Filtrage Collaboratif avec un Algorithme d'Ordonnancement

Jean-François Pessiot, Vinh Truong, Nicolas Usunier, Massih-Reza Amini, Patrick Gallinari

Laboratoire d'Informatique de Paris 6
104 avenue du Président Kennedy, 75016 Paris
{pessiot, truong, usunier, amini, gallinari}@poleia.lip6.fr

RÉSUMÉ. *A ce jour, la plupart des travaux en filtrage collaboratif se basent sur la prédiction de notes pour générer des recommandations. Dans ce papier, nous choisissons d'explorer une autre voie, consistant à ordonner correctement les articles selon les goûts des utilisateurs. D'abord, nous définissons une erreur d'ordonnancement qui prend en compte les préférences par paires d'articles. Puis nous construisons un algorithme efficace qui optimise cette erreur. Enfin, nous testons notre approche sur une base standard de filtrage collaboratif. Pour cela, nous adaptons le protocole d'évaluation initialement proposé par (Marlin, 2004a) pour la prédiction de notes à notre cas, où ce sont des préférences par paires qui sont prédites. Nous comparons nos performances avec celles de deux méthodes basées sur la prédiction de notes. Nous suggérons différentes directions pour continuer l'exploration de notre approche basée sur la prédiction d'ordres pour le filtrage collaboratif.*

ABSTRACT. *Up to now, most contributions to collaborative filtering rely on rating prediction to generate the recommendations. We, instead, try to correctly rank the items according to the users' tastes. First, we define a ranking error function which takes available pairwise preferences between items into account. Then we design an effective algorithm that optimizes this error. Finally we illustrate the proposal on a standard collaborative filtering dataset. We adapted the evaluation protocol proposed by (Marlin, 2004a) for rating prediction based systems to our case, where pairwise preferences are predicted instead. The preliminary results are between those of two reference rating prediction based methods. We suggest different directions to further explore our ranking based approach for collaborative filtering.*

MOTS-CLÉS : *Filtrage Collaboratif, Systèmes de Recommandation, Apprentissage Statistique, Ordonnancement.*

KEYWORDS: *Collaborative Filtering, Recommender Systems, Machine Learning, Ranking.*

CORIA 2007, pp. 165–180
Saint-Étienne, 28–30 mars 2007

1. INTRODUCTION

Avec le développement du commerce électronique, les internautes se voient proposer un choix grandissant de produits et de services en ligne. Pour les guider, la plupart des sites utilisent des systèmes de recommandation. Leur but est de générer des recommandations personnalisées, c'est à dire de déterminer pour chaque utilisateur les produits ou articles les plus susceptibles de l'intéresser. Pour y parvenir, les implémentations les plus efficaces à ce jour utilisent les préférences des autres utilisateurs pour générer ces recommandations : c'est le principe du filtrage collaboratif. Le filtrage collaboratif est particulièrement adapté pour recommander des produits culturels comme des films, des livres ou de la musique, et est utilisé avec succès par des systèmes de recommandation commerciaux en ligne comme Amazon.com ou CDnow.com.

Une manière simple et naturelle de modéliser les préférences d'un utilisateur est d'associer à chaque article un score numérique mesurant à quel point il apprécie cet article. Tous les articles sont ensuite ordonnés selon ces scores, de ses articles favoris à ceux qui l'intéressent le moins. Dans la formulation standard du filtrage collaboratif, les scores sont des notes entières de 1 à 5. Chaque utilisateur a noté quelques articles, les autres notes étant donc inconnues. La plupart des méthodes de filtrage collaboratif basent leur approche sur la prédiction de notes : elles prennent toutes les notes disponibles en entrée, leur but étant de prédire les notes inconnues en sortie. La recommandation se fait simplement en présentant à chaque utilisateur les articles non notés dont les prédictions sont les plus élevées.

Grâce à sa simplicité et à l'existence de protocoles d'évaluation facilitant les comparaisons de performances, l'approche basée sur la prédiction de notes est la plus étudiée en filtrage collaboratif. La littérature fait état de nombreux travaux d'inspirations diverses, empruntant à la classification supervisée, non supervisée, à la régression, à la réduction dimensionnelle ou encore aux modèles probabilistes. En ramenant le problème de la recommandation à une tâche de prédiction de notes, toutes ces méthodes ont un but commun : prédire les notes le plus précisément possible. Pourtant, du point de vue de la recommandation, la manière dont les articles sont ordonnés est plus importante que les notes elles-mêmes. En effet, un utilisateur attend avant tout d'un système de recommandation qu'il lui suggère les articles les plus intéressants, la prédiction des notes manquantes n'étant qu'une stratégie parmi d'autres pour y parvenir. L'approche que nous proposons est basée sur la prédiction d'ordre : plutôt que d'essayer de prédire des notes le mieux possible, nous prédisons des scores respectant au mieux les préférences par paires d'articles, c'est à dire les préférences indiquant qu'un article est préféré à un autre. En utilisant ces préférences, notre but est d'améliorer la qualité de la recommandation.

Cet article est organisé de la façon suivante. Dans la section 2, nous faisons un état de l'art des principales approches pour la prédiction de notes en filtrage collaboratif. Dans la section 3, nous présentons notre approche, puis détaillons notre algorithme et son implantation. Le protocole expérimental et les résultats sont donnés dans la

section 4. Enfin, nous concluons et donnons les perspectives d'amélioration dans la section 5.

