenir une meilleure estimation du risque.

Enfin, nous nous intéressons à un problème de décomposition en valeurs singulières d'ordre deux. Nous voulons étendre ce problème à un ordre supérieur en reformalisant le problème pour une décomposition tensorielle CP. Le challenge sera ici (tout comme dans les perspectives concernant le bruit) de recalculer la divergence de la fonction de seuillage.

Un cas d'étude de recommandation contextuelle

A Case Study in a Recommender System Based on Purchase Data

Bruno Pradel
Université Paris 6
Paris, France
pradelb@poleia.lip6.fr

Savaneary Sean
KXEN Europe
Suresnes, France
savaneary.sean@kxen.com

Julien Delporte
INSA de Rouen
Saint-Étienne-du-Rouvray, France
julien.delporte@insa-rouen.fr

Sébastien Guérif, Céline Rouveirol
Université Paris-Nord
Villetaneuse, France
surname.name@lipn.univ-paris13.fr

Nicolas Usunier
Université Paris 6
Paris, France
usunier@poleia.lip6.fr

Françoise Fogelman-Soulié
KXEN Europe
Suresnes, France
francoise.souliefogelman@kxen.com

Frédéric Dufau-Joel
La Boîte à Outils
France
frederic-dufau-joel@samse.fr

ABSTRACT

Collaborative filtering has been extensively studied in the context of ratings prediction. However, industrial recommender systems often aim at predicting a few items of immediate interest to the user, typically products that (s)he is likely to buy in the near future. In a collaborative filtering setting, the prediction may be based on the user's purchase history rather than rating information, which may be unreliable or unavailable. In this paper, we present an experimental evaluation of various collaborative filtering algorithms on a real-world dataset of purchase history from customers in a store of a French home improvement and building supplies chain. These experiments are part of the development of a prototype recommender system for salespeople in the store. We show how different settings for training and applying the models, as well as the introduction of domain knowledge may dramatically influence both the absolute and the relative performances of the different algorithms. To the best of our knowledge, the influence of these parameters on the quality of the predictions of recommender systems has rarely been reported in the literature.

Categories and Subject Descriptors

H.3.0 [Information Storage and Retrieval]: General;
H.2.8 [Database Applications]: Data Mining

General Terms

Experimentation, Algorithms, Performance

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'11, August 21–24, 2011, San Diego, California, USA.
Copyright 2011 ACM 978-1-4503-0813-7/11/08 ...\$10.00.

Keywords

Recommender Systems, Collaborative Filtering

1. INTRODUCTION

Systems for personalized recommendation aim at predicting a few items (movies, books, ...) of interest to a particular user. They have been extensively studied in the past decade [2, 17], with a particular attention given to the collaborative filtering (CF) approach [18, 24]. In contrast to content-based recommender systems, collaborative filters rely solely on the detection of interaction patterns between users and items, where an interaction can be the rating or the purchase of an item. For instance, a system can recommend to a target user some products purchased by other users whose purchase history is similar to the target user's, making recommendation possible even if relevant descriptions of users or items are unavailable.

The development of collaborative filters has followed several, mostly independent directions. The first direction, which accounts for the majority of works in CF, is the development of different types of algorithms, such as memory-based techniques, matrix factorization algorithms, probabilistic models and ensemble methods [18, 24, 6, 14]. Almost all of these works have focused on ratings prediction on publicly available benchmark datasets such as MovieLens¹ or Netflix². A second direction is the identification of domain dependent variables that describe interactions between users and items and which may influence the recommendation performance. For instance, purchase time can be used to take into account the recency of past customer purchases for the recommendation [7, 15]. Another example is contextual variables such as the intent of a purchase (e.g. whether a book is purchased for personal or professional purposes), which allow to perform situated recommendations of potentially greater accuracy [1, 3, 5, 19]. These works generally focus on the impact of introducing domain knowledge,

¹<http://www.grouplens.org/node/73>

²<http://www.netflixprize.com/>

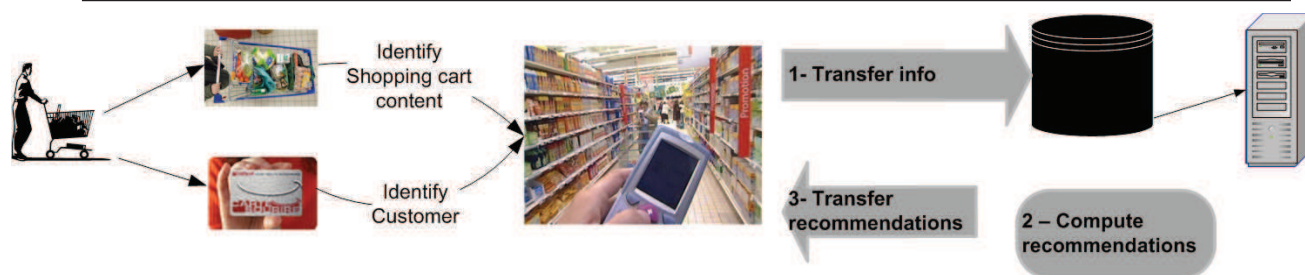


Figure 1: Vendors in the store use a PDA which is equipped to read product bar-codes and loyalty cards. An application has been developed to receive recommendations.

and have often been evaluated on medium-scale proprietary purchase datasets with very little emphasis on performance comparisons between algorithms.

The abundance of academic papers on CF should be contrasted with many critical aspects of the design of commercial recommender systems which remain mostly undocumented. A data miner who wants to build a recommender system has to choose the appropriate algorithm to use, and probably incorporate domain knowledge to increase the recommendation quality. However, for many retailers, rating data is unavailable, or may be unreliable because ratings are purely declarative. Personalized recommendations should then be based on the customers' purchase history or other kind of implicit feedback, which is a different type of data because it does not contain *negative feedback* [10]: high and low ratings inform us about what a user likes and dislikes, but past purchases do not reliably inform us about the products which will not interest the customer in the future. The choice of the best algorithm becomes an issue as most collaborative filtering algorithms have been compared and studied in depth only on rating data. Moreover, since the algorithmic dimension is often studied independently of the introduction of domain knowledge, the relative and joint impact of these two parameters on the recommendation quality is mostly unknown.

In this paper, we present a case-study on CF recommender systems using a dataset containing the purchase history of more than 50,000 loyal customers of a home improvement store over 3 years, containing about 4.6 million transactions. We systematically evaluate three families of CF algorithms, namely item-based recommendation [16], matrix factorization [14, 26, 20] and association rule mining algorithms [22, 11] on various settings in which we introduce increasing amounts of domain knowledge. In our case, domain knowledge is inferred from purchase data in which we identify *high purchase activity* periods of the customers. This leads us first to consider the recency of the purchases in a customer's history as a critical parameter, and secondly to a clustering of the high purchase activity periods, where the categories have a meaningful interpretation in terms of contextual variables. Our main results are:

- The blind application of the algorithms leads to very low performances, which are unacceptable for an industrial recommender system. The influence of incorporating domain knowledge in the recommendation process has already been reported in the literature, even for the task of ratings prediction [13] where taking into account the changes in user tastes over time improves performances. Domain knowledge is even more

important for short-term recommendation (as in our case) since the prediction task is more difficult.

- The difference in performance between the various algorithms is very small compared to what can be gained by introducing context in the recommendations. These results provide additional evidence for the need of context-aware recommender systems, in line with recent works [5, 19].
- Matrix factorizations algorithms and item-based techniques detect slightly different patterns between customers and items, as was already noticed in the context of ratings prediction [12]. On our purchase data, this leads factorization methods to mostly recommend best-sellers while item-based CF, like association rules, tends to recommend more products that are rarely bought.
- Surprisingly, the most accurate recommendations on our data are given by the simplest form of bigram association rules. This contrasts with the results on rating data where memory-based and matrix factorization are among the best performing methods [6, 14] and association rules are almost never used.

The paper is organized as follows. In Section 2, we present the dataset and the original recommendation task which lead to this study. We then describe our experimental protocol and the algorithms we evaluate in Sections 3 and 4 respectively. The results of an approach which uses no domain knowledge are reported in Section 5. We show the impact on the performance of a more business-oriented modeling in Section 6. Finally, Section 7 concludes the paper.

2. RECOMMENDATION TASK AND DATASET

This study is motivated by a need for an automatic recommender system, expressed by a French retailer, La Boîte à Outils³ (BAO). The company possesses 27 medium and large size stores in the South-East of France, and sells products for home improvement. BAO wants to equip salespeople in the stores with tools helping them provide their customers with better and more professional answers. Very often, customers come to the store and leave, without noticing they forgot some products they will need for the job they have in mind. After that, they start working and realize they have to come back, or try a closer competitor's store. BAO thus wants to help the salespeople make sure that the customer

³<http://www.laboiteaoutils.fr/>

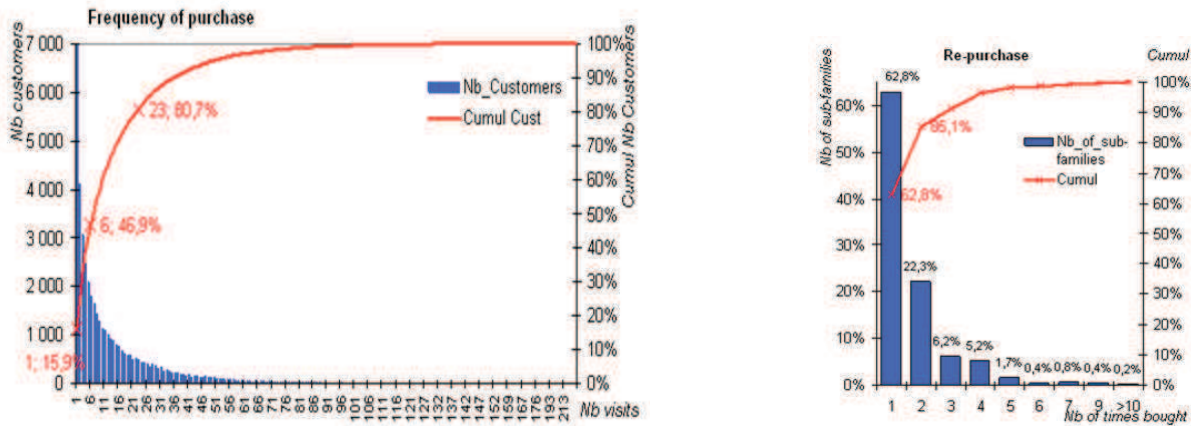


Figure 2: Frequency of purchase (left) and Re-purchase (right)

purchases the complete set of products he'll need: they can suggest missing products and show him the products in the relevant department. BAO thus launched a project to have recommendations produced in real-time for the salesperson to use in his discussion with the customer: salespeople are equipped with a PDA, which can read the loyalty card and the products bar-codes and to which the recommendations are to be sent. This scenario is represented in Figure 1. Less experienced vendors would certainly benefit more from the recommendation application, but even seasoned vendors have expressed their interest for it, saying that this could work for them as a check list of all the products needed. As can be seen, the setting for our recommender system is rather original, since most systems in the literature have been developed for on-line retailers.

The deployment scenario chosen by BAO was to first design and evaluate a recommender system on data from its major store; then to test it on the field in that store and finally deploy it throughout all the 27 stores. This paper reports on the first phase, where various techniques had to be evaluated for designing the best system, while phase 2 is presently ongoing.

The dataset we used contains all purchase history from loyal customers at BAO's major store from 2005 to 2008. As we shall see later, data from 2005 to 2007 were used for training and data for 2008 for testing. It corresponds to 43,779 customers and 3,425,048 transactions for training (from 2005 to 2007), and 30,784 customers (of which 6,923 are new customers) and 1,140,510 transactions for testing. The items we consider are not individual products but rather groups of products or *sub-families*, because BAO decided that recommendations should be made at this level of their product hierarchy. An example of sub-family is "pipe", while individual products are pipes of varying length, color, ... This will leave more space for the salesperson's expertise and should be easier to use in the intended context. We selected only the sub-families which were sold every year of the training data, and obtained 484 sub-families.

Figure 2 shows various characteristics of the 2005-2007 data: a RFM analysis showed that 15.9% customers visited only once (and 80.7% 23 times at most) and 15% sub-families were bought at least 3 times by the same customer. Since we are working at the sub-families level, this situation should be expected in our case, as well as in a DVD retail industry

for example, where customers typically buy various times at the genre – sub-family – level, yet each product once at most at the product level. This implies in particular that we need to potentially recommend an item (i.e. a sub-family) which has already been purchased by the same user.

As usual in retail contexts, data show a marked Long Tail effect [4]: 30% of the 484 sub-families (vintiles 1 to 6) represent 81.8% of total sales (these are the *Head* sub-families); while the last 11 vintiles (10 to 20) represent 55% of the sub-families and account for only 8.1% of sales (the *Tail*). Obviously, data are very sparse in the Tail and it will thus be harder to produce relevant recommendations in the Tail than in the Head. Being able to accurately recommend Middle or Tail items can however be an interesting property of a recommender system since it may correspond to more unexpected suggestions. We will see in our experiments that some algorithms essentially predict Head items, while some others are able to find some relevant Tail products.

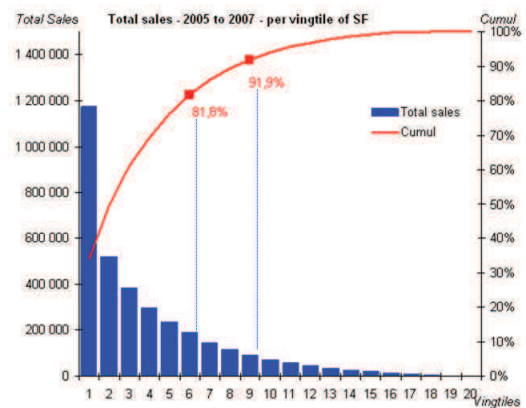


Figure 3: Total number of sales for each vintile (5%) of sub-families.

3. GENERAL EXPERIMENTAL PROTOCOL AND EVALUATION METRICS

We now present our experimental protocol. In our setup, a CF algorithm can be seen as a black box, which takes as input a *purchase matrix*, and outputs a predictor. The purchase matrix summarizes the training data (the transactions from 2005 to 2007 in our case) as described in Subsection

3.1. A predictor can be seen as a black box as well, which takes as input a customer's purchase history and outputs a sorted list of the 484 items. The quality of the sorted list is measured by comparing the first recommended items with items bought by the same customer in a near future. Our goal is to measure the ability of the system to perform relevant recommendations when the user is in the shop. In many cases, a customer needs a product for a particular short-term project, so our evaluation has to measure the quality of the predictions in the short-term. To that end, we decompose our test data (i.e. the 2008 purchase data) into several smaller sets, each of which is based on a specific day d of 2008, and we try to predict the items purchased by a customer within 2 weeks after that day, using the customer's purchase history known at d . The construction of these test sets is detailed in Subsection 3.2, and the evaluation metrics are defined in Subsection 3.3. How we train and test each algorithm is described later in the next Section.