2. Etat de l'Art

Le filtrage collaboratif est un domaine de recherche très actif depuis plusieurs années. Parmi les solutions proposées, nombreuses sont les méthodes à variables latentes : elles font l'hypothèse que les comportements des utilisateurs (dont nous ne connaissons que la partie visible à travers les notes qu'ils ont fournies) peuvent être expliqués par un petit nombre de facteurs cachés qui représentent des comportements type d'utilisateurs. Chaque utilisateur de la base est alors vu comme une combinaison de ces comportements type, et l'apprentissage est utilisé pour identifier ces derniers dans le but de faire des prédictions pour les articles non notés. Dans cette section, nous commençons par faire une présentation générale de deux grandes familles de méthodes à variables latentes : les méthodes de factorisation matricielle et les modèles probabilistes. Nous présentons également les approches adaptant l'ordonnancement d'instances au problème du filtrage collaboratif. Toutes ces méthodes ont le même objectif commun : la prédiction des notes manquantes.

2.1. *Factorisation Matricielle*

Les approches à base de factorisation matricielle pour le filtrage collaboratif reposent sur la décomposition de matrices comportant des "trous", c'est à dire que les valeurs de certaines entrées sont inconnues. Ces trous correspondent par exemple aux notes manquantes que l'on veut prédire. Les méthodes de factorisation classiques n'étant capables de factoriser que des matrices pleines, la difficulté principale consiste à les adapter au cas de données manquantes. Une fois calculée, la factorisation permet de reconstruire une matrice pleine, contenant les entrées initiales ainsi que les prédictions pour les entrées manquantes.

Dans la suite, R est une matrice de réels $(n \times p)$, k est un entier positif tel que $k < np$, et W est une matrice $(n \times p)$ de réels positifs où W_{ij} est un coefficient associé à R_{ij}. La norme de Frobenius de R vaut : $\|R\|^2 = \sum_{ij} R_{ij}^2$. Dans sa formulation classique, le but de la factorisation matricielle est de trouver l'approximation de R optimale pour la norme de Frobenius sous la contrainte de rang k , c'est à dire de trouver la matrice $\hat{R}$ de rang k minimisant $\|R - \hat{R}\|^2$. La Décomposition en Valeurs Singulières (DVS) est la méthode la plus connue pour y parvenir. Dans le cas particulier où la matrice à factoriser et les matrices résultantes ne doivent contenir que des réels positifs, on décompose R en utilisant la Factorisation en Matrices Non-Négatives (FMN) proposée par (Lee *et al.,* 1999).

La plupart des méthodes de factorisation matricielle étant incapables de prendre en compte des éléments manquants dans R, de récents travaux proposent d'optimiser la norme de Frobenius pondérée plutôt que la norme de Frobenius classique, c'est à

dire : $\|W \odot (R - \hat{R})\|^2$, où $\odot$ est le produit de schur élément par élément. Les éléments manquants R_{ij} sont simplement pris en compte en fixant les coefficients W_{ij} correspondants à 0. Ainsi, des variantes pondérées de la DVS et de la FMN ont récemment été proposées : la DVS pondérée de (Srebro *et al.,* 2003) et la FMN généralisée de (Dhillon *et al.,* 2006). Pour ces deux méthodes, l'application au filtrage collaboratif est immédiate. Les matrices à factoriser sont les matrices de notes $(n \times p)$ où n est le nombre d'utilisateurs, p le nombre d'articles. Les coefficients W_{ij} sont fixés à 1 si la note R_{ij} est connue, 0 sinon. Les notes inconnues de R sont initialisées aléatoirement. Elles seront prédites par la matrice $\hat{R}$ de rang k issue de la factorisation : la note prédite pour l'utilisateur i et l'article j est simplement $\hat{R}_{ij}$. Notons que la DVS pondérée et la FMN généralisée offrent des performances similaires en terme de prédiction de notes, mais que la deuxième est meilleure en terme de complexité algorithmique.

Citons également les travaux de (Rennie *et al.,* 2005a) sur la Factorisation Matricielle Maximisant la Marge (FMMM). Les auteurs proposent d'apprendre une matrice de scores S de taille $(n \times p)$ et de rang k, ainsi qu'une matrice de seuils Θ de taille $(n \times (k-1))$. La note prédite pour l'utilisateur i et l'article j ne dépend que de la position du score S_{ij} par rapport aux seuils $\{\theta_{ir}\}_{r=1..k-1}$. La FMMM se place dans une optique légèrement différente de celles des méthodes de factorisation précédentes, puisqu'elle ne tente pas d'approximer directement la matrice de notes R. Notons enfin que cette méthode est la plus performante de la littérature à ce jour en terme de prédiction de notes sur des bases standard de filtrage collaboratif (DeCoste, 2006), ce qui nous laisse penser que l'apprentissage de scores relatifs plutôt que de notes est une voie intéressante à explorer.

2.2. *Modèles Probabilistes*

De nombreux modèles probabilistes ont été proposés pour le filtrage collaboratif. Les plus aboutis modélisent les comportements des utilisateurs à l'aide d'une variable latente pouvant prendre K valeurs, chaque valeur correspondant à un comportement type. Dans la suite, Z est une variable latente dans $\{1, ..., K\}$, U et u sont une variable et un indice d'utilisateurs, Y et y sont une variable et un indice d'articles, R et r sont une variable note et une note dans $\{1, ..., V\}$. Avec ces notations, chaque utilisateur u est représenté par une distribution $p(Z|U = u)$, modélisant à quel point chaque comportement type intervient dans le comportement de u. A chaque comportement type z et à chaque item y est associée une distribution multinomiale $p(R|Z = z, Y = y)$ qui modélise les goûts du comportement type z pour l'article y. D'un point de vue génératif, la note attribuée par l'utilisateur u à l'article y est générée de la façon suivante : un comportement type z est choisi suivant la distribution $p(Z|U = u)$, puis la note r est choisie suivant la distribution $p(R|Z = z, Y = y)$. Lorsque les distributions sont connues, la prédiction de notes pour l'utilisateur u et l'article y consiste à calculer les $p(R = r|U = u, Y = y)$ pour toutes les notes r, et à prédire la note médiane $\hat{r}$ telle que $\hat{r} = \{r|p(R = r|U = u, Y = y) \leq 1/2, p(R = r|U = u, Y = y) \geq 1/2\}$.

Il existe plusieurs modèles probabilistes basés sur les hypothèses précédentes. (Hofmann, 2004) a d'abord proposé le modèle d'aspects, dans lequel il considère les distributions $p(Z|U=u)$ et $p(R|U=u,Y=y)$ comme les paramètres du modèle à apprendre. L'apprentissage se fait à l'aide d'un algorithme EM sur le principe du maximum de vraisemblance. Mais le fait de considérer ces distributions comme des paramètres pose problème : les $p(Z|U=u)$ ne peuvent être déterminés que pour les utilisateurs qui étaient présents au moment de l'apprentissage. Le modèle d'aspects ne peut donc pas faire de prédictions pour de nouveaux utilisateurs, ce qui rend impossible son utilisation dans un cadre en ligne. Une autre conséquence indésirable est que le nombre de paramètres augmente avec le nombre d'utilisateurs, augmentant ainsi le temps de l'apprentissage. Une réponse possible à ces problèmes est de considérer les distributions $p(Z|U=u)$ non plus comme des paramètres mais comme des variables aléatoires suivant des distributions paramétrées. Dans le cas particulier de distributions de Dirichlet, on se retrouve avec le modèle URP proposé par (Marlin, 2004b). Mentionnons également le modèle d'attitudes de (Marlin, 2004a), qui est une amélioration du modèle URP permettant aux différents comportements type d'interagir lors de la génération des notes pour chaque article. En terme de prédiction de notes, le modèle URP et le modèle d'attitudes sont les approches les plus performantes à ce jour parmi les modèles probabilistes (Marlin, 2004a).

Remarquons que le problème de la prédiction de notes en filtrage collaboratif est étroitement lié à celui de la classification non supervisée d'un ensemble de documents textuels. En effet, de la même manière qu'on tente en filtrage collaboratif d'extraire K comportements type d'un ensemble d'utilisateurs et d'articles notés, on tente en texte d'extraire K thématiques d'un ensemble de documents (ou paragraphes) et d'occurrences de mots (Caillet *et al.,* 2004). D'ailleurs ces deux formulations sont si proches que le modèle d'aspects et URP sont des adaptations directes de deux méthodes standard pour la classification non supervisée de textes : l'approche pLSA de (Hofmann, 1999) et LDA de (Blei *et al.,* 2003). Signalons également les travaux de (Gaussier *et al.,* 2005) montrant les liens étroits qui unissent le modèle probabiliste pLSA et l'approche FMN en factorisation matricielle, confirmant un peu plus l'intérêt d'une étude approfondie des liens entre le problème de la prédiction de notes en filtrage collaboratif et celui de la classification non supervisée de textes.

2.3. *Ordonnancement d'Instances*

En ordonnancement d'instances (également appelé régression ordinale), on dispose d'un ensemble d'exemples dans $\mathbb{R}^d$ et d'un ensemble ordonné d'étiquettes (ou rangs) $Y = \{1, ..., k\}$. Le but est de trouver une fonction $h : \mathbb{R}^d \rightarrow Y$ qui associe à chaque exemple un rang de Y. Pour le filtrage collaboratif, chaque utilisateur est considéré comme un seul problème d'ordonnancement. Les exemples sont les articles et les étiquettes sont les notes de 1 à 5. La base d'apprentissage est constituée des couples (article, note) fournis par l'utilisateur. Remarquons que les représentations vectorielles des articles, nécessaires pour apprendre ces ordonnancements, ne sont gé-

néralement pas disponibles en filtrage collaboratif. Une solution proposée consiste à utiliser les notes des autres utilisateurs de la base pour représenter les articles. L'ordonnancement d'instances permet alors d'assigner à chaque article non noté un rang de 1 à 5, et s'apparente donc aux méthodes qui prédisent les notes pour faire la recommandation.

(Crammer *et al.,* 2002) ont proposé une approche pour l'ordonnancement d'instances basée sur le perceptron, baptisée PRank. Les paramètres appris par PRank sont un vecteur de poids $w \in \mathbb{R}^d$ et un vecteur de seuils $\theta \in \mathbb{R}^{k-1}$. A chaque exemple $x \in \mathbb{R}^d$ est associé le score $\langle w, x \rangle$. Comme pour l'approche FMMM en factorisation matricielle, le rang prédit pour x ne dépend que de la position du score par rapport aux seuils $\{\theta_r\}_{r=1..k-1}$. Grâce à sa parenté avec le perceptron, PRank a l'avantage d'être une approche en ligne : chaque fois qu'un nouvel exemple est présenté, les vecteurs w, θ sont mis à jour de telle sorte que les erreurs sur les rangs soient les plus faibles possibles. Les auteurs mettent en avant la faible complexité de PRank par rapport à la plupart des méthodes existantes pour l'ordonnancement d'instances, permettant ainsi à PRank d'être applicable à de grosses bases comme celles qui existent en filtrage collaboratif. Notons qu'une telle approche n'exploite pas l'aspect collaboratif des données, puisque les notes des autres utilisateurs n'interviennent que pour fournir les représentations vectorielles des articles. (Yu *et al.,* 2006) ont proposé une autre approche basée sur l'ordonnancement d'instances. Les auteurs supposent l'existence, pour chaque utilisateur, d'une fonction latente expliquant son comportement, c'est à dire la façon dont il note les articles. Ils modélisent les relations entre les comportements d'utilisateurs à l'aide de processus gaussiens, dont les matrices de covariance servent à modéliser les dépendances entre les différentes fonctions latentes.