3.1 Purchase Matrix

The training stage of the algorithms we evaluate takes as input a *purchase matrix*, where each column corresponds to an item, and each row represents a *purchase session*. A (*purchase*) *session* should be understood as “the set of purchases of a given customer over a period of time”. In standard collaborative filtering approaches (see e.g. [2]) and in Section 5 of this paper, a row of the purchase matrix represents the total purchase history of a specific customer. In that case, a purchase session is defined as “all known purchases of a customer” without limitation in time, but we will also consider purchase sessions defined on smaller periods of time in Section 6. In all cases, a purchase matrix, denoted $\mathbf{P}$, contains only Boolean values (0 or 1), where a 1 in the $(i, k)^{\text{th}}$ entry means that item k has been bought at least once in the i -th session (0 means the opposite). In the following, R (resp. C) will denote the number of rows (resp. columns) of $\mathbf{P}$.

3.2 Test data

As we intend to predict the future purchases of customers in the short term, we evaluate the various algorithms with a sliding window approach: for every day d of the testing period and every customer who made a purchase on that day, we predict his purchases for the next 14 days (including the *reference day* d , to predict the final purchases of the customer that day), based on his purchase history (i.e. the products bought strictly before the reference day). The window length of 14 days has been chosen by business constraints. It is motivated by the observation that most sales are *complete* within 14 days after the first purchase of this sale. In the end, there are 298 reference days and thus 298 test sets: each day of 2008 is a reference day except Sundays, holidays (when the shop is closed), and the last 13 days of 2008 (the purchases for the next 14 days are not available).

After creating the test sets, we filter some customers out of each set. For each test set, we discard all customers who did not make any purchase within 14 days before the corresponding reference day (excluded).⁴ This selection is once again motivated by the fact that sales are complete within 14 days, and will allow us to compare various experimental settings on the same set of customers.

⁴Note that customers not present on the reference day are not included in the test set either, since we can only make recommendations to customers present in the store.

3.3 Evaluation Metrics

The recommendation algorithms output a sorted list of items given the purchase history of a customer at a given date. Following [9, 8], we use precision, recall and F1-score to evaluate the quality of a recommendation list. These metrics are then averaged over customers and time to provide a final performance value. In our approach, we have a set of customers for each reference day, together with a target set of products for each customer (recall that the target set contains the customer's purchases for the next 14 days, and thus depends on the reference day). For a given customer u and a given reference day d , the precision, recall and F1-score of the N first recommended items are respectively defined as:

$$\begin{aligned} \bullet \text{ } prec_{u,d}@N &= \frac{\text{card}(R_{u,d}@N \cap T_{u,d})}{N}, \\ \bullet \text{ } rec_{u,d}@N &= \frac{\text{card}(R_{u,d}@N \cap T_{u,d})}{\text{card}(T_{u,d})} \end{aligned}$$

where given u and d , $T_{u,d}$ is the target set and $R_{u,d}@N$ the set of top- N recommended items.

$$\bullet \text{ } F1_{u,d}@N = \frac{2 \cdot rec_{u,d}@N \cdot prec_{u,d}@N}{rec_{u,d}@N + prec_{u,d}@N}.$$

The value of the cutoff N is chosen by business constraints. In our case, only 5 recommendations will be presented, so these measures are of greatest interest with $N = 5$.

To compute the final performance values, we first average over users for each reference day. The precision obtained for the reference day d is given by:

$$prec_d@N = \frac{\sum_{u: \text{test user for day } d} prec_{u,d}@N}{\text{nb of test users for day } d}.$$

Similar formulas are used to obtain $rec_d@N$ (resp. $F1_d@N$), the recall (resp. F1-score) for the reference day d . The final performances are then computed by averaging over all possible reference days:

$$perf@N = \frac{\sum_{d: \text{reference day}} perf_d@N}{\text{nb of reference days}} \quad \text{for } perf \in \{prec, rec, F1\}.$$

4. ALGORITHMS EVALUATED

We have selected a number of algorithms from the three main families of collaborative filtering approaches: memory-based, model-based and association rule learning. For the memory-based technique, we consider the item-based approach [16], in which the bulk of the computation is done offline. For model-based approaches, we selected two state-of-the-art matrix factorization methods, Singular Value Decomposition [23] and Non-Negative Matrix Factorization [27]. For association rule learning, we build the bigram matrix [22] (i.e. all rules of the form item $i \Rightarrow$ item j). The choices of the algorithms are arbitrary, we only intended to have representative algorithms for each of the main CF approaches.

4.1 Training the Algorithms

Item-Based Collaborative Filtering.

The item-based approach computes, on the training purchase data, a similarity matrix between products. Following Linden et al. [16], we compute the similarity between the

two products k and ℓ using the cosine between their corresponding columns of the purchase matrix:

$$\text{sim}(k, \ell) = \frac{\mathbf{P}_{\bullet k}^T \mathbf{P}_{\bullet \ell}}{\sqrt{\mathbf{P}_{\bullet k}^T \mathbf{P}_{\bullet k}} \sqrt{\mathbf{P}_{\bullet \ell}^T \mathbf{P}_{\bullet \ell}}}$$

where $\mathbf{P}_{\bullet k}$ is the k -th column of $\mathbf{P}$, and $\mathbf{X}^T$ is the transpose of matrix $\mathbf{X}$. The output of the training process is then the similarity matrix between products $\mathbf{S}$, where the $(k, \ell)^{\text{th}}$ entry of $\mathbf{S}$ is equal to $\text{sim}(k, \ell)$.

Matrix Factorization Methods.

Matrix factorization methods have shown great performance for ratings prediction [14, 26]. To apply these methods on purchase data, we use the AMAN (*all missing as negative*) reduction [20]. In this setup, matrix factorizations approximate the purchase matrix $\mathbf{P} \in \mathbb{R}^{R \times C}$ by a matrix of smaller rank r ($r \ll \text{rank}(\mathbf{P})$). The approximation can be written as the product $\mathbf{U}^* \mathbf{V}^*$ of two matrices $\mathbf{U}^* \in \mathbb{R}^{R \times r}$ and $\mathbf{V}^* \in \mathbb{R}^{r \times C}$. We first evaluate the factorization which gives the best rank- r approximation of $\mathbf{P}$ in the sense of the Frobenius norm which can be found using SVD [23]:

$$\mathbf{U}^*, \mathbf{V}^* = \arg \min_{\mathbf{U} \in \mathbb{R}^{R \times r}, \mathbf{V} \in \mathbb{R}^{r \times C}} \|\mathbf{P} - \mathbf{UV}\|_F^2 \quad (1)$$

where $\|\mathbf{X}\|_F = \sqrt{\sum_{i,j} x_{i,j}^2}$ is the Frobenius norm of $\mathbf{X}$ and $x_{i,j}$ the $(i, j)^{\text{th}}$ entry of $\mathbf{X}$. We also tried to weight missing entries as in [20], but it did not improve performances, so these results are not presented in our experiments.

We also evaluate non-negative matrix factorization (NNMF) which was shown to have good performances in CF [27]. NNMF finds an approximation of the purchase matrix with factors constrained to only have non-negative elements:

$$\mathbf{U}^*, \mathbf{V}^* = \arg \min_{\mathbf{U} \in \mathbb{R}_+^{R \times r}, \mathbf{V} \in \mathbb{R}_+^{r \times C}} \|\mathbf{P} - \mathbf{UV}\|_F^2. \quad (2)$$

For any matrix factorization method (SVD, NNMF, or any other), the rightmost matrix $\mathbf{V}^*$ can be seen as an *embedding matrix*, which projects every product k in $\mathbb{R}^r$ (or $\mathbb{R}_+^r$ for NNMF). Thus, the k -th column of $\mathbf{V}^*$, $\mathbf{V}_{\bullet k}^*$, can be interpreted as a representation of product k in the embedding space. On the other hand, $\mathbf{U}^*$ expresses the rows of the purchase matrix as a linear combination of lines of $\mathbf{V}^*$, and is thus representative of the training data only. Consequently, after obtaining a factorization of the training purchase matrix $\mathbf{P}$ of the form $\mathbf{U}^* \mathbf{V}^*$, $\mathbf{U}^*$ is not stored. The only parameter resulting from training is matrix $\mathbf{V}^*$.

Association Rules.

We also evaluate association rule learning for CF [22, 11]. We first consider *bigram* rules, where training simply consists in computing the following matrix on the purchase data:

$$\mathbf{B} = [b_{k,\ell}]_{\substack{k=1..C \\ \ell=1..C}} \quad \text{with} \quad b_{k,\ell} = \frac{\mathbf{P}_{\bullet k}^T \mathbf{P}_{\bullet \ell}}{\|\mathbf{P}_{\bullet k}\|_1}, \quad \|\mathbf{X}\|_1 = \sum_{k=1}^C |x_k| \quad (3)$$

$b_{k,\ell}$ is thus the confidence of rule $k \Rightarrow \ell$ (i.e. *item k is purchased implies item ℓ is purchased within the same session*).

Special care needs to be taken for diagonal elements $b_{k,k}$: they are all equal to 1 in (3), which does not reflect the real probability that item k was bought twice at different times within the same session. Although the true confidence of the rule $k \Rightarrow k$ cannot be computed from the purchase matrix,

we compute it from the initial transactional data as follows:

$$b_{k,k} = \frac{\text{nb sessions where } k \text{ was bought at least twice}}{\text{nb of sessions where } k \text{ was bought}}.$$

Since bigram rules consider at most two items without any consideration of their support, we tried a more refined version of associations rules (called *alpha-rules* in the paper) based on the concepts of alpha Galois lattice [25] and frequent closed itemsets [21]. alpha-rule learning aims at scaling up the rule mining using domain knowledge provided as a segmentation of purchase sessions: the algorithm only finds closed itemsets and rules satisfied by at least $\alpha\%$ sessions of a given segment (the parameter α controls the complexity of the method). This method is only applicable when having a typology of purchase sessions (which will be built in Section 6), so we only present its results after Section 6.

4.2 Applying the Algorithms

At apply time, an algorithm is given a customer's purchase history (which contains all past purchases of a customer or only the most recent ones), and is required to provide a list of N recommendations for this user ($N = 5$ in our case). A purchase history is represented by a Boolean column vector $\mathbf{H} \in \{0, 1\}^{1 \times C}$, where 1 at the k^{th} line means that item k has been bought by the customer at least once over the period considered for building the history. $\mathbf{H}$ contains the purchases of a single customer, so the prediction processes described below are applied to each customer of a test set. Note that our prediction processes can be applied to any customer, even if he was not seen during training.

Item-Based CF Prediction.

The recommendation list for item-based collaborative filtering is created as follows:

1. For each item k in $\mathbf{H}$ (i.e. k s.t. $h_k = 1$), select the S items which are the most similar to k using the similarity matrix $\mathbf{S}$ computed during training. We obtain $\|\mathbf{H}\|_1$ lists of items with similarity scores for each item in each list. In our experiments we use $S = 20$ as it tended to give the best results.
2. Merge the lists: concatenate the lists, and sort the items by decreasing associated score. If an item is in several lists, assign it its maximum score.

Prediction with Matrix Methods.

The inference procedure corresponds to the setting of *strong generalization* [18]. This setting allows us to make prediction for new customers or to take into account the updates (if any) of the customer's purchase history. Given the embedding matrix $\mathbf{V}^*$ learnt on the training data, we compute a parameter vector $\mathbf{W}^*$ such that $\mathbf{W}^{*T} \mathbf{V}^*$ approximates best the user's purchase history $\mathbf{H}$. Depending on the method, $\mathbf{W}^*$ is defined as follows:

- SVD: $\mathbf{W}^* = \arg \min_{\mathbf{W} \in \mathbb{R}^r} \|\mathbf{W}^T \mathbf{V}^* - \mathbf{H}\|_F^2$ with $\mathbf{V}^*$ given by (1),
- NNMF: $\mathbf{W}^* = \arg \min_{\mathbf{W} \in \mathbb{R}_+^r} \|\mathbf{W}^T \mathbf{V}^* - \mathbf{H}\|_F^2$ with $\mathbf{V}^*$ given by (2).

Both are convex quadratic problems which can be solved very efficiently at apply time. Once the parameter vector

Table 1: Test results for the *Complete History* (left) and the *2 Weeks History* (right) apply settings, when the algorithms are trained on the *user*×*item* purchase matrix. *Relative* differences of more than 5% on any metric and between any two algorithms are significant.

	<i>prec@5</i>	<i>rec@5</i>	<i>F1@5</i>
item-based	6.87%	6.10%	5.62%
SVD	13.22%	12.39%	10.92%
NNMF	11.28%	9.35%	8.85%
bigram rules	10.88%	9.33%	8.64%

	<i>prec@5</i>	<i>rec@5</i>	<i>F1@5</i>
item-based	11.76%	9.98%	9.39%
SVD	13.20%	12.37%	10.91%
NNMF	12.27%	11.25%	10.05%
bigram rules	14.54%	13.47%	11.98%

$\mathbf{W}^*$ is computed, all items are sorted by decreasing scores (the predicted score for item k is $\mathbf{W}^{*T} \mathbf{V}_{\bullet,k}$ in both cases).

Prediction with Association Rules.

For bigram rules, we follow [22]: items k are sorted by the maximum confidence of the rules $\ell \Rightarrow k$ supported by the user history (i.e. ℓ is such that $h_\ell = 1$). The prediction with alpha-rules is similar.

5. A STANDARD APPROACH

We first evaluate what we refer to as the *standard approach*. This corresponds to the case where each row of the training purchase matrix contains all known purchases of all training customers, called the *user*×*item* matrix. In that case, the training purchase matrix has a sparsity coefficient of 5.7%. After training each algorithm according to Section 4.1, we evaluate 2 different *apply settings*:

Complete History In the *Complete History* apply setting, the prediction algorithms (described in Section 4.2) receive as input a purchase vector which contains *all past purchases of this customer without limitation in time*. Thus, for a given reference day and a given customer, the prediction will be based on all of this customer’s purchases made strictly before the reference day.

2 Weeks History In the *2 Weeks History* apply setting, the prediction algorithms receive as input a purchase vector containing only the customer’s purchases made in the last 14 days before the reference day (excluded).