2.4. *Conclusion*

Toutes les approches présentées dans cette section ont le même objectif : la prédiction de notes. Les méthodes à variables latentes nous paraissent intéressantes, car elles cherchent à implémenter l'idée intuitive que les comportements utilisateurs d'une base peuvent être expliqués par un petit nombre de facteurs cachés. En réduisant ainsi la dimension du problème, elles permettent de découvrir des comportements type d'utilisateurs, facilitant ainsi la compréhension et l'exploration des comportements des utilisateurs (Pessiot *et al.,* 2006). En particulier, l'approche matricielle permet d'apprendre explicitement les représentations vectorielles des utilisateurs et des articles, ce qui peut être utile pour la fouille de données ou encore l'apprentissage actif. Il nous a également semblé que l'approche consistant à approximer directement les notes était une tâche potentiellement difficile. Aussi l'idée consistant à apprendre des scores et des seuils comme le font (Rennie *et al.,* 2005a) avec la FMMM et (Crammer *et al.,* 2002) avec PRank, nous a paru particulièrement intéressante. Et si eux l'utilisent dans l'optique de la prédiction de notes, nous pensons que ces scores pourraient être directement utilisés pour ordonner les articles selon les goûts des utilisateurs. Par manque de place, nous nous somme limités à présenter dans cette section trois grandes familles

d'approches pour la prédiction de notes ; le lecteur intéressé pourra appronfondir les problématiques générales liées au filtrage collaboratif en consultant la revue spéciale ACM qui lui est consacrée[1].

3. ORDONNANCEMENT POUR LE FILTRAGE COLLABORATIF

3.1. *Motivation*

La plupart des travaux proposés dans la littérature du filtrage collaboratif ont une approche commune, dans laquelle le processus de la recommandation est décomposé en deux étapes : la prédiction de notes et la recommandation elle-même. Bien sûr, une fois que les notes sont prédites, la recommandation se fait simplement en ordonnant les articles selon leurs prédictions et en suggérant à chaque utilisateur les articles dont les prédictions sont les plus élevées. En ramenant ainsi le problème de la recommandation à une tâche de prédiction de notes, toutes ces méthodes partagent un objectif commun : prédire les notes aussi précisément que possible. Un tel but parait naturel pour faire de la recommandation, et il a été le sujet de nombreux travaux de recherche en filtrage collaboratif (Marlin, 2004a). La formulation du problème en terme de prédiction de notes est simple et facilite les comparaisons de performances au niveau de l'évaluation. Cependant, il est important de noter que la prédiction de notes n'est qu'une étape intermédiaire vers la recommandation, et que d'autres voies sont envisageables.

En particulier, compte tenu de l'utilisation typique des systèmes de recommandation où le système présente à chaque utilisateur les N meilleurs articles sans montrer les notes associées, nous pensons qu'ordonner correctement les articles est plus important que prédire correctement leurs notes. Bien que ces deux objectifs soient proches, ils ne sont pas équivalents du point du vue de la recommandation. En effet, n'importe quelle méthode prédisant correctement toutes les notes ordonnera aussi correctement tous les articles. En revanche, à performances égales en terme de prédiction de notes, deux méthodes peuvent avoir des performances différentes en terme de prédiction d'ordres. Illustrons ce phénomène sur un exemple simple. Soient $[2, 3]$ les notes de deux articles A et B, $r_1 = [2.5, 3.6]$ et $r_2 = [2.5, 2.4]$ deux vecteurs de prédictions obtenus par deux méthodes différentes. Bien que r_1 et r_2 soient équivalents en terme d'erreur carrée[2] (les deux sont égales à $0.5^2 + 0.6^2$), seule r_1 prédit l'ordre correctement, puisque le score qu'elle attribue à B est supérieur à celui de A.

Dans cet article, nous proposons une alternative à l'approche traditionnelle basée sur la prédicion de notes : notre but est d'ordonner correctement les articles plutôt que de prédire correctement leurs notes. Pour y parvenir, nous définissons d'abord une erreur d'ordonnancement qui pénalise les mauvaises prédictions sur les préférences

1. ACM Transactions on Information Systems, Vol. 22, No. 1, Janvier 2004
2. C'est la somme des différences carrées entre les notes et leurs prédictions.

par paires d'articles. Puis nous proposons notre modèle, et construisons un algorithme efficace pour minimiser l'erreur d'ordonnancement.

3.2. *Définition et Notation*

Définition Considérons n utilisateurs et p articles, chaque utilisateur ayant noté au moins un article. Un exemple d'apprentissage est un triplet (x, a, r) où x est un indice d'utilisateur dans $\mathcal{X} = \{1, \ldots, n\}$, a est un indice d'article $\mathcal{Y} = \{1, \ldots, p\}$ et r est une note dans $\mathcal{V} = \{1, \ldots, v\}$. A partir des notes fournies par chaque utilisateur, on construit l'ensemble de préférences par paires $y_x = \{(a, b) | a \in \mathcal{Y}, b \in \mathcal{Y}, r_x(a) > r_x(b)\}$, où $r_x(a)$ est la note fournie par l'utilisateur x pour l'article a. Chaque préférence par paire $(a, b) \in y_x$ exprime que l'utilisateur x préfère l'article a à l'article b.

Fonctions d'Utilité Les fonctions d'utilité permettent de représenter les préférences de manière simple et intuitive. Une fonction d'utilité $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ modélise la préférence d'un utilisateur pour un article par un score réel. Si un utilisateur $x \in \mathcal{X}$ préfère un article a à un article b, cette préférence est simplement représentée par l'inégalité $f(x, a) > f(x, b)$. Dans le cadre des systèmes de recommandation, les articles sont présentés à chaque utilisateur par ordre décroissant des scores $f(x, .)$.

Nous faisons l'hypothèse que la fonction d'utilité f s'écrit sous la forme d'un produit scalaire : $\forall (x, a) \in \mathcal{X} \times \mathcal{Y}, f(x, a) = \langle i_a, u_x \rangle$, où $i_a \in \mathbb{R}^k$ et $u_x \in \mathbb{R}^k$ sont respectivement les vecteurs représentant l'article a et l'utilisateur x. En notant U et I les matrices $(n \times k)$ et $(k \times p)$ représentant respectivement les n utilisateurs et les p articles dans $\mathbb{R}^k$, le produit matriciel UI est la matrice de scores $(n \times p)$ associée à la fonction d'utilité f, où $(UI)_{xa} = f(x, a)$. Remarquons que le paramètre de taille k, commun aux matrices U et I, définit la dimension des espaces de représentation des utilisateurs et des articles. Dans la suite, nous verrons que k définit ainsi le nombre de variables latentes de notre modèle.

Fonction de Coût Avec les notations précédentes, une prédiction d'ordre est correcte si $f(x, a) > f(x, b)$ avec $(a, b) \in y_x$. Nous pouvons maintenant définir la fonction de coût D comme le nombre d'erreurs de prédiction sur les préférences par paires de la base d'apprentissage $(a, b) \in y_x$:

$$D(U, I) \; = \sum_{x} \sum_{(a,b) \in y_x} [[(UI)_{xa} \leq (UI)_{xb}]]$$

où $[[pr]]$ vaut 1 si pr est vraie, 0 sinon. Le but de l'apprentissage est de déterminer les matrices de paramètres U et I qui minimisent cette erreur. C'est un problème d'optimisation difficile car la fonction D n'est pas dérivable. Grâce à l'inégalité $[[x \leq 0]] \leq e^{-x}$, nous proposons de borner D de la façon suivante :

$$D(U,I) \leq \underbrace{\sum_{x} \sum_{(a,b)\in y_x} e^{(UI)_{xb}-(UI)_{xa}}}_{\mathcal{E}(U,I)}$$

La fonction $\mathcal{E}$ n'est pas convexe en U et I simultanément, en revanche elle est convexe en chacune des matrices séparément. Afin de pouvoir régulariser la factorisation, nous ajoutons également deux coefficients $\mu_U, \mu_I \geq 0$ pénalisant les normes de U et I. Finalement, nous obtenons la fonction $\mathcal{R}$:

$$\mathcal{R}(U,I) = \sum_{x} \sum_{(a,b)\in y_x} e^{(UI)_{xb}-(UI)_{xa}} + \mu_U \|U\|^2 + \mu_I \|I\|^2$$

Et le problème d'optimisation associé s'écrit simplement :

$$(U^*, I^*) = \underset{U,I}{argmin}\, \mathcal{R}(U,I)$$

3.3. *Modèle*

Optimisation Pour minimiser la fonction de coût $\mathcal{R}$, nous optons pour une approche itérative en 2 étapes, chaque étape consistant à fixer une des deux matrices et à minimiser $\mathcal{R}$ par rapport à l'autre grâce à la méthode du gradient conjugué. L'algorithme correspondant est donné en pseudo-langage.

Algorithm 1: Apprentissage d'ordres pour le FC

Entrée :

– L'ensemble de préférences par paires
$\forall x \in \mathcal{X}, y_x = \{(a,b) | a \in \mathcal{Y}, b \in \mathcal{Y}, r_x(a) > r_x(b)\}$

Initialiser:

– Initialiser $U^{(1)}$ et $I^{(1)}$ aléatoirement

– $t \leftarrow 1$

repeat

– $U^{(t+1)} \leftarrow \underset{U^{(t)}}{argmin}\, \mathcal{R}(U^{(t)}, I^{(t)})$

– $I^{(t+1)} \leftarrow \underset{I^{(t)}}{argmin}\, \mathcal{R}(U^{(t+1)}, I^{(t)})$

– $t \leftarrow t+1$

until *convergence de* $\mathcal{R}(U,I)$;

Sortie : U et I

Rappelons que $\mathcal{R}$ n'est pas convexe en U et I simultanément, et donc qu'une telle exploration de l'espace de recherche peut mener à des minima locaux de $\mathcal{R}$. Nous

donnons les formules permettant de calculer les gradients de $\mathcal{R}$ par rapport à U et à I, nécessaires à l'optimisation :

$$\frac{\partial \mathcal{R}}{\partial I_{jd}} = \sum_x \Big[\sum_a U_{xd}\, e^{(I^T u_x)_j - (I^T u_x)_a}\, \delta^x_{ja} - \sum_b U_{xd}\, e^{(I^T u_x)_b - (I^T u_x)_j}\, \delta^x_{bj} \Big] \; + \; 2\mu_I I_{jd}$$

$$\frac{\partial \mathcal{R}}{\partial U_x} = \sum_{(a,b)\in y_x} (i_b - i_a)\, e^{(I^T u_x)_b - (I^T u_x)_a} + 2\mu_U u_x$$

où u_x est la x-ième ligne de U, i_a est la a-ième colonne de I, $(I^T u_x)_j$ est la j-ième composante du vecteur $I^T u_x$, et δ^x_{ja} vaut 1 si $(j,a) \in y_x$, 0 sinon.