We remind to the reader that (1) the training procedure of the algorithms is the same for both apply settings, (2) thanks to the filter applied to the test sets (Subsection 3.2), the algorithms are evaluated on exactly the same customers and target sets, and finally (3) the reported performance for both apply settings are computed on the test data following the protocol described in Subsection 3.2.

The test performances in terms of *prec@5*, *rec@5* and *F1@5* for the various algorithms and the two apply settings are shown in Table 1. The rank of matrix factorization is chosen based on *F1@5* ($r = 20$ for both methods).

The results first show that bigram association rules in the 2 weeks history apply setting obtain the best results, with relative performance gains between 8% and 10% on all metrics. However, we have to notice that in the *Complete History* apply setting, factorization methods obtain the best results. Thus the relative performances of the methods change when taking recency into account. It implies that no general conclusion can be drawn from the relative performances of each algorithm in a specific application scenario, and that testing each method for each scenario is mandatory.

When comparing the two apply settings, we see that considering only the most recent purchases has a positive impact

on all performances, except for the SVD on which the difference is not significant. This suggests that only purchases made within a short period of time are strongly correlated, so that the recency of the purchase may be a critical factor for the recommendation. Similar observations have already been reported (see e.g. [14, 15]) on other datasets where users’ tastes evolve with time. Even though the reason might be different in our case (see Section 6), our results give additional evidence that the recency of user feedback is an important factor in CF. In particular, it appears to be crucial for item-based CF and association rules. This is due to the aggregation scheme we chose (the max of the items similarities or rules’ confidences), which is not robust to this noise. We carried out additional experiments (not reported here) where we used the *sum* of the rules’ confidence (as in [11]) instead of the max for association rules. The algorithm becomes then more robust, with less than 15% relative difference in *prec@5* between the two settings (instead of about 30% with the max as reported in Table 1). The same behavior was observed for item-based CF. The algorithms do not perform better though, because adding low confidences eventually leads to overcome the confidence of relevant rules.

In terms of absolute performance values, the best precision at 5 is lower than 15%. Acceptance of the system on the field will be hard: when the salesperson recommends 5 products to his customer, the latter will buy less than once on average. Sales staff will soon stop proposing recommendations to avoid losing the customers’ confidence. We need better performances, at least 20% of precision to ensure that at least one product is bought by the customer on average.

6. A BUSINESS-ORIENTED APPROACH

The business of BAO is to sell products to customers for home improvement. This customers’ activity has its specificities, one of which was highlighted in the previous section: customers have a purchasing behavior which varies a lot in time. At some periods, they only buy little, and only standard products (electric bulbs, nails, ...); while at other periods, they launch a *project* such as refurbishing the bathroom or putting a new roof. Projects requires more spending, more - very specific - products and this over a relatively short period. In this section, we try and exploit this behavior to focus on such specific activity, because it should allow us to narrow the potential choice of relevant products and thus recommend products at a given time depending on what the customer is doing at that time. From a business perspective, the recommender system should be most useful if it can help getting a complete sale for customers running a project, i.e. a sale where all the products necessary for that project are actually purchased.

We show how to detect projects periods in customers’ purchase histories and automatically categorize them in Subsec-

Table 2: Project categories found by k-means on the training data.

Category	Bathroom	Plumbing	Exterior	Exterior improvement	Carpentry	Tiling	Isolation	Paint	Electricity	Flooring/Paneling	Interior improvement
Nb of projects	1221	3207	14359	6136	3398	2588	1845	3221	2720	1802	1075
%	2.89	7.60	34.02	14.54	8.05	6.13	4.37	7.63	6.45	4.27	4.04
Avg nb of unique items in a project	31	15	3	8	10	11	12	10	14	14	13
Avg duration (days)	26	14	5	9	12	13	13	11	13	15	13
Nb of different items used	414	427	457	443	429	406	393	419	421	401	397

Table 3: Test results for the *Complete History* (left) and the *2 Weeks History* (right) apply settings when the algorithms are trained on the *project* $\times$ *item* matrix. A relative difference of more than 5% in any metric is always statistically significant.

	<i>prec@5</i>	<i>rec@5</i>	<i>F1@5</i>
item-based	5.41%	4.91%	4.43%
SVD	12.85%	11.80%	10.53%
NNMF	11.08%	9.43%	8.81%
bigram rules	9.52%	8.71%	7.71%
alpha-rules	14.19%	6.65%	8.53%

	<i>prec@5</i>	<i>rec@5</i>	<i>F1@5</i>
item-based	10.85%	9.57%	8.77%
SVD	12.90%	11.83%	10.56%
NNMF	12.48%	11.39%	10.20%
bigram rules	15.84%	16.42%	13.67%
alpha-rules	16.10%	7.57%	9.70%

tion 6.1. We then study two strategies to integrate projects and their categories in the recommendation process.

6.1 Project Definition and Categorization

The domain experts at BAO observed that past customers' projects can be extracted from their purchase history by identifying high purchase activity periods, defined as periods where a given customer spends at least 200 € over a duration of 14 days. BAO has specialized salespeople for customers running very large projects (i.e. building a house) and wants to exclude these from the perimeter of the recommender system. We thus extracted from the 2005-2007 dataset all projects by isolating high-purchase activity periods of the customers, discarding projects that last more than 50 days or with a total expense greater than 2,000 €. We then merged projects separated by less than a week and eliminated all no-purchase days at the end of the 42,202 identified projects to calculate the real project duration. Finally, projects account for about 1/3 of all the transactions, and 50% of the customers are involved in at least one project.

We built a typology of projects using k-means⁵ supervised by the number of items used in the project. We found a typology of 11 segments which were analyzed by business-owners who found a description as meaningful project categories. These categories are descriptive of the context of a purchase, such as plumbing or painting. The segments are summarized in Table 2. The two larger categories, account for 34% and 14.5% of all projects. All other categories (except for the smallest one) contain between 4% and 8% of the projects. Most categories contain projects which involve, on average, about 10 different unique items (with possibility of re-purchase), but more than 400 items are used in at least one project of each category (thus, more than 80% of the total number of items). Thus, even if the project categories contain more homogeneous sets of purchases, recommendation must cover most of the catalog.

⁵We used KXEN K2S segmentation module (KXEN AF v5.0.5, www.kxen.com.)

6.2 Training on Projects Only

We saw in Section 5 that considering only a short-term purchase history at prediction time increases the recommendation accuracy. Together with the analysis of projects, these results suggest that only short-term correlations between two purchases are relevant, long-term correlations being most probably accidental. We propose to integrate this additional knowledge into the training process, by learning on a modified purchase matrix which contains one row for each project detected in the period 2005-2007. We obtain a *project* $\times$ *item* purchase matrix of size 42,202 $\times$ 484. It has a higher sparsity than the *user* $\times$ *item* matrix (2.3% non-zero entries instead of 5.7%) but is also less noisy. Remember that the purchase matrix is only used to compute item/item similarities for item-based CF, the item projection matrix for factorization methods, and the confidence of the association rules. Thus, the models trained on this new purchase matrix are applied the same way as in Section 5. alpha-rules are trained on the *project* $\times$ *item* matrix like the other algorithms, but unlike them uses the categorization of projects for a faster exploration of the search space.

Recommendation performance.

The performances of the models in both the *Complete History* and the *2 Weeks History* apply settings are shown in Table 3. The optimal rank for SVD and NNMF is still $r = 20$. alpha-rules are with $\alpha = 10\%$ and a minimal support of 1%. The differences in performances between the two apply settings are similar to those of the previous section. In the *Complete History* apply setting, the models perform worse than in the previous section, which is natural since they are trained on purchase sessions that span over shorter periods of times. In the *2 Weeks History* apply setting, the results are similar to the previous results, except for the bigram rules which obtain significantly better performances, with a greater improvement of the recall (more than 20% relative increase). Thus, the benefit of training on the *project* $\times$ *item* matrix mostly concerns customers for which few items had to be predicted. alpha-rules obtain slightly better performances in terms of precision, but their

Table 4: Proportion of recommended and purchased items in the Head, the Middle and the Tail when training on the $user \times item$ matrix (left) and the $project \times item$ matrix (right) in the 2 Weeks History apply setting.

	Head	Middle	Tail
item-based	96.4%	2.5%	1.1%
SVD	98.7%	1.3%	0.0%
NNMF	99.4%	0.6%	0.0%
bigram rules	98.2%	1.1%	0.7%
alpha-rules	100%	0%	0%

	Head	Middle	Tail
item-based	93.0%	4.8%	2.2%
SVD	97%	2.7%	0.3%
NNMF	96.5%	3.2%	0.3%
bigram rules	90.8%	6.1%	3.1%
alpha-rules	95.4%	4.1%	0%

Table 5: Test results of non-contextual (left) and contextual (right) recommendations on the subset of users running a project on the reference day, in the 2 Weeks History apply setting. The non-contextual $project \times item$ model is trained and applied as in Section 6.2. The contextual model is described in Section 6.3.

	$prec@5$	$rec@5$	$F1@5$
item-based	15.08%	9.73%	10.53%
SVD	21.17%	11.21%	13.56%
NNMF	20.64%	10.97%	13.23%
bigram rules	24.15%	13.46%	15.88%
alpha-rules	25.70%	7.31%	9.40%

	$prec@5$	$rec@5$	$F1@5$
item-based	20.61%	12.83%	14.17%
SVD	24.30%	12.17%	15.22%
NNMF	26.26%	13.38%	16.57%
bigram rules	32.46%	17.69%	21.15%
alpha-rules	27.57%	12.81%	16.59%

recall and $F1$ is rather low. This is essentially due to the high setting of the minimal support, but decreasing it leads to very high complexity.

Recommended items.

The models trained on the $project \times item$ matrix produce slightly different recommendations than those trained on the $user \times item$ matrix. Table 4 compares these models in the 2 Weeks History apply setting. It shows, among the items that were recommended and actually purchased by the customer, the percentage of items belong to the Head (i.e. best-sellers), the Middle or the Tail. The Tail, which contains 55% of the items but account for only 8.1% of the sales (see Section 2) contains items that are very difficult to recommend at the right moment. Indeed, the models of the previous section are only able to make accurate recommendations in the Head (more than 95% of the correct recommendations for all models). The reason is that complete customers' purchase histories contain mostly Head items, so that Middle and Tail items are filtered out as noise by the algorithms. A larger part of the correct recommendations lies in the Middle or Tail when training on the $project \times item$ where short-term correlations between Middle or Tail products can be discovered. In particular, more than 9% of the bigram rules' correct recommendation are now Middle or Tail items.

6.3 Contextual Recommendation

The analysis of projects shows that a typology of more homogeneous purchase patterns can be found, and that these categories, such as "interior" or "exterior" improvement, have a clear interpretation in terms of contextual variables that describe the intent of the purchase. Following the multidimensional model of [2], we propose to evaluate the gain in predictive performance if one *knew* this context at prediction time. This is particularly relevant for our application, since we have very broad context descriptions: the context might be simply inferred by the salesperson during a normal dialogue with the customer; even asking for it has not be seen as too intrusive by professionals.

In the simplest form of the multidimensional model, given the context in which a recommendation is made, the prediction is based solely on past purchases made within the same context. In our case, this is implemented as follows:

- At training time, we train 11 models, one for each

category of projects (or, equivalently, we train one model for each possible context). Each of these models is trained on a specific $project \times item$ purchase matrix containing one row for each project of 2005-2007 of the corresponding category.

- At apply time, we assume that we know the category of the current customer's project (i.e. we know the context of the purchase). On the test data, this is simulated by filtering out all transactions of 2008 which do not belong to a project, and by assigning to each remaining transaction the project category it belongs to. Then, for each user and each reference day, we apply the model which corresponds to the project category.

As we do not evaluate on exactly the same set of transactions as in Sections 5 and 6.2, we also provide, on this set of transactions, the results obtained by the non-contextual model of Sections 6.2 as a baseline. Note that it is exactly the same model and predictions as in Section 6.2, but we consider now only the test users and transactions that belong to a high purchase activity at the reference day. Due to lack of space, we only report the results of the models in the 2 Weeks History apply setting⁶. These performances are shown in Table 5. Factorization methods use a rank $r = 5$.

The non-contextual models, on this subset of transactions, have higher precision and lower recall than in general (compare Table 3 (right) and Table 5 (left)), because people running projects buy more products, artificially inflating $prec@5$ and decreasing $rec@5$. The effect of contextualization, however, is critical: item-based CF, NNMF and bigram rules show a relative increase of at least 30% in all performance values. The effect is less dramatic for SVD (which tends to be the most constant model on our data) and for alpha-rules which already used the typology in the exploration of the rules. More interestingly, bigram rules have a precision of 32%. Thus, on average, between 1 and 2 recommended products are actually bought by the customer. We also note that despite their simplicity, bigram rules perform significantly better than other methods on all measures.

Our results, in line with recent works [5, 19], provide additional evidence of the importance of the context of a recom-

⁶The performances of the algorithms in the *Complete History* apply setting, and the results of the models trained as in Section 5, are consistently worse.

mentation. Such performances are acceptable in a dialogue with a customer. Even though this only affects a part of our test data (about 1/3 of transactions), these are the transactions that involve the greater number of items, and thus for which the recommender system should be of greater interest.

When looking at which items are recommended and purchased, we see that the better performances obtained by the contextual models are due to more accurate recommendations of best-sellers: for all algorithms, at least 95% of the correct recommendations involve Head items.

7. CONCLUSION

We presented a systematic comparison of various collaborative filtering systems on a real-life purchase dataset. We showed that recommender systems can be developed on *purchase* rather than *rating* data, and that, in such cases, algorithms comparison is different from the standard rating case. In particular, on our data, the simplest algorithm based on bigram association rules obtained the best performances. We believe that such case-studies are necessary to better understand the specificities of purchase datasets and the factors that impact recommender systems for retailers.

Our results show that factors such as purchase recency and context-awareness may be at least as important as the choice or the design of a well-performing algorithm. They also show that the relative performances of the algorithms vary depending on the setting to which they are applied, so that all algorithms have to be tested at all stages of the development of the recommender system.

Acknowledgements

The authors thank A. Bordes, D. Buffoni, S. Canu, A. Spengler and G. Wisniewski for fruitful discussions. This work was partially funded by the ANR (CADI project).