Implémentation et Complexité Algorithmique Face à des quantités toujours plus grandes de données à traiter, une caractéristique importante de tout algorithme de filtrage collaboratif est de pouvoir apprendre en un temps raisonnable, c'est à dire d'avoir une complexité algorithmique faible par rapport aux dimensions du problème : nombre de notes, nombre d'utilisateurs, nombre d'articles et nombre de facteurs cachés. Un inconvénient majeur des approches considérant les préférences par paires est que la complexité devient carrée par rapport au nombre d'articles. En effet, le calcul de la fonction de coût $\mathcal{R}$ nécessite de considérer p^2 préférences par paires d'articles où p est le nombre d'articles, ce qui est particulièrement coûteux lorsque p est grand comme c'est souvent le cas en filtrage collaboratif. Mais nous allons voir qu'il est possible de réduire cette complexité. En effet, on peut montrer que parcourir l'ensemble des préférences par paires est équivalent à parcourir l'ensemble des valeurs de notes possibles, et considérer pour chaque valeur deux ensembles disjoints d'articles, linéarisant ainsi le temps de calcul par rapport à p. Plus précisément, on peut montrer que la fonction $\mathcal{R}$ peut se ré-écrire de la façon suivante :

$$\mathcal{R}(U,I) = \sum_x \sum_{r\in\mathcal{V}} \left(\sum_{a|r_x(a)<r} e^{(UI)_{xa}} \times \sum_{b|r_x(b)=r} e^{-(UI)_{xb}} \right) + \mu_U \|U\|^2 + \mu_I \|I\|^2$$

Sous cette forme, la complexité du calcul de $\mathcal{R}$ n'est plus qu'en $O(npvk)$ (avec p articles, n utilisateurs, v le nombre de valeurs que peuvent prendre les notes et k le rang de la factorisation). On réduit la complexité des calculs de gradients de manière similaire, de sorte que la complexité totale de notre algorithme est en $O(Tnpvk)$, où T est le nombre maximal d'itérations.

Complexité de la Recommandation Pour générer les recommandations d'un utilisateur donné, le système de recommandation doit calculer son vecteur de scores en multipliant le vecteur utilisateur et la matrice des articles. La complexité du calcul est

en $O(pk)$, les articles recommandés étant ceux dont les scores sont les plus élevés. La complexité d'une recommandation est donc en $O(p(k+h))$, où h est le nombre d'articles recommandés à l'utilisateur.

4. EXPÉRIENCES

4.1. *Protocole Expérimental et Mesure d'Erreur*

Nous avons repris le protocole d'évaluation de la prédiction de notes de (Marlin, 2004a), afin de l'adapter pour l'évaluation de la prédiction d'ordres. Ainsi, nous avons (re)défini les notions suivantes :

La *généralisation faible* mesure la capacité d'une méthode à prédire correctement de nouvelles préférences par paires. Dans ces expériences, les notes fournies par chaque utilisateur sont divisées en un ensemble de notes observées, et un ensemble de notes cachées. Chaque ensemble de notes est ensuite utilisé pour générer un ensemble de préférences par paires. Le premier est utilisé pour l'apprentissage, tandis que le second est utilisé pour évaluer les performances en test. Dans nos expériences, nous avons aléatoirement choisi pour chaque utilisateur 2 notes cachées, obtenant ainsi une préférence par paires pour le test. Remarquons que ce protocole mesure la capacité de généralisation d'une méthode seulement sur les utilisateurs qui étaient présents au moment de l'apprentissage.

Dans le but de comparer les performances de différentes méthodes en terme de prédiction d'ordre, nous définissons l'erreur MRE (pour *mean ranking error*), qui compte le nombre moyen d'erreurs de prédictions sur les préférences par paires de test. Avec n utilisateurs et une préférence par paire de test par utilisateur :

$$MRE = \frac{1}{n} \sum_x [[(I^T u_x)_{a_x} \leq (I^T u_x)_{b_x}]]$$

où (a_x, b_x) est la préférence par paire de test pour l'utilisateur i. En mesurant ainsi la capacité d'un algorithme à retrouver la manière dont un utilisateur aurait ordonné un ensemble d'articles, l'erreur MRE appartient à la famille des mesures de précision d'ordre définie par (Herlocker *et al.,* 2004) dans laquelle figurent également le coefficient de corrélation de Spearman et le tau de de Kendall.

4.2. *Base de Test*

Nous avons utilisé la base de notes de films MovieLens. Elle contient $1,000,209$ notes concernant $6,040$ et $3,706$ films. Les notes sont des entiers de 1 à 5. La base contient 95.5% de notes manquantes. Pour chaque utilisateur nous avons aléatoirement choisi 2 notes de test, laissant ainsi $988,129$ notes pour l'apprentissage et $12,080$ pour le test. Ce processus a été répété 5 fois, générant un total de 10 bases pour l'éva-

luation de la généralisation faible. Tous les résultats présentés ci-dessous sont moyennés sur les bases ainsi générées.

4.3. *Résultats*

Nous avons comparé notre approche à la FMN généralisée et à la DVS pondérée pour plusieurs valeurs du rang k de la factorisation. Nous avons arrêté notre algorithme après 50 cycles des étapes (a) et (b) ; la FMN a été arrêtée après $1,000$ itérations. Nous avons simplifié le problème du choix des coefficients de régularisation en imposant $\mu_U = \mu_I$ pour la FMN ainsi que pour notre approche. Plusieurs coefficients ont été essayés, et les erreurs MRE présentées correspondent aux meilleures performances. La FMN a été utilisée avec $\mu_U = \mu_I = 1$, et notre approche d'ordonnancement avec $\mu_U = \mu_I = 100$. Les principaux résultats sont résumés dans le tableau suivant :

	FMN gén.	DVS pond.	Ordres
k	9	9	8
MRE	0.2658	0.2770	0.2737

Discussion Les valeurs optimales pour le rang k sont presque identiques pour les trois approches. Ce n'est pas surprenant concernant la FMN et la DVS car les deux méthodes sont très similaires (leur seule différence réside dans les contraintes de non-négativité de la FMN). En revanche c'est intéressant pour notre approche, et cela semble indiquer que la prédiction d'ordres soit aussi difficile que la prédiction de notes, puisque les deux tâches nécessitent à peu près le même nombre de facteurs cachés.

Bien qu'elles ne soient pas strictement équivalentes, l'erreur MRE définie pour l'évaluation est très proche de l'erreur de prédictions d'ordres optimisée par notre approche, alors que la FMN et la DVS optimisent une erreur carrée mesurant si elles sont capables de prédire les notes. C'est pourquoi ces premiers résultats sont un peu décevants, puisque nous nous attendions logiquement à ce que notre approche soit la meilleure en terme de prédiction d'ordres. Les bonnes performances de la FMN ne sont pas surprenantes puisque l'on savait qu'elle était déjà performante sur la même base en terme de prédiction de notes, au moins meilleure que la DVS (Pessiot *et al.,* 2006). Concernant la DVS, l'erreur MRE aurait pu être plus faible en augmentant le nombre d'itérations, mais sa complexité algorithmique élevée la rend difficilement utilisable sur de grosses bases, particulièrement quand le nombre d'articles est élevé comme c'est le cas pour MovieLens. Dans nos expériences, nous avons dû l'arrêter après 20 itérations à cause de son extrême lenteur. Un autre inconvénient de la DVS est l'absence de régularisation, ce qui la rend plus sensible au problème du surapprentissage.

Plusieurs directions doivent encore être explorées pour complérer et améliorer ces premiers résultats. La première concerne la normalisation par utilisateur ; lorsque l'on minimise une somme non normalisée d'erreurs (la somme des erreurs carrées pour chaque note avec la FMN et la DVS, la somme des erreurs de prédiction d'ordres pour chaque préférence par paires avec notre approche), les utilisateurs ayant noté beaucoup d'articles auront tendance à être associées à de grosses erreurs, concentrant ainsi l'apprentissage sur eux au détriment des autres utilisateurs. Ce problème peut être évité si l'on donne à chaque utilisateur la même importance en considérant des erreurs normalisées, c'est à dire en divisant l'erreur de chaque utilisateur par le nombre de ses notes ou de ses préférences par paires en apprentissage. D'ailleurs, la définition de notre erreur MRE est proche de celle d'une erreur normalisée, puisque nous ne considérons pour le test qu'une seule préférence par paire par utilisateur. C'est pourquoi nous espérons que l'optimisation d'une erreur normalisée donnera de meilleurs résultats.

Une deuxième direction que nous voulons explorer est une étude plus poussée des critères d'arrêt. Nous avons arrêté la FMN et notre approche après des nombres fixés d'itérations, qui semblaient correspondre à la convergence empirique en test. Dans nos expériences futures, nous essaierons plutôt de les arrêter lorsque les erreurs d'apprentissage seront stabilisées, ce qui nous permettra une comparaison plus approfondie des trois méthodes en terme de temps d'apprentissage.

Une autre question importante concerne la régularisation. C'est une caractéristique importante de tout algorithme d'apprentissage puisqu'elle permet d'éviter le phénomène du surapprentissage. Pour la FMN ainsi que pour notre approche, μ_U and μ_I sont les coefficients de la régularisation. Fixer $\mu_U = \mu_I = 0$ revient à ne pas régulariser du tout. Plus les coefficients sont élevés et plus on pénalise les normes des matrices. Dans nos expériences, nous avons fixé $\mu_U = \mu_I$ pour simplifier le problème du choix des coefficients. En faisait cela nous avons implicitement donné autant d'importance à chacun des paramètres de notre modèle. Dans nos travaux futurs, nous étudierons l'influence exacte de ces termes de régularisation, et comment ils devraient être choisis.

Résultats Détaillés Erreurs MRE pour plusieurs valeurs du rang k :

k	7	8	9	10	11
FMN	0.2696	0.2688	**0.2658**	0.2679	0.2684

k	6	7	8	9	10
DVS	0.2862	0.2803	0.2779	**0.2770**	0.2782

k	6	7	8	9	10
Ordres	0.2752	0.2744	**0.2737**	0.2743	0.2753

5. CONCLUSION ET PERSPECTIVES

Les approches basées sur la prédiction de notes sont encore l'objet de nombreux travaux en filtrage collaboratif. Les solutions proposées empruntent à des domaines variés de l'apprentissage statistique comme la classification supervisée, non supervisée, la régression, la réduction dimensionnelle ou encore l'estimation de densité. Ces méthodes ont toutes en commun de décomposer le processus de la recommandation en deux étapes : la prédiction de notes, puis la recommandation elle-même. Mais du point de vue de la recommandation, nous pensons que d'autres voies devraient être envisagées. Dans cet article, nous avons proposé une nouvelle approche basée sur la prédiction d'ordres pour le filtrage collaboratif : plutôt que de prédire des notes comme le plupart des méthodes proposées dans la littérature, nous prédisons des scores qui respectent au mieux les préférences par paires d'articles, car nous pensons qu'ordonner correctement les articles est plus important que prédire correctement leurs notes. Nous avons proposé un nouvel algorithme pour la prédiction d'ordre, nous avons défini un nouveau protocole d'évaluation et nous avons comparé notre approche à deux méthodes de prédiction de notes. Bien que les premiers résultats ne soient pas aussi bons que ce que nous attendions, nous pensons qu'ils peuvent être expliqués et améliorés en explorant plusieurs questions importantes que nous étudierons dans nos travaux futurs, comme la normalisation par utilisateur, les critères de convergence et la régularisation de l'apprentissage. Une autre direction de recherche intéressante consiste à explorer les liens entre notre travail et l'apprentissage multi-tâches (Ando *et al.,* 2005), dans le but d'étendre notre modèle au cadre de l'apprentissage semi-supervisé (Amini *et al.,* 2003).