8. REFERENCES

- [1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach. *Trans. on Inf. Sys.*, 23(1):103–145, 2005.
- [2] G. Adomavicius and A. Tuzhilin. Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Trans. on Knowledge and Data Eng.*, 17(6):734–749, 2005.
- [3] G. Adomavicius and A. Tuzhilin. Context-aware Recommender Systems. In *Proc. of the 2008 ACM Conf. on Rec. Sys.*, pages 335–336, 2008.
- [4] C. Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006.
- [5] L. Baltrunas. Exploiting Contextual Information in Recommender Systems. In *Proc of the 2008 ACM Conf. on Rec. Sys.*, pages 295–298, 2008.
- [6] R. Bell, Y. Koren, and C. Volinsky. Chasing 1,000,000\$: How We Won the Netflix Prize. *ASA Stat. and Computing Graphics Newsletter*, 18(2), 2007.
- [7] Y. Ding, X. Li, and M. E. Orlowska. Recency-based Collaborative Filtering. In *Proc. of the 17th Australasian Database Conf.*, pages 99–107, 2006.
- [8] A. Gunawardana and G. Shani. A survey of accuracy evaluation metrics of recommendation tasks. *J. Mach. Learn. Res.*, 10:2935–2962, 2009.
- [9] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. on Inf. Sys.*, 22(1):5–53, 2004.
- [10] Y. Hu, Y. Koren, and C. Volinsky. Collaborative Filtering for Implicit Feedback Datasets. In *Proc. of ICDM*, pages 263–272, 2008.
- [11] C. Kim and J. Kim. A Recommendation Algorithm Using Multi-Level Association Rules. In *Proc. of the 2003 IEEE/WIC Intl. Conf. on Web Intel.*, 2003.
- [12] Y. Koren. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. In *Proc. of SIGKDD*, pages 426–434, 2008.
- [13] Y. Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, 2010.
- [14] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, 2009.
- [15] T. Q. Lee, Y. Park, and Y.-T. Park. An Empirical Study on Effectiveness of Temporal Information as Implicit Ratings. *Expert Syst. Appl.*, 36(2):1315–1321, 2009.
- [16] G. Linden, B. Smith, and J. York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7:76–80, 2003.
- [17] J. Liu, M. Z. Q. Chen, J. Chen, F. Deng, H. Zhang, Z. Zhang, and T. Zhou. Recent Advances in Personal Recommender Systems. *J. of Inf. and Sys. Science*, 5(2):230–247, 2009.
- [18] B. Marlin. Collaborative Filtering: A Machine Learning Perspective. Master’s thesis, University of Toronto, 2004.
- [19] C. Palmisano, A. Tuzhilin, and M. Gorgoglione. Using Context to Improve Predictive Modeling of Customers in Personalization Applications. *IEEE Trans. on Knowl. and Data Eng.*, 20(11):1535–1549, 2008.
- [20] R. Pan and M. Scholz. Mind the Gaps: Weighting the Unknown in Large-Scale One-Class Collaborative Filtering. In *Proc of SIGKDD*, pages 667–676, 2009.
- [21] N. Pasquier, R. Taouil, Y. Bastide, G. Stumme, and L. Lakhal. Generating a condensed representation for association rules. *J. Intell. Inf. Syst.*, 24(1):29–60, 2005.
- [22] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of Recommendation Algorithms for e-Commerce. In *Proc. of the 2nd ACM Conf. on Electronic Commerce*, pages 158–167, 2000.
- [23] N. Srebro and T. Jaakkola. Weighted Low-Rank Approximations. In *Proc. of the 20th Intl. Conf. on Mach. Learn.*, pages 720–727, 2003.
- [24] X. Su and T. M. Khoshgoftaar. A survey of Collaborative Filtering Techniques. *Adv. in Artif. Intell.*, 2009:2–2, 2009.
- [25] V. Ventos and H. Soldano. Alpha galois lattices: An overview. In *ICFCA*, pages 299–314, 2005.
- [26] M. Weimer, A. Karatzoglou, and A. J. Smola. Improving maximum margin matrix factorization. *Machine Learning*, 72(3):263–276, 2008.
- [27] S. Zhang, W. Wang, J. Ford, and F. Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *Proc. of SDM*, pages 549–553, 1996.

Annexes diverses

B.1 Calcul de la divergence

Nous détaillons ici le calcul de $\text{div}_Y(f(Y))$ la divergence de la fonction spectrale $f(Y)$ par rapport à la matrice Y . La fonction spectrale f est définie par :

$$f(Y) = \sum_{i=1}^p f_i(\sigma_i) U_i V_i^\top \triangleq U f(\Sigma) V^\top, \quad (\text{B.1})$$

où les fonctions f_i forment une famille de fonctions différentiables telles que $f(\Sigma) = \text{diag}(f_1(\sigma_1), \dots, f_p(\sigma_p))$, U_i désigne la i -ème colonne de la matrice U et V_j désigne la j -ième colonne de la matrice V . $Y = U \Sigma V^\top$ est la forme dite réduite de la décomposition en valeurs singulières de Y et $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_p)$ la matrice diagonale des valeurs singulières de Y . Notons que $U \in \mathcal{L}(p, n)$, et nous définissons $\tilde{U} \in \mathcal{L}(n, n)$, où $\tilde{U} = [U \quad Q]$ tel que $U^\top Q = 0$ et $Q^\top Q = I_{n-p}$.

Théorème B.1. *Si $Y \in \mathbb{R}^{n \times p}$ ($n > p$) est une matrice simple et de plein rang et f une fonction spectrale continue et différentiable en Y , alors :*

$$\text{div}_Y(f(Y)) = \sum_{i=1}^p \left(f'_i(\sigma_i) + (n-p) \frac{f_i(\sigma_i)}{\sigma_i} \right) + 2 \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p \frac{\sigma_i f_i(\sigma_i)}{\sigma_i^2 - \sigma_j^2} \quad (\text{B.2})$$

Ce qui suit constitue la preuve du théorème B.1 reprise d'après [Candès et al. 2012]. L'idée générale de la preuve est de commencer par exprimer la divergence de f en fonction de la différentielle de f . Ensuite il s'agit de décomposer cette différentielle en une somme de termes. Nous identifierons et donnerons une expression de chacun de ces termes.

Commençons par définir la différentielle utilisée :

Définition B.1. *Soit une matrice $Y \in \mathcal{L}(p, n)$ et une fonction $f : \mathcal{L}(p, n) \rightarrow \mathcal{L}(p, n)$. La différentielle au sens de Gateaux de f au point Y , si elle existe, est l'opérateur linéaire continu $d_Y f : \mathcal{L}(p, n) \rightarrow \mathcal{L}(p, n)$ tel que $\forall \Delta \in \mathcal{L}(p, n)$:*

$$\lim_{\epsilon \rightarrow 0} \left\| \frac{f(Y + \epsilon \Delta) - f(Y)}{\epsilon} - d_Y f(\Delta) \right\|_F = 0. \quad (\text{B.3})$$

Le lemme B.1 nous donne l'expression de la divergence en fonction de la différentielle.

Lemme B.1. *Si Y est une matrice simple et de plein rang et f une fonction spectrale continue et différentiable en Y , alors :*

$$\text{div}_Y(f(Y)) = \sum_{i=1}^n \sum_{j=1}^p M_{ij}^{(ij)}, \quad (\text{B.4})$$

où l'on définit la famille de matrices $M^{(ij)}$ par $M^{(ij)} = \tilde{U}^\top d_Y f(\tilde{U}_i V_j^\top) V$ pour $(i, j) \in (1 \dots n, 1 \dots p)$.

Preuve du Lemme B.1. Soit $Y \in \mathcal{L}(p, n)$ une matrice simple et de plein rang et $f : \mathcal{L}(p, n) \rightarrow \mathcal{L}(p, n)$ une fonction continue et différentiable sur Y . On a par définition :

$$\text{div}_Y f(Y) = \text{tr}(d_Y f).$$

Soit la famille $\{B^{(ij)}\}$ une base orthonormale de $\mathcal{L}(p, n)$. La trace d'un opérateur linéaire (en particulier $d_Y f$) étant indépendante du choix de la base on a :

$$\text{tr}(d_Y f) = \sum_{i=1}^n \sum_{j=1}^p \langle B^{(ij)}, d_Y f(B^{(ij)}) \rangle_F,$$

où $\langle \cdot, \cdot \rangle_F$ désigne le produit scalaire associé à la forme de Frobenius dans $\mathcal{L}(p, n)$.

En prenant $B^{(ij)} = \tilde{U}_i V_j^\top$, on obtient :

$$\begin{aligned} \operatorname{div}_Y f(Y) &= \sum_{i=1}^n \sum_{j=1}^p \langle B^{(ij)}, d_Y f(B^{(ij)}) \rangle \\ &= \sum_{i=1}^n \sum_{j=1}^p \langle \tilde{U}^\top B^{(ij)} V, \tilde{U}^\top d_Y f(B^{(ij)}) V \rangle, \end{aligned}$$

puisque les matrices $\tilde{U}$ et V sont orthonomales.

Or $\tilde{U}^\top B^{(ij)} V = \tilde{U}^\top U_i V_j^\top V = E^{(ij)}$, matrice de terme général $E_{kl}^{(ij)} = 1$ si $(k, l) = (i, j)$, 0 sinon. Notons que $\{E^{(ij)}\}$ forme la base canonique de $\mathcal{L}(p, n)$ et que, par conséquent, $\forall X \in \mathcal{L}(p, n)$, $\langle E^{(ij)}, X \rangle = X_{ij}$. Ainsi,

$$\operatorname{div}_Y f(Y) = \sum_{i=1}^n \sum_{j=1}^p \langle E^{(ij)}, \tilde{U}^\top d_Y f(B^{(ij)}) V \rangle = \sum_{i=1}^n \sum_{j=1}^p \left[\tilde{U}^\top d_Y f(B^{(ij)}) V \right]_{ij}.$$

□

Nous avons maintenant une écriture de la divergence en fonction de la différentielle. Développons le calcul de cette différentielle pour n'importe quelle matrice $X \in \mathcal{L}(p, n)$.

$$d_Y f(X) = d_Y \mathcal{U}(X) f(\Sigma) V^\top + U d_Y f_{\mathcal{S}}(X) V^\top + U f(\Sigma) d_Y \mathcal{V}(X)^\top, \quad (\text{B.5})$$

où $\mathcal{U}, \mathcal{V}, \mathcal{S}$ et $f_{\mathcal{S}}$ sont des applications définies de la façon suivante :

$$\begin{aligned} \mathcal{U} : \mathcal{L}(p, n) &\rightarrow \mathcal{L}(n, n) & \mathcal{S} : \mathcal{L}(p, n) &\rightarrow \mathcal{L}(p, n) \\ X &\rightarrow \mathcal{U}(X) = U_X, & X &\rightarrow \mathcal{S}(X) = \Sigma_X, \\ \mathcal{V} : \mathcal{L}(p, n) &\rightarrow \mathcal{L}(p, p) & f_{\mathcal{S}} : \mathcal{L}(p, n) &\rightarrow \mathcal{L}(p, n) \\ X &\rightarrow \mathcal{V}(X) = V_X, & X &\rightarrow f_{\mathcal{S}}(X) = f(\Sigma_X), \end{aligned}$$

avec $X = U_X \Sigma_X V_X^\top$ la décomposition en valeurs singulières de X . En multipliant à gauche par $\tilde{U}$ et à droite par V les termes de l'équation B.5, nous obtenons un terme analogue à celui du lemme B.1 :

$$\tilde{U}^\top d_Y f(X) V = \tilde{U}^\top d_Y \mathcal{U}(X) f(\Sigma) + I_{n \times p} d_Y f_{\mathcal{S}}(X) + I_{n \times p} f(\Sigma) d_Y \mathcal{V}(X)^\top V. \quad (\text{B.6})$$

Définissons deux autres applications $\Omega_U(X) = \tilde{U}^\top d_Y \mathcal{U}(X)$ et $\Omega_V(X) = d_Y \mathcal{V}(X)^\top V$ et prenons le cas particulier de $X = B^{(ij)}$. L'équation B.6 devient :

$$\begin{aligned} \tilde{U}^\top d_Y f(B^{(ij)}) V &= \Omega_U(B^{(ij)}) f(\Sigma) + I_{n \times p} d_Y f_{\mathcal{S}}(B^{(ij)}) + I_{n \times p} f(\Sigma) d_Y \Omega_V(B^{(ij)}) \\ &= S_{\mathcal{U}} + S_{\mathcal{S}} + S_{\mathcal{V}}. \end{aligned} \quad (\text{B.7})$$

On retrouve à gauche de l'équation le terme explicité dans le lemme B.1. Il faut désormais calculer les trois termes de droite de l'équation. Nous allons nous intéresser aux différentes parties de la matrice $\tilde{U}^\top d_Y f(B^{(ij)}) V$ et à partir de la décomposition suivante :

$$\begin{aligned} \operatorname{div}_Y f(Y) &= \sum_{i=1}^n \sum_{j=1}^p \left[\tilde{U}^\top d_Y f(B^{(ij)}) V \right]_{ij} \\ &= \sum_{i=1}^p \left[\tilde{U}^\top d_Y f(B^{(ii)}) V \right]_{ii} + \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p \left[\tilde{U}^\top d_Y f(B^{(ij)}) V \right]_{ij} + \sum_{i=p+1}^n \sum_{j=1}^p \left[\tilde{U}^\top d_Y f(B^{(ij)}) V \right]_{ij} \quad (\text{B.8}) \\ &= \sum_{i=1}^p D_{ii} + \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p T_{ij} + \sum_{i=p+1}^n \sum_{j=1}^p L_{ij}. \end{aligned}$$

La figure B.1 illustre ce que représentent les matrices D , T et L . L'objectif est désormais de calculer les parties qui nous intéressent dans les matrices $S_{\mathcal{U}}$, $S_{\mathcal{S}}$ et $S_{\mathcal{V}}$ en identifiant leur contribution dans les matrices D , T et L .

$$\tilde{U}^\top d_Y f(B^{(\bullet\bullet)})V = \begin{array}{|c|} \hline \begin{array}{c} T \\ \hline D \\ \hline T \end{array} \\ \hline L \\ \hline \end{array}$$

FIGURE B.1 – Représentation des différentes composantes D , T et L de $\tilde{U}^\top d_Y f(B^{(ij)})V$.

Le calcul de $S_{\mathcal{S}}$ est donné par le lemme suivant :

Lemme B.2. $\forall (i, j)$ tel que $B^{(ij)} = U_i V_j^\top$, on a :

$$[d_Y f_{\mathcal{S}}(B^{(ij)})]_{kl} = \begin{cases} f'_i(\sigma_i) & \text{si } i = j = k = l \\ 0 & \text{sinon.} \end{cases}$$

Preuve du lemme B.2. Par définition $\forall X \in \mathcal{L}(p, n)$:

$$\lim_{\epsilon \rightarrow 0} \left\| \frac{f_{\mathcal{S}}(Y + \epsilon X) - f_{\mathcal{S}}(Y)}{\epsilon} - d_Y f_{\mathcal{S}}(X) \right\| = 0.$$

Dans la limite, il est possible de remplacer $\mathcal{S}(Y + \epsilon X)$ par son développement limité $\mathcal{S}(Y) + \epsilon d_Y \mathcal{S}(X)$, ainsi nous obtenons :

$$\begin{aligned} \lim_{\epsilon \rightarrow 0} \left\| \frac{f(\mathcal{S}(Y) + \epsilon d_Y \mathcal{S}(X)) - f(\mathcal{S}(Y))}{\epsilon} - d_Y f_{\mathcal{S}}(X) \right\| &= 0, \\ \lim_{\epsilon \rightarrow 0} \left\| \frac{f(\Sigma + \epsilon d_Y \mathcal{S}(X)) - f(\Sigma)}{\epsilon} - d_Y f_{\mathcal{S}}(X) \right\| &= 0. \end{aligned}$$

On a donc $d_Y f_{\mathcal{S}}(X) = d_\Sigma f(d_Y \mathcal{S}(X))$, qui est la règle de dérivation des fonctions composées. Il faut maintenant calculer $d_\Sigma f(X)$ et $d_Y \mathcal{S}(X)$ pour toute matrice X .

Commençons par calculer $d_Y \mathcal{S}(X)$. Pour cela définissons deux nouvelles applications : $\tilde{\mathcal{V}}(X) = V_X^\top V_X$ et $\tilde{\mathcal{U}}(X) = U_X^\top U_X$. Par la règle de dérivation de fonctions composées, nous avons $\forall X$:

$$\begin{aligned} d_Y \tilde{\mathcal{V}}(X) &= d_Y \mathcal{V}(X)^\top V + V^\top d_Y \mathcal{V}(X), & d_Y \tilde{\mathcal{U}}(X) &= d_Y \mathcal{U}(X)^\top U + U^\top d_Y \mathcal{U}(X), \\ 0 &= d_Y \mathcal{V}(X)^\top V + V^\top d_Y \mathcal{V}(X), & 0 &= d_Y \mathcal{U}(X)^\top U + U^\top d_Y \mathcal{U}(X), \\ \Omega_V(X) &= -\Omega_V(X)^\top, & \Omega_U(X) &= -\Omega_U(X)^\top. \end{aligned} \tag{B.9}$$

Donc Ω_U et Ω_V sont antisymétriques. De plus, dans le cas où $f(X) = Id(X)$ est la fonction identité, nous pouvons réécrire l'équation B.6 de la façon suivante :

$$\tilde{U}^\top X V = \Omega_U(X) \Sigma + I_{n \times p} d_Y \mathcal{S}(X) + I_{n \times p} \Sigma d_Y \Omega_V(X). \tag{B.10}$$

Ainsi, comme $\Omega_U(X)$ et $\Omega_V(X)$ sont antisymétriques, alors le terme central $d_Y \mathcal{S}(X)$ est l'unique composante diagonale de $\tilde{U}^\top X V$, c'est-à-dire que $[\tilde{U}^\top X V]_{kk} = [d_Y \mathcal{S}(X)]_{kk}$. Notons de plus que $\mathcal{S}(X)$

est diagonale, alors $d_Y \mathcal{S}(X)$ est diagonale pour tout X . Pour $X = B^{(ij)}$ et pour tout $k = 1, \dots, p$, nous avons enfin :

$$\begin{aligned} [d_Y \mathcal{S}(B^{(ij)})]_{kk} &= [\tilde{U}^\top B^{(ij)} V]_{kk} \\ &= E_{kk}^{(ij)} \\ &= \begin{cases} 1 & \text{si } i = j = k \\ 0 & \text{sinon.} \end{cases} \end{aligned} \quad (\text{B.11})$$

Il faut maintenant calculer $d_\Sigma f(X)$. Par définition :

$$\lim_{\epsilon \rightarrow 0} \left\| \frac{f(\Sigma + \epsilon X) - f(\Sigma)}{\epsilon} - d_\Sigma f(X) \right\| = 0.$$

Comme la somme de termes positifs tend vers 0 ssi chacun de ces termes tendent vers 0, On regardons le comportement de chaque terme de la matrice.

$$\forall (i, j), \lim_{\epsilon \rightarrow 0} \left| \frac{[f(\Sigma + \epsilon X)]_{ij} - [f(\Sigma)]_{ij}}{\epsilon} - [d_\Sigma f(X)]_{ij} \right| = 0. \quad (\text{B.12})$$

Dans le cas où X est diagonale, on a : si $i = j$, alors

$$\begin{aligned} \lim_{\epsilon \rightarrow 0} \left| \frac{f_i(\sigma_i + \epsilon X_{ii}) - f_i(\sigma_i)}{\epsilon} - [d_\Sigma f(X)]_{ii} \right| &= 0, \\ \lim_{\epsilon \rightarrow 0} |f'_i(\sigma_i) X_{ii} - [d_\Sigma f(X)]_{ii}| &= 0, \\ [d_\Sigma f(X)]_{ii} &= f'_i(\sigma_i) X_{ii}. \end{aligned}$$

Si $i \neq j$, alors

$$\begin{aligned} \lim_{\epsilon \rightarrow 0} \left| \frac{[f(\Sigma + \epsilon X)]_{ij} - f(\Sigma)_{ij}}{\epsilon} - [d_\Sigma f(X)]_{ij} \right| &= 0, \\ \lim_{\epsilon \rightarrow 0} |0 - [d_\Sigma f(X)]_{ij}| &= 0, \\ [d_\Sigma f(X)]_{ij} &= 0. \end{aligned}$$

Finalement, on obtient effectivement :

$$\begin{aligned} [d_Y f_{\mathcal{S}}(B^{(ij)})]_{kl} &= [d_\Sigma f(d_Y \mathcal{S}(B^{(ij)}))]_{kl} \\ &= \begin{cases} f'_k(\sigma_k) d_Y \mathcal{S}(B^{(ij)}) & \text{si } k = l \\ 0 & \text{sinon} \end{cases} \\ &= \begin{cases} f'_k(\sigma_k) & \text{si } i = j = k = l \\ 0 & \text{sinon.} \end{cases} \end{aligned}$$

□

Nous avons vu au cours de la preuve du lemme B.2 que $S_{\mathcal{S}}$ est l'unique contribution à D . Nous allons maintenant nous intéresser aux termes qui contribuent à T .

Nous l'avons vu également dans la preuve du lemme B.2, les matrices $\Omega_U(X)$ et $\Omega_V(X)$ sont antisymétriques tandis que $d_Y f_{\mathcal{S}}(X)$ est diagonale. Ainsi $S_{\mathcal{U}}$ et $S_{\mathcal{V}}$ contribuent seules au bloc T . On a donc, $\forall (i, j) \in (1 \dots p, 1 \dots p)$ tq $i \neq j$:

$$\begin{aligned}
 & \begin{cases} [\tilde{U}^\top X V]_{ij} = [\Omega_U(X) \Sigma]_{ij} + [\Sigma \Omega_V(X)]_{ij}, \\ [\tilde{U}^\top X V]_{ji} = [\Omega_U(X) \Sigma]_{ji} + [\Sigma \Omega_V(X)]_{ji}, \end{cases} \\
 \Rightarrow & \begin{cases} [\tilde{U}^\top X V]_{ij} = [\Omega_U(X)]_{ij} \sigma_j + \sigma_i [\Omega_V(X)]_{ij}, \\ [\tilde{U}^\top X V]_{ji} = [\Omega_U(X)]_{ji} \sigma_i + \sigma_j [\Omega_V(X)]_{ji}, \end{cases} \quad (\text{par définition de } \Sigma), \\
 \Rightarrow & \begin{cases} [\tilde{U}^\top X V]_{ij} = [\Omega_U(X)]_{ij} \sigma_j + \sigma_i [\Omega_V(X)]_{ij}, \\ -[\tilde{U}^\top X V]_{ji} = [\Omega_U(X)]_{ij} \sigma_i + \sigma_j [\Omega_V(X)]_{ij}, \end{cases} \quad (\text{par antisymétrie}), \\
 \Rightarrow & \begin{bmatrix} [\tilde{U}^\top X V]_{ij} \\ -[\tilde{U}^\top X V]_{ji} \end{bmatrix} = \begin{bmatrix} \sigma_j & \sigma_i \\ \sigma_i & \sigma_j \end{bmatrix} \begin{bmatrix} [\Omega_U(X)]_{ij} \\ [\Omega_V(X)]_{ij} \end{bmatrix}, \\
 \Rightarrow & \begin{bmatrix} [\Omega_U(X)]_{ij} \\ [\Omega_V(X)]_{ij} \end{bmatrix} = \frac{1}{\sigma_j^2 - \sigma_i^2} \begin{bmatrix} \sigma_j & -\sigma_i \\ -\sigma_i & \sigma_j \end{bmatrix} \begin{bmatrix} [\tilde{U}^\top X V]_{ij} \\ -[\tilde{U}^\top X V]_{ji} \end{bmatrix}, \\
 \Rightarrow & \begin{bmatrix} [\Omega_U(X)]_{ij} \\ [\Omega_V(X)]_{ij} \end{bmatrix} = \frac{1}{\sigma_j^2 - \sigma_i^2} \begin{bmatrix} \sigma_j & \sigma_i \\ -\sigma_i & -\sigma_j \end{bmatrix} \begin{bmatrix} [\tilde{U}^\top X V]_{ij} \\ [\tilde{U}^\top X V]_{ji} \end{bmatrix}.
 \end{aligned}$$

En particulier pour $X = B^{(kl)}$, on obtient :

$$\begin{aligned}
 & \Rightarrow \begin{bmatrix} [\Omega_U(B^{(kl)})]_{ij} \\ [\Omega_V(B^{(kl)})]_{ij} \end{bmatrix} = \frac{1}{\sigma_j^2 - \sigma_i^2} \begin{bmatrix} \sigma_j & \sigma_i \\ -\sigma_i & -\sigma_j \end{bmatrix} \begin{bmatrix} [\tilde{U}^\top B^{(kl)} V]_{ij} \\ [\tilde{U}^\top B^{(kl)} V]_{ji} \end{bmatrix} \\
 & \Rightarrow \begin{bmatrix} [\Omega_U(B^{(kl)})]_{ij} \\ [\Omega_V(B^{(kl)})]_{ij} \end{bmatrix} = \frac{1}{\sigma_j^2 - \sigma_i^2} \begin{bmatrix} \sigma_j & \sigma_i \\ -\sigma_i & -\sigma_j \end{bmatrix} \begin{bmatrix} \delta_{ik} \delta_{jl} \\ \delta_{il} \delta_{jk} \end{bmatrix}. \tag{B.13}
 \end{aligned}$$

où δ_{ij} est le symbole de Kronecker. Nous connaissons maintenant les valeurs de $\Omega_U(B^{(ij)})$ et de $\Omega_V(B^{(ij)})$. Nous pouvons calculer $S_{\mathcal{V}}$, mais pas encore $S_{\mathcal{U}}$ qui contribue également au bloc L (à lui seul) en plus du bloc T .

Pour calculer le bloc L il nous faut identifier certains éléments auparavant, à savoir $d_Y \mathcal{U}(X)$. En repartant de l'équation B.10, nous avons $\forall (i, j) = (p+1 \dots n, 1 \dots p)$:

$$\begin{aligned}
 & [\tilde{U}^\top X V]_{ij} = [\Omega_U(X) \Sigma]_{ij} \\
 \Rightarrow & [\tilde{U}^\top X V]_{ij} = [\tilde{U}^\top d_Y \mathcal{U}(X) \Sigma]_{ij} \\
 \Rightarrow & [X V]_{ij} = [d_Y \mathcal{U}(X) \Sigma]_{ij} \\
 \Rightarrow & [X V \Sigma^{-1}]_{ij} = [d_Y \mathcal{U}(X)]_{ij}. \tag{B.14}
 \end{aligned}$$

Nous avons identifié tous les éléments dont nous avons besoin. La figure B.2 permet de mieux visualiser le rôle de chaque élément de l'équation B.7.

Par identification en utilisant le lemme B.1, nous obtenons :

$$\sum_{i=1}^p D_{ii} = \sum_{i=1}^p S_{\mathcal{S}ii} = \sum_{i=1}^p \left[d_Y f_{\mathcal{S}}(B^{(ii)}) \right]_{ii} = \sum_{i=1}^p f'_i(\sigma_i) \left[d_Y \mathcal{S}(B^{(ii)}) \right]_{ii} = \sum_{i=1}^p f'_i(\sigma_i). \tag{B.15}$$

En utilisant l'équation B.13, nous obtenons :

$$\tilde{U}^\top d_Y f(B^{(ij)}) V = S_{\mathcal{U}} + S_{\mathcal{D}} + S_{\mathcal{V}}$$

FIGURE B.2 – Illustration de la contribution des matrices $S_{\mathcal{U}}$, $S_{\mathcal{D}}$ et $S_{\mathcal{V}}$ de l'équation B.7.

$$\begin{aligned}
 \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p T_{ij} &= \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p S_{\mathcal{U}ij} + S_{\mathcal{V}ij} \\
 &= \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p \left[\Omega_U(B^{(ij)}) f(\Sigma) \right]_{ij} + \left[f(\Sigma) \Omega_V(B^{(ij)}) \right]_{ij} \\
 &= \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p \frac{(\sigma_j \delta_{ii} \delta_{jj} + \sigma_i \delta_{ij} \delta_{ji}) f_j(\sigma_j)}{\sigma_j^2 - \sigma_i^2} + \frac{f_i(\sigma_i) (-\sigma_i \delta_{ii} \delta_{jj} - \sigma_j \delta_{ij} \delta_{ji})}{\sigma_j^2 - \sigma_i^2} \\
 &= \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p \frac{(\sigma_j \delta_{ii} \delta_{jj} + \sigma_i \delta_{ij} \delta_{ji}) f_j(\sigma_j)}{\sigma_j^2 - \sigma_i^2} + \frac{f_i(\sigma_i) (-\sigma_i \delta_{ii} \delta_{jj} - \sigma_j \delta_{ij} \delta_{ji})}{\sigma_j^2 - \sigma_i^2} \\
 &= \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p \frac{\sigma_j f_j(\sigma_j)}{\sigma_j^2 - \sigma_i^2} + \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p \frac{-f_i(\sigma_i) \sigma_i}{\sigma_j^2 - \sigma_i^2} = 2 \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p \frac{\sigma_i f_i(\sigma_i)}{\sigma_i^2 - \sigma_j^2}.
 \end{aligned} \tag{B.16}$$