REMERCIEMENTS

Ce travail a été partiellement financé par le programme IST de la Communauté Européenne, dans le cadre du réseau d'excellence PASCAL, IST-2002-506778.

6. Bibliographie

Aiolli F., Sperduti A., « Learning Preferences for Multiclass Problems », *in* , L. K. Saul, , Y. Weiss, , L. Bottou (eds), *Advances in Neural Information Processing Systems 17*, MIT Press, Cambridge, MA, p. 17-24, 2005.

Amini M.-R., Gallinari P., « Semi-Supervised Learning with Explicit Misclassification Modeling », *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03), Acapulco, Mexico, August 9-15, 2003*, p. 555-560, 2003.

Ando R. K., Zhang T., « A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data », *Journal of Machine Learning Research*, 2005.

Balabanovic M., Shoham Y., « Fab : content-based, collaborative recommendation », *Commun. ACM*, vol. 40, n° 3, p. 66-72, 1997.

Basu C., Hirsh H., Cohen W., « Recommendation as classification : using social and content-based information in recommendation », *AAAI '98/IAAI '98 : Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, American Association for Artificial Intelligence, Menlo Park, CA, USA, p. 714-720, 1998.

Blei D., Ng A., Jordan M., « Latent Dirichlet Allocation », *Journal of Machine Learning Research*, 2003.

Caillet M., Pessiot J.-F., Amini M.-R., Gallinari P., « Unsupervised Learning with Term Clustering for Thematic Segmentation of Texts », *Proceedings of the 7th Recherche d'Information Assistée par Ordinateur, Avignon, France*, CID, p. 648-656, 2004.

Caruana R., Baluja S., Mitchell T., « Using the Future to "Sort Out" the Present : Rankprop and Multitask Learning for Medical Risk Evaluation », *in* , D. S. Touretzky, , M. C. Mozer, , M. E. Hasselmo (eds), *Advances in Neural Information Processing Systems*, vol. 8, The MIT Press, p. 959-965, 1996.

Crammer K., Singer Y., « Pranking for Ranking », *Neural Information Processing Systems*, 2002.

DeCoste D., « Collaborative Prediction Using Ensembles of Maximum Margin Matrix Factorizations », *International Conference on Machine Learning*, 2006.

Deshpande M., Karypis G., « Item-Based Top-N Recommendation Algorithms », *ACM*, 2004.

Dhillon I. S., Sra S., « Generalized Nonnegative Matrix Approximations with Bregman Divergences », *NIPS*, 2006.

Gaussier E., Goute C., « Relation between PLSA and NMD and Applications », *SIGIR*, 2005.

Herlocker J. L., Konstan J. A., Terveen L. G., Riedl J. T., « Evaluating Collaborative Filtering Recommender Systems », *ACM Transactions on Information Systems*, 2004.

Hofmann T., « Probabilistic Latent Semantic Analysis », *Uncertainty in Artificial Intelligence*, 1999.

Hofmann T., « Latent semantic models for collaborative filtering », *ACM Trans. Inf. Syst.*, vol. 22, n° 1, p. 89-115, 2004.

Lee D. D., Seung H. S., « Learning the parts of objects by non-negative matrix factorization », *Nature*, 1999.

Marlin B., « Collaborative filtering : A machine learning perspective », 2004a.

Marlin B., « Modeling User Rating Profiles for Collaborative Filtering », *Neural Information Processing Systems*, 2004b.

Pessiot J.-F., Truong V., Usunier N., Amini M., Gallinari P., « Factorisation en Matrices Non-Negatives pour le Filtrage Collaboratif », *3eme Conference en Recherche d'Information et Applications (CORIA'06)*, Lyon, p. 315-326, March, 2006.

Rennie J. D. M., Srebro N., « Fast maximum margin matrix factorization for collaborative prediction », *ICML '05 : Proceedings of the 22nd international conference on Machine learning*, ACM Press, New York, NY, USA, p. 713-719, 2005a.

Rennie J. D. M., Srebro N., « Loss Functions for Preference Levels : Regression with Discrete Ordered Labels », *Proceedings of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling*, 2005b.

Schafer J. B., Konstan J., Riedi J., « Recommender systems in e-commerce », *EC '99 : Proceedings of the 1st ACM conference on Electronic commerce*, ACM Press, New York, NY, USA, p. 158-166, 1999.

Srebro N., Jaakkola T., « Weighted low rank approximation », *ICML '03. Proceedings of the 20th international conference on machine learning*, 2003.

Srebro N., Rennie J. D. M., Jaakkola T. S., « Maximum-Margin Matrix Factorization », *in* , L. K. Saul, , Y. Weiss, , l. Bottou (eds), *Advances in Neural Information Processing Systems 17*, MIT Press, Cambridge, MA, 2004.

Yu S., Yu K., Tresp V., Kriegel H.-P., « Collaborative ordinal regression », *ICML '06 : Proceedings of the 23rd international conference on Machine learning*, ACM Press, New York, NY, USA, p. 1089-1096, 2006.

Zhang S., Wang W., Ford J., Makedon F., « Learning from Incomplete Ratings Using Non-negative Matrix Factorization. », *SDM*, 2006.