En utilisant l'équation B.14, nous obtenons :

$$\begin{aligned}
 \sum_{i=p+1}^n \sum_{j=1}^p L_{ij} &= \sum_{i=p+1}^n \sum_{j=1}^p S_{\mathcal{U}ij} \\
 &= \sum_{i=p+1}^n \sum_{j=1}^p \left[\Omega_U(B^{(ij)}) f(\Sigma) \right]_{ij} \\
 &= \sum_{i=p+1}^n \sum_{j=1}^p \left[\tilde{U} B^{(ij)} V \Sigma^{-1} f(\Sigma) \right]_{ij} \\
 &= \sum_{i=p+1}^n \sum_{j=1}^p \left[\tilde{U} B^{(ij)} V \right]_{ij} \sigma_j^{-1} f_j(\sigma_j) \\
 &= \sum_{i=p+1}^n \sum_{j=1}^p \frac{f_j(\sigma_j)}{\sigma_j} = \sum_{j=1}^p (n-p) \frac{f_j(\sigma_j)}{\sigma_j}.
 \end{aligned} \tag{B.17}$$

Enfin, à partir des équations B.8, B.15, B.16 et B.17 on en déduit :

$$\text{div}_Y(f(Y)) = \sum_{i=1}^p \left(f'_i(\sigma_i) + (n-p) \frac{f_i(\sigma_i)}{\sigma_i} \right) + 2 \sum_{i=1}^p \sum_{\substack{j=1 \\ j \neq i}}^p \frac{\sigma_i f_i(\sigma_i)}{\sigma_i^2 - \sigma_j^2}.$$

B.2 Introduction à la notion de différentielle pour les matrices

La notion de différentielle pour les matrices est peu commune. Nous allons, à travers un exemple simple, présenter des détails permettant d'exprimer la divergence d'une fonction f à l'aide de sa différentielle. Soit $Y \in \mathcal{L}(n, n)$. L'exemple choisi est celui de la fonction $f(Y) = Y Y = Y^2$. Commençons par définir la différentielle utilisée :

Définition B.2. Soit une matrice $Y \in \mathcal{L}(p, n)$ et une fonction $f : \mathcal{L}(p, n) \rightarrow \mathcal{L}(p, n)$. La différentielle au sens de Gateaux de f au point Y , si elle existe, est l'opérateur linéaire continu $d_Y f : \mathcal{L}(p, n) \rightarrow \mathcal{L}(p, n)$ tel que $\forall \Delta \in \mathcal{L}(p, n)$:

$$\lim_{\epsilon \rightarrow 0} \left\| \frac{f(Y + \epsilon \Delta) - f(Y)}{\epsilon} - d_Y f(\Delta) \right\|_F = 0. \quad (\text{B.18})$$

Pour notre exemple, $p = n$ et nous avons :

$$\begin{aligned} \frac{f(Y + \epsilon \Delta) - f(Y)}{\epsilon} &= \frac{(Y + \epsilon \Delta)^2 - Y^2}{\epsilon} \\ &= \frac{\epsilon Y \Delta + \epsilon \Delta Y + \epsilon^2 \Delta^2}{\epsilon} = Y \Delta + \Delta Y + \epsilon \Delta^2 \end{aligned}$$

et donc

$$d_Y f(\Delta) = Y \Delta + \Delta Y. \quad (\text{B.19})$$

La différentielle de f au point Y au sens de Gateaux est donc l'opérateur linéaire :

$$\begin{aligned} d_Y f : \mathbb{R}^{n^2} &\longrightarrow \mathbb{R}^{n^2}, \\ \text{vec}(\Delta) &\longmapsto \text{vec}(d_Y f(\Delta)) = D_Y \text{vec}(\Delta), \end{aligned}$$

où $D_Y = I_n \otimes Y + Y \otimes I_n$ est une matrice de taille $n^2 \times n^2$ et $\otimes$ désigne ici le produit de Kronecker.

Nous allons maintenant calculer la divergence de f de trois manières différentes.

1. En utilisant la définition de la divergence :

$$\begin{aligned} \text{div}_Y(Y^2) &= \sum_{i=1}^n \sum_{j=1}^n \frac{\partial (Y^2)_{ij}}{\partial Y_{ij}} \\ &= \sum_{i=1}^n \sum_{j=1}^n \frac{\partial (\sum_k Y_{ik} Y_{kj})}{\partial Y_{ij}} \\ &= \sum_{i=1}^n \sum_{j=1}^n Y_{ii} + Y_{jj} = 2n \sum_{i=1}^n Y_{ii} \end{aligned}$$

2. En utilisant le fait que la divergence est égale à la trace de la jacobienne :

$$\text{div}_Y(Y^2) = \text{tr}(D_Y) = \text{tr}(I_n \otimes Y + Y \otimes I_n) = 2n \text{tr}(Y) = 2n \sum_{i=1}^n Y_{ii}$$

3. En utilisant l'analogie d'un changement de base avec E la matrice identité de $\mathbb{R}^{n^2 \times n^2}$ et $E^{(ij)}$ la matrice $n \times n$ telle que $\text{vec}(E^{(ij)})$ soit la $(i-1)n + j$ ème colonne de la matrice E . On a alors :

$$\begin{aligned} \text{div}_Y(Y^2) &= \text{tr}(E^t D_Y E) \\ &= \sum_{i=1}^n \sum_{j=1}^n \text{vec}(E^{(ij)})^t \text{vec}(Y E^{(ij)} + E^{(ij)} Y) \\ &= \sum_{i=1}^n \sum_{j=1}^n \langle E^{(ij)}, (Y E^{(ij)} + E^{(ij)} Y) \rangle_F = 2n \sum_{i=1}^n Y_{ii} \end{aligned}$$

B.3 Calcul de la fonction de répartition de la loi de Marchenko-Pastur

La loi de Marchenko-Pastur de paramètre $c > 0$ est définie par sa fonction de densité de probabilité qui vaut :

$$P_c(x) = \begin{cases} \frac{1}{2\pi xc} \sqrt{(x-a)(b-x)} & \text{si } a \leq x \leq b, \\ 0 & \text{sinon,} \end{cases}$$

où $a = (1 - \sqrt{c})^2$ et $b = (1 + \sqrt{c})^2$. La fonction de répartition F_c de cette loi est :

$$F_c(x) = \frac{1}{2\pi c} \left[\sqrt{-x^2 + (a+b)x - ab} - \frac{a+b}{2} \arcsin\left(\frac{-2x+a+b}{b-a}\right) - \sqrt{ab} \arcsin\left(\frac{(a+b)x-2ab}{x(b-a)}\right) \right] + \frac{1}{2}. \quad (\text{B.20})$$

Nous aurons besoin de l'expression de certaines primitives connues :

$$\int \frac{\sqrt{ax^2+bx+c}}{x} dx = \sqrt{ax^2+bx+c} + \frac{b}{2} \int \frac{dx}{\sqrt{ax^2+bx+c}} + c \int \frac{dx}{x\sqrt{ax^2+bx+c}}, \quad (\text{B.21})$$

$$\int \frac{dx}{\sqrt{ax^2+bx+c}} = -\frac{1}{\sqrt{-a}} \arcsin\left(\frac{2ax+b}{\sqrt{b^2-4ac}}\right) \quad (\text{si } a < 0) \quad (\text{B.22})$$

$$\int \frac{dx}{x\sqrt{ax^2+bx+c}} = \frac{1}{\sqrt{-c}} \arcsin\left(\frac{bx+2c}{|x|\sqrt{b^2-4ac}}\right) \quad (\text{si } c < 0) \quad (\text{B.23})$$

Nous allons détailler le calcul de la fonction de répartition de la loi de Marchenko-Pastur

$$\begin{aligned} F_c(x) &= \int_{-\infty}^x \frac{1}{2\pi cz} \sqrt{(z-a)(b-z)} dz, \\ &= \frac{1}{2\pi c} \int_a^x \frac{\sqrt{(z-a)(b-z)}}{z} dz, \\ &= \frac{1}{2\pi c} \int_a^x \frac{\sqrt{-z^2 + (a+b)z - ab}}{z} dz, \end{aligned}$$

en utilisant B.21 :

$$F_c(x) = \frac{1}{2\pi c} \left(\left[\sqrt{-z^2 + (a+b)z - ab} \right]_a^x + \frac{a+b}{2} \int_a^x \frac{dz}{\sqrt{-z^2 + (a+b)z - ab}} - ab \int_a^x \frac{dz}{z\sqrt{-z^2 + (a+b)z - ab}} \right),$$

en utilisant B.22 :

$$F_c(x) = \frac{1}{2\pi c} \left(\sqrt{-x^2 + (a+b)x - ab} + \frac{a+b}{2} \left[-\arcsin\left(\frac{-2z+a+b}{\sqrt{(a+b)^2 - 4ab}}\right) \right]_a^x - ab \int_a^x \frac{dz}{z\sqrt{-z^2 + (a+b)z - ab}} \right),$$

en utilisant B.23 :

$$\begin{aligned}
F_c(x) &= \frac{1}{2\pi c} \left(\sqrt{-x^2 + (a+b)x - ab} - \frac{a+b}{2} \left(\arcsin\left(\frac{-2x+a+b}{b-a}\right) - \frac{\pi}{2} \right) \right. \\
&\quad \left. - ab \left[\frac{1}{\sqrt{ab}} \arcsin\left(\frac{(a+b)z-2ab}{|z|\sqrt{(a+b)^2-4ab}}\right) \right]_a^x \right), \\
&= \frac{1}{2\pi c} \left(\sqrt{-x^2 + (a+b)x - ab} - \frac{a+b}{2} \left(\arcsin\left(\frac{-2x+a+b}{b-a}\right) - \frac{\pi}{2} \right) \right. \\
&\quad \left. - \sqrt{ab} \left(\arcsin\left(\frac{(a+b)x-2ab}{x(b-a)}\right) + \frac{\pi}{2} \right) \right), \\
&= \frac{1}{2\pi c} \left(\sqrt{-x^2 + (a+b)x - ab} - \frac{a+b}{2} \left(\arcsin\left(\frac{-2x+a+b}{b-a}\right) \right) \right. \\
&\quad \left. - \sqrt{ab} \left(\arcsin\left(\frac{(a+b)x-2ab}{x(b-a)}\right) \right) \right) + \frac{1}{2\pi c} \left(-\frac{(a+b)(-\pi)}{2} - \frac{\pi\sqrt{ab}}{2} \right), \\
&= \frac{1}{2\pi c} \left(\sqrt{-x^2 + (a+b)x - ab} - \frac{a+b}{2} \left(\arcsin\left(\frac{-2x+a+b}{b-a}\right) \right) \right. \\
&\quad \left. - \sqrt{ab} \left(\arcsin\left(\frac{(a+b)x-2ab}{x(b-a)}\right) \right) \right) + \frac{1}{4c} \left(\frac{a+b}{2} - \sqrt{ab} \right), \\
&= \frac{1}{2\pi c} \left(\sqrt{-x^2 + (a+b)x - ab} - \frac{a+b}{2} \left(\arcsin\left(\frac{-2x+a+b}{b-a}\right) \right) \right. \\
&\quad \left. - \sqrt{ab} \left(\arcsin\left(\frac{(a+b)x-2ab}{x(b-a)}\right) \right) \right) + \frac{1}{4c} \left(\frac{1}{2}(\sqrt{a} - \sqrt{b})^2 \right), \\
&= \frac{1}{2\pi c} \left(\sqrt{-x^2 + (a+b)x - ab} - \frac{a+b}{2} \left(\arcsin\left(\frac{-2x+a+b}{b-a}\right) \right) \right. \\
&\quad \left. - \sqrt{ab} \left(\arcsin\left(\frac{(a+b)x-2ab}{x(b-a)}\right) \right) \right) + \frac{1}{8c} (1 - \sqrt{c} - 1 - \sqrt{c})^2, \\
&= \frac{1}{2\pi c} \left(\sqrt{-x^2 + (a+b)x - ab} - \frac{a+b}{2} \left(\arcsin\left(\frac{-2x+a+b}{b-a}\right) \right) \right. \\
&\quad \left. - \sqrt{ab} \left(\arcsin\left(\frac{(a+b)x-2ab}{x(b-a)}\right) \right) \right) + \frac{1}{8c} (2\sqrt{c})^2, \\
&= \frac{1}{2\pi c} \left(\sqrt{-x^2 + (a+b)x - ab} - \frac{a+b}{2} \left(\arcsin\left(\frac{-2x+a+b}{b-a}\right) \right) \right. \\
&\quad \left. - \sqrt{ab} \left(\arcsin\left(\frac{(a+b)x-2ab}{x(b-a)}\right) \right) \right) + \frac{1}{2}.
\end{aligned}$$

□

Références

- [Abernethy *et al.* 2009] Jacob Abernethy, Francis Bach, Theodoros Evgeniou et Jean-Philippe Vert. *A new approach to collaborative filtering : Operator estimation with spectral regularization*. The Journal of Machine Learning Research (JMLR), vol. 10, pages 803–826, 2009. (cité en page 26.)
- [Adomavicius & Tuzhilin 2011] Gediminas Adomavicius et Alexander Tuzhilin. Recommender systems handbook, Chapter Context-aware recommender systems, pages 217–253. Springer, 2011. (cité en page 36.)
- [Aggarwal 2011] Charu C. Aggarwal. Social network data analytics, Chapter An introduction to social network data analytics, pages 1–15. Springer, 2011. (cité en page 8.)
- [Amatriain *et al.* 2009] Xavier Amatriain, Josep M. Pujol et Nuria Oliver. *I Like It... I Like It Not : Evaluating User Ratings Noise in Recommender Systems*. In Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP '09), pages 247–258, Berlin, Heidelberg, 2009. Springer-Verlag. (cité en page 35.)
- [Artz & Gil 2007] Donovan Artz et Yolanda Gil. *A survey of trust in computer science and the semantic web*. Web Semantics : Science, Services and Agents on the World Wide Web, vol. 5, no. 2, pages 58–71, 2007. (cité en page 47.)
- [Bai & Silverstein 2010] Zhidong Bai et Jack William Silverstein. Spectral analysis of large dimensional random matrices. Springer, 2010. (cité en page 76.)
- [Bell & Koren 2007] Robert M. Bell et Yehuda Koren. *Lessons from the Netflix prize challenge*. ACM SIGKDD Explorations Newsletter, vol. 9, no. 2, pages 75–79, 2007. (cité en page 7.)
- [Bell *et al.* 2009] Robert M. Bell, Yehuda Koren et Chris Volinsky. *The BellKor solution to the Netflix prize*. Netflix prize documentation, vol. 1, page 1, 2009. (cité en pages 7 et 26.)
- [Blei *et al.* 2003] David M. Blei, Andrew Y. Ng et Michael I. Jordan. *Latent dirichlet allocation*. The Journal of machine Learning research (JMLR), vol. 3, pages 993–1022, 2003. (cité en page 14.)
- [Boisbunon 2013] Aurélie Boisbunon. *Model selection : a decision-theoretic approach*. PhD thesis, Université de Rouen, 2013. (cité en pages 66, 67, 71 et 87.)
- [Bouveyron *et al.* 2011] Charles Bouveyron, Gilles Celeux et Stéphane Girard. *Intrinsic dimension estimation by maximum likelihood in isotropic probabilistic PCA*. Pattern Recognition Letters, vol. 32, no. 14, pages 1706–1713, 2011. (cité en page 68.)
- [Bradley & Mangasarian 1998] Paul S. Bradley et Olvi L. Mangasarian. *Feature selection via concave minimization and support vector machines*. In Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98), pages 82–90, 1998. (cité en page 17.)
- [Bruce & Gao 1996] Andrew G. Bruce et Hong-Ye Gao. *Understanding waveshrink : variance and bias estimation*. Biometrika, vol. 83, no. 4, pages 727–745, 1996. (cité en pages 18 et 71.)
- [Candès *et al.* 2011] Emmanuel J. Candès, Xiaodong Li, Yi Ma et John Wright. *Robust principal component analysis ?* Journal of the ACM (JACM), vol. 58, no. 3, pages 11 :1–11 :37, June 2011. (cité en pages 14, 15 et 17.)

- [Candès *et al.* 2012] Emmanuel J. Candès, Carlos A. Sing-Long et Joshua D Trzasko. *Unbiased risk estimates for singular value thresholding and spectral estimators*. arXiv preprint arXiv :1210.4139, 2012. (cité en pages 2, 68, 70, 71, 72, 73, 84 et 100.)
- [Carroll & Chang 1970] J. Douglas Carroll et Jih-Jie Chang. *Analysis of individual differences in multidimensional scaling via an N-way generalization of “Eckart-Young” decomposition*. *Psychometrika*, vol. 35, no. 3, pages 283–319, 1970. (cité en page 26.)
- [Castagnos & Boyer 2006] Sylvain Castagnos et Anne Boyer. *A client/server user-based collaborative filtering algorithm : Model and implementation*. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI’06)*, pages 617–621, 2006. (cité en page 9.)
- [Cattell 1966] Raymond B. Cattell. *The scree test for the number of factors*. *Multivariate behavioral research*, vol. 1, no. 2, pages 245–276, 1966. (cité en page 68.)
- [Cravo 2009] Glória Cravo. *Matrix completion problems*. *Linear Algebra and Its Applications*, vol. 430, no. 8, pages 2511–2540, 2009. (cité en page 15.)
- [Cremonesi *et al.* 2010] Paolo Cremonesi, Yehuda Koren et Roberto Turrin. *Performance of recommender algorithms on top-n recommendation tasks*. In *Proceedings of the fourth ACM Conference on Recommender Systems (RecSys’10)*, pages 39–46, 2010. (cité en page 57.)
- [Delporte *et al.* 2013] Julien Delporte, Alexandros Karatzoglou, Tomasz Matuszczyk et Stéphane Canu. *Socially Enabled Preference Learning from Implicit Feedback Data*. In *Proceedings of the 13th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD’13)*, pages 145–160. Springer, 2013. (cité en pages 3 et 45.)
- [Delporte *et al.* 2014] Julien Delporte, Alexandros Karatzoglou et Stéphane Canu. *Apprentissage et factorisation pour la recommandation*. *Revue des Nouvelles Technologies de l’Information (RNTI)*, vol. RNTI-A-6, 2014. (cité en page 3.)
- [Dias & Krzanowski 2003] Carlos T. dos Santos Dias et Wojtek J. Krzanowski. *Model selection and cross validation in additive main effect and multiplicative interaction models*. *Crop Science*, vol. 43, no. 3, pages 865–873, 2003. (cité en page 68.)
- [Ding *et al.* 2006] Yi Ding, Xue Li et Maria E Orlowska. *Recency-based collaborative filtering*. In *Proceedings of the 17th Australasian Database Conference (ADC’06)*, volume 49, pages 99–107. Australian Computer Society, Inc., 2006. (cité en page 36.)
- [DuBois *et al.* 2011] Thomas DuBois, Jennifer Golbeck et Aravind Srinivasan. *Predicting trust and distrust in social networks*. In *Proceedings of the third IEEE International Conference on Privacy, security, risk and trust (passat) and social computing (socialcom)*, pages 418–424. IEEE, 2011. (cité en page 47.)
- [Eastment & Krzanowski 1982] Henry T. Eastment et Wojtek J. Krzanowski. *Cross-validatory choice of the number of components from a principal component analysis*. *Technometrics*, vol. 24, no. 1, pages 73–77, 1982. (cité en page 68.)
- [Eckart & Young 1936] Carl Eckart et Gale Young. *The approximation of one matrix by another of lower rank*. *Psychometrika*, vol. 1, no. 3, pages 211–218, 1936. (cité en page 20.)
- [Fan & Li 2001] Jianqing Fan et Runze Li. *Variable selection via nonconcave penalized likelihood and its oracle properties*. *Journal of the American Statistical Association*, vol. 96, no. 456, pages 1348–1360, 2001. (cité en pages 18 et 72.)

- [Fang & Si 2011] Yi Fang et Luo Si. *Matrix co-factorization for recommendation with rich side information and implicit feedback*. In Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec'11), pages 65–69. ACM, 2011. (cité en page 30.)
- [Fang 2004] Kai-Tai Fang. *Elliptically contoured distributions*. Encyclopedia of Statistical Sciences, 2004. (cité en page 19.)
- [Févotte *et al.* 2009] Cédric Févotte, Nancy Bertin et Jean-Louis Durrieu. *Nonnegative matrix factorization with the itakura-saito divergence : With application to music analysis*. Neural Computation, vol. 21, no. 3, pages 793–830, 2009. (cité en pages 17 et 26.)
- [Fu *et al.* 2000] Xiaobin Fu, Jay Budzik et Kristian J Hammond. *Mining navigation history for recommendation*. In Proceedings of the 5th International Conference on Intelligent User Interfaces (IUI'00), pages 106–112. ACM, 2000. (cité en page 35.)
- [Golbeck & Hendler 2006] Jennifer Golbeck et James Hendler. *Inferring binary trust relationships in web-based social networks*. ACM Transactions on Internet Technology (TOIT), vol. 6, no. 4, pages 497–529, 2006. (cité en page 46.)
- [Golbeck *et al.* 2003] Jennifer Golbeck, Bijan Parsia et James Hendler. *Trust networks on the semantic web*. In Proceedings of Cooperative Information Agents VII (CIA'03), pages 238–249. Springer, 2003. (cité en page 46.)
- [Golub & Kahan 1965] Gene H. Golub et William Kahan. *Calculating the singular values and pseudo-inverse of a matrix*. Journal of the Society for Industrial & Applied Mathematics (SIAM), Series B : Numerical Analysis, vol. 2, no. 2, pages 205–224, 1965. (cité en page 20.)
- [Golub & Van Loan 1989] Gene H. Golub et Charles F. Van Loan. *Matrix computations*. JohnsHopkinsPress, Baltimore, MD, USA, second édition, 1989. (cité en page 22.)
- [Guha *et al.* 2004] Ramanathan Guha, Ravi Kumar, Prabhakar Raghavan et Andrew Tomkins. *Propagation of trust and distrust*. In Proceedings of the 13th international conference on World Wide Web (WWW'04), pages 403–412. ACM, 2004. (cité en page 47.)
- [Hansen *et al.* 1999] Lars Kai Hansen, Jan Larsen, Finn rArup Nielsen, Stephen C Strother, Egill Rosstrup, Robert Savoy, Nicholas Lange, John Sidtis, Claus Svarer et Olaf B Paulson. *Generalizable patterns in neuroimaging : How many principal components ?* NeuroImage, vol. 9, no. 5, pages 534–544, 1999. (cité en page 68.)
- [Harshman 1970] Richard A. Harshman. *Foundations of the PARAFAC procedure : models and conditions for an " explanatory " multimodal factor analysis*. UCLA Working Papers in Phonetics, 1970. (cité en page 27.)
- [He *et al.* 2009] Zhaoshui He, Andrzej Cichocki et Shengli Xie. *Efficient method for Tucker3 model selection*. Electronics letters, vol. 45, no. 15, pages 805–806, 2009. (cité en page 68.)
- [Herlocker *et al.* 1999] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers et John Riedl. *An algorithmic framework for performing collaborative filtering*. In Proceedings of the 22nd annual international ACM conference on Research and development in information retrieval (SIGIR'99), pages 230–237, New York, NY, USA, 1999. ACM. (cité en page 30.)
- [Herlocker *et al.* 2004] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen et John T. Riedl. *Evaluating collaborative filtering recommender systems*. ACM Transactions on Information Systems (TOIS), vol. 22, no. 1, pages 5–53, 2004. (cité en page 30.)

- [Hidasi & Tikk 2012] Balázs Hidasi et Domonkos Tikk. *Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback*. In Proceedings of the 12th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD'12), pages 67–82. Springer, 2012. (cité en page 28.)
- [Hiriart-Urruty & Lemarechal 1993] Jean-Baptiste Hiriart-Urruty et Claude Lemarechal. *Convex Analysis and Minimization Algorithms I : Fundamentals* (Grundlehren Der Mathematischen Wissenschaften). Springer, October 1993. (cité en page 18.)
- [Hitchcock 1927] Frank Lauren Hitchcock. *The expression of a tensor or a polyadic as a sum of products*. Massachusetts Institute of Technology, 1927. (cité en page 26.)
- [Hoff 2007] Peter D. Hoff. *Model averaging and dimension selection for the singular value decomposition*. Journal of the American Statistical Association (JASA), vol. 102, no. 478, 2007. (cité en page 68.)
- [Hoyer 2002] Patrik O. Hoyer. *Non-negative sparse coding*. In Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing (NNSP'02), pages 557–565. IEEE, 2002. (cité en page 25.)
- [Hu *et al.* 2008] Yifan Hu, Yehuda Koren et Chris Volinsky. *Collaborative filtering for implicit feedback datasets*. In Proceedings of the eighth IEEE International Conference on Data Mining (ICDM'08), pages 263–272. Ieee, 2008. (cité en pages 24, 30, 52, 57, 58 et 86.)
- [Jamali & Ester 2010] Mohsen Jamali et Martin Ester. *A matrix factorization technique with trust propagation for recommendation in social networks*. In Proceedings of the fourth ACM conference on Recommender systems (RecSys'10), pages 135–142, New York, NY, USA, 2010. ACM. (cité en page 50.)
- [Jolliffe 2002] Ian T. Jolliffe. *Principal component analysis*, second edition. Springer, 2002. (cité en page 68.)
- [Kaptein & Eckles 2010] Maurits Kaptein et Dean Eckles. *Selecting effective means to any end : futures and ethics of persuasion profiling*. In Proceedings of the 5th international conference on Persuasive Technology (PERSUASIVE'10), pages 82–93, Berlin, Heidelberg, 2010. Springer-Verlag. (cité en page 11.)
- [Karatzoglou *et al.* 2010] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas et Nuria Oliver. *Multiverse recommendation : n-dimensional tensor factorization for context-aware collaborative filtering*. In Proceedings of the fourth ACM conference on Recommender systems (RecSys'10), pages 79–86. ACM, 2010. (cité en pages 28 et 37.)
- [Kolda & Bader 2009] Tamara G. Kolda et Brett W. Bader. *Tensor decompositions and applications*. Journal of the Society for Industrial & Applied Mathematics (SIAM), vol. 51, no. 3, pages 455–500, 2009. (cité en page 28.)
- [Konno 1992] Yoshihiko Konno. *Improved estimation of matrix of normal mean and eigenvalues in the multivariate F-distribution*. PhD thesis, University of Tsukuba, 1992. (cité en page 70.)
- [Koren 2009] Yehuda Koren. *Collaborative filtering with temporal dynamics*. In Proceedings of the 15th ACM international conference on Knowledge discovery and data mining (SIGKDD'09), pages 447–456, 2009. (cité en pages 34 et 35.)
- [Kowalski 2009] Matthieu Kowalski. *Sparse regression using mixed norms*. Applied and Computational Harmonic Analysis (ACHA), vol. 27, no. 3, pages 303 – 324, 2009. (cité en page 18.)

- [Krohn-Grimberghe *et al.* 2012] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaier et Lars Schmidt-Thieme. *Multi-relational matrix factorization using bayesian personalized ranking for social network data*. In Proceedings of the fifth ACM international conference on Web Search and Data Mining (WSDM'12), pages 173–182, 2012. (cité en page 14.)
- [Kroonenberg & De Leeuw 1980] Pieter M Kroonenberg et Jan De Leeuw. *Principal component analysis of three-mode data by means of alternating least squares algorithms*. Psychometrika, vol. 45, no. 1, pages 69–97, 1980. (cité en page 27.)
- [Labiod & Nadif 2011a] Lazhar Labiod et Mohamed Nadif. *Co-Clustering for binary and categorical data with maximum modularity*. In Proceedings of the 11th International Conference on Data Mining (ICDM'11), pages 1140–1145. IEEE, 2011. (cité en page 68.)
- [Labiod & Nadif 2011b] Lazhar Labiod et Mohamed Nadif. *Co-clustering under Nonnegative Matrix Tri-Factorization*. Neural Information Processing, pages 709–717, 2011. (cité en pages 16 et 26.)
- [Larsen 1998] Rasmus Munk Larsen. *Lanczos bidiagonalization with partial reorthogonalization*. DAIMI Report Series, vol. 27, no. 537, 1998. (cité en pages 20 et 22.)
- [Lazarsfeld & Merton 1954] Paul F. Lazarsfeld et Robert K. Merton. *Friendship as a social process : A substantive and methodological analysis*. Freedom and control in modern society, vol. 18, no. 1, pages 18–66, 1954. (cité en page 46.)
- [Lee & Seung 1999] Daniel D. Lee et H. Sebastian Seung. *Learning the parts of objects by non-negative matrix factorization*. Nature, vol. 401, no. 6755, pages 788–791, 1999. (cité en page 24.)
- [Lee & Seung 2001] Daniel D. Lee et H. Sebastian Seung. *Algorithms for non-negative matrix factorization*. Advances in Neural Information Processing Systems (NIPS'01), vol. 13, pages 556–562, 2001. (cité en pages 17 et 26.)
- [Ma *et al.* 2008] Hao Ma, Haixuan Yang, Michael R. Lyu et Irwin King. *Sorec : social recommendation using probabilistic matrix factorization*. In Proceedings of the 17th ACM conference on Information and knowledge management (CIKM'08), pages 931–940. ACM, 2008. (cité en pages 38 et 49.)
- [Ma *et al.* 2009] Hao Ma, Irwin King et Michael R. Lyu. *Learning to recommend with social trust ensemble*. In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval (SIGIR'09), pages 203–210, New York, NY, USA, 2009. ACM. (cité en pages 50 et 58.)
- [Ma *et al.* 2011] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu et Irwin King. *Recommender systems with social regularization*. In Proceedings of the fourth ACM international conference on Web search and data mining (WSDM'11), pages 287–296, New York, NY, USA, 2011. ACM. (cité en pages 38, 49 et 58.)
- [Marchenko & Pastur 1967] Vladimir Alexandrovich Marchenko et Leonid Andreevich Pastur. *Distribution of eigenvalues for some sets of random matrices*. Matematicheskii Sbornik, vol. 114, no. 4, pages 507–536, 1967. (cité en page 76.)
- [Mehta 2004] Madan Lal Mehta. Random matrices, volume 142. Access Online via Elsevier, 2004. (cité en page 76.)
- [Milkman *et al.* 2009] Katherine L. Milkman, Todd Rogers et Max H. Bazerman. *Highbrow films gather dust : Time-inconsistent preferences and online DVD rentals*. Management Science, vol. 55, no. 6, pages 1047–1059, 2009. (cité en page 34.)

- [Minka 2000] Thomas P Minka. *Automatic choice of dimensionality for PCA*. In Advances in Neural Information Processing Systems (NIPS'00), volume 13, pages 598–604, 2000. (cité en page 68.)
- [Paige 1974] Christopher C. Paige. *Bidiagonalization of matrices and solution of linear equations*. Journal of the Society for Industrial & Applied Mathematics (SIAM), Series B : Numerical Analysis, vol. 11, no. 1, pages 197–209, 1974. (cité en page 21.)
- [Pan et al. 2008] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan Lukose, Martin Scholz et Qiang Yang. *One-class collaborative filtering*. In Proceedings of the eighth IEEE International Conference on Data Mining (ICDM'08), pages 502–511. IEEE, 2008. (cité en page 24.)
- [Peharz & Pernkopf 2012] Robert Peharz et Franz Pernkopf. *Sparse nonnegative matrix factorization with ℓ^0 -constraints*. Neurocomputing, vol. 80, pages 38–46, 2012. (cité en page 25.)
- [Penot 1984] Jean-Paul Penot. *Variations on the theme of nonsmooth analysis : another subdifferential*. In Non-Differentiable Optimization : Motivations and Applications, pages 8–24, 1984. (cité en page 18.)
- [Peres-Neto et al. 2005] Pedro R. Peres-Neto, Donald A. Jackson et Keith M. Somers. *How many principal components? Stopping rules for determining the number of non-trivial axes revisited*. Computational Statistics & Data Analysis (CSDA), vol. 49, no. 4, pages 974–997, 2005. (cité en page 68.)
- [Pessiot et al. 2006] Jean-François Pessiot, Vinh Truong, Nicolas Usunier, Massih Amini et Patrick Gallinari. *Factorisation en matrices non-négatives pour le filtrage collaboratif*. In CORIA, pages 315–326, 2006. (cité en page 26.)
- [Piotte & Chabbert 2009] Martin Piotte et Martin Chabbert. *The pragmatic theory solution to the netflix grand prize*. Netflix prize documentation, vol. 1, page 2, 2009. (cité en page 7.)
- [Pradel et al. 2011] Bruno Pradel, Savaneary Sean, Julien Delporte, Sébastien Guérif, Céline Rouveirol, Nicolas Usunier, Françoise Fogelman-Soulié et Frédéric Dufau-Joel. *A case study in a recommender system based on purchase data*. In Proceedings of the 17th ACM international conference on Knowledge discovery and data mining (SIGKDD'11), pages 377–385. ACM, 2011. (cité en pages 3 et 38.)
- [Pradel 2013] Bruno Pradel. *Évaluation des systèmes de recommandation à partir d'historiques de données*. PhD thesis, Université Pierre et Marie Curie (UPMC), 2013. Type : Thèse de Doctorat – Soutenue le : 2013-10-02 – Dirigée par : Gallinari, Patrick – Encadrée par : USUNIER Nicolas. (cité en pages 10 et 30.)
- [Pu et al. 2012] Pearl Pu, Li Chen et Rong Hu. *Evaluating recommender systems from the user's perspective : survey of the state of the art*. User Modeling and User-Adapted Interaction (UMUAI), vol. 22, pages 317–355, 2012. (cité en page 10.)
- [Recht et al. 2010] Benjamin Recht, Maryam Fazel et Pablo A. Parrilo. *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*. Journal of the Society for Industrial & Applied Mathematics (SIAM), vol. 52, no. 3, pages 471–501, 2010. (cité en page 17.)
- [Rendle & Lars 2010] Steffen Rendle et Schmidt-Thie Lars. *Pairwise interaction tensor factorization for personalized tag recommendation*. In Proceedings of the third ACM international conference on Web search and data mining (WSDM'10), pages 81–90, New York, NY, USA, 2010. ACM. (cité en page 30.)
- [Rendle et al. 2009] Steffen Rendle, Leandro Balby Marinho, Alexandros Nanopoulos et Schmidt-Thie Lars. *Learning optimal ranking with tensor factorization for tag recommendation*. In Procee-

- dings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '09, pages 727–736, New York, NY, USA, 2009. ACM. (cité en pages 28 et 30.)
- [Rendle *et al.* 2011] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler et Lars Schmidt-Thieme. *Fast context-aware recommendations with factorization machines*. In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, pages 635–644. ACM, 2011. (cité en pages 28 et 87.)
- [Rendle 2010] Steffen Rendle. Context-aware ranking with factorization models, volume 330. Springer, 2010. (cité en pages 37 et 38.)
- [Rendle 2012] Steffen Rendle. *Factorization Machines with libFM*. ACM Transactions on Intelligent Systems and Technology (TIST), vol. 3, no. 3, pages 57 :1–57 :22, May 2012. (cité en page 28.)
- [Ricci *et al.* 2011] Francesco Ricci, Lior Rokach et Bracha Shapira. *Introduction to recommender systems handbook*. In Recommender Systems Handbook, volume 1, pages 1–35. Springer, 2011. (cité en pages 8, 10 et 16.)
- [Richard *et al.* 2010] Emile Richard, Nicolas Baskiotis, Theodoros Evgeniou et Nicolas Vayatis. *Link discovery using graph feature tracking*. In Advances in Neural Information Processing Systems (NIPS'10), pages 1966–1974, 2010. (cité en page 35.)
- [Richardson *et al.* 2003] Matthew Richardson, Rakesh Agrawal et Pedro Domingos. *Trust management for the semantic web*. In Proceedings in the second International Semantic Web Conference (ISWC'03), pages 351–368. Springer, 2003. (cité en page 46.)
- [Salakhutdinov & Mnih 2008] Ruslan Salakhutdinov et Andriy Mnih. *Bayesian probabilistic matrix factorization using Markov chain Monte Carlo*. In Proceedings of the 25th International Conference on Machine learning (ICML'08), pages 880–887, New York, NY, USA, 2008. ACM. (cité en page 14.)
- [Sarwar *et al.* 2001] Badrul Sarwar, George Karypis, Joseph Konstan et John Riedl. *Item-based collaborative filtering recommendation algorithms*. In Proceedings of the 10th international conference on World Wide Web (WWW'01), pages 285–295. ACM, 2001. (cité en page 9.)
- [Schaul *et al.* 2012] Tom Schaul, Sixin Zhang et Yann LeCun. *No more pesky learning rates*. arXiv preprint arXiv :1206.1106, 2012. (cité en page 24.)
- [Schwarz *et al.* 1978] Gideon Schwarz *et al.* *Estimating the dimension of a model*. The Annals of Statistics, vol. 6, no. 2, pages 461–464, 1978. (cité en page 66.)
- [Seghouane & Cichocki 2007] Abd-Krim Seghouane et Andrzej Cichocki. *Bayesian estimation of the number of principal components*. Signal Processing, vol. 87, no. 3, pages 562–568, 2007. (cité en page 68.)
- [Shi *et al.* 2010] Yue Shi, Martha Larson et Alan Hanjalic. *List-wise learning to rank with matrix factorization for collaborative filtering*. In Proceedings of the fourth ACM conference on Recommender systems (RecSys'10), pages 269–272. ACM, 2010. (cité en page 30.)
- [Shi *et al.* 2012] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Alan Hanjalic et Nuria Oliver. *TFMAP : Optimizing MAP for top-n context-aware recommendation*. In Proceedings of the 35th international ACM conference on Research and development in information retrieval (SIGIR'12), pages 155–164. ACM, 2012. (cité en pages 17 et 28.)
- [Singh & Gordon 2008] Ajit P. Singh et Geoffrey J. Gordon. *A Unified View of Matrix Factorization Models*. In Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases (ECMLPKDD'08), pages 358–373. Springer-Verlag, 2008. (cité en page 19.)

- [Sra 2011] Suvrit Sra. *Nonconvex proximal splitting : batch and incremental algorithms*. arXiv pre-print arXiv:1109.0258, 2011. (cité en page 18.)
- [Srebro & Shraibman 2005] Nathan Srebro et Adi Shraibman. *Rank, trace-norm and max-norm*. In Proceedings of the 18th annual Conference on Learning Theory (COLT'05), pages 545–560. Springer-Verlag, 2005. (cité en page 17.)
- [Srebro *et al.* 2005] Nathan Srebro, Jason Rennie et Tommi Jaakkola. *Maximum-Margin Matrix Factorization*. In Advances in Neural Information Processing Systems (NIPS'05), Cambridge, MA, 2005. MIT Press. (cité en pages 14 et 22.)
- [Stein 1973] Charles M. Stein. *Estimation of the mean of a multivariate normal distribution*. Technical report number 48, 1973. (cité en page 70.)
- [Takacs *et al.* 2009] Gabor Takacs, Istvan Pilaszy, Bottyan Nemeth et Domonkos Tikk. *Scalable Collaborative Filtering Approaches for Large Recommender Systems*. The Journal of Machine Learning Research (JMLR), vol. 10, pages 623–656, 2009. (cité en pages 14 et 23.)
- [Töscher *et al.* 2009] Andreas Töscher, Michael Jahrer et Robert M. Bell. *The BigChaos solution to the Netflix grand prize*. Netflix prize documentation, vol. 1, page 3, 2009. (cité en page 7.)
- [Tucker 1963] Ledyard R. Tucker. *Implications of factor analysis of three-way matrices for measurement of change*. Problems in measuring change, pages 122–137, 1963. (cité en page 27.)
- [Ulfarsson & Solo 2008] Magnus O. Ulfarsson et Victor Solo. *Dimension estimation in noisy PCA with SURE and random matrix theory*. IEEE Transactions on Signal Processing, vol. 56, no. 12, pages 5804–5816, 2008. (cité en page 68.)
- [Valizadegan *et al.* 2009] Hamed Valizadegan, Rong Jin, Ruofei Zhang et Jianchang Mao. *Learning to rank by optimizing ndcg measure*. Advances in Neural Information Processing Systems (NIPS'09), vol. 22, pages 1883–1891, 2009. (cité en page 16.)
- [Wachter 1978] Kenneth W. Wachter. *The strong limits of random matrix spectra for sample matrices of independent elements*. The Annals of Probability, pages 1–18, 1978. (cité en page 76.)
- [Wang *et al.* 2006] Jun Wang, Arjen P. De Vries et Marcel J.T. Reinders. *Unifying user-based and item-based collaborative filtering approaches by similarity fusion*. In Proceedings of the 29th annual international ACM conference on Research and development in information retrieval (SIGIR'06), pages 501–508. ACM, 2006. (cité en page 9.)
- [Weimer *et al.* 2008a] Markus Weimer, Alexandros Karatzoglou, Quoc V. Le et Alex Smola. *COFIRANK - Maximum Margin Matrix Factorization for Collaborative Ranking*. In Advances in Neural Information Processing Systems (NIPS'08), Cambridge, MA, 2008. MIT Press. (cité en page 14.)
- [Weimer *et al.* 2008b] Markus Weimer, Alexandros Karatzoglou et Alex Smola. *Improving Maximum Margin Matrix Factorization*. The Journal Machine Learning Research (JMLR), vol. 72, no. 3, pages 263–276, September 2008. (cité en page 22.)
- [Wermser *et al.* 2011] Hendrik Wermser, Achim Rettinger et Volker Tresp. *Modeling and learning context-aware recommendation scenarios using tensor decomposition*. In Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM'11), pages 137–144. IEEE, 2011. (cité en page 37.)
- [Weston *et al.* 2003] Jason Weston, André Elisseeff, Bernhard Schölkopf et Mike Tipping. *Use of the zero norm with linear models and kernel methods*. The Journal of Machine Learning Research (JMLR), vol. 3, pages 1439–1461, 2003. (cité en page 18.)

- [Xu *et al.* 2008] Jun Xu, Tie-Yan Liu, Min Lu, Hang Li et Wei-Ying Ma. *Directly optimizing evaluation measures in learning to rank*. In Proceedings of the 31st annual international ACM conference on Research and development in information retrieval (SIGIR'08), pages 107–114. ACM, 2008. (cité en page 30.)
- [Yang *et al.* 2011] Shuang-Hong Yang, Bo Long, Alex Smola, Narayanan Sadagopan, Zhaohui Zheng et Hongyuan Zha. *Like like alike : joint friendship and interest propagation in social networks*. In Proceedings of the 20th International Conference on World Wide Web (WWW'11), pages 537–546, New York, NY, USA, 2011. ACM. (cité en pages 30, 38, 48, 49, 50, 58, 86 et 87.)
- [Ye 1998] Jianming Ye. *On measuring and correcting the effects of data mining and model selection*. Journal of the American Statistical Association, vol. 93, no. 441, pages 120–131, 1998. (cité en page 81.)
- [Young *et al.* 1976] Forrest W. Young, Jan De Leeuw et Yoshio Takane. *Regression with qualitative and quantitative variables : An alternating least squares method with optimal scaling features*. Psychometrika, vol. 41, no. 4, pages 505–529, 1976. (cité en page 24.)
- [Zheng *et al.* 2011] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie et Wei-Ying Ma. *Recommending friends and locations based on individual location history*. ACM Transactions on the Web (TWEB), vol. 5, no. 1, page 5, 2011. (cité en pages 9, 35 et 36.)
- [Zou 2006] Hui Zou. *The adaptive lasso and its oracle properties*. Journal of the American statistical association (JASA), vol. 101, no. 476, pages 1418–1429, 2006. (cité en pages 18 et 72.)