Technique de factorisation multi-biais pour des recommandations dynamiques

Modou Gueye[*,**], Talel Abdesssalem[*], Hubert Naacke[***]

[*]Institut Telecom, Telecom ParisTech
46, rue Barrault 75013, Paris, France
prénom.nom@telecom-paristech.fr
http://www.telecom-paristech.fr

[**]LID, Université Cheikh Anta Diop
BP. 16432, Dakar-Fann, Sénégal
gmodou@ucad.sn,
http://www.ucad.sn

[***]LIP6, UPMC Sorbonne Universités
4, place Jussieu 75005, Paris, France
hubert.naacke@lip6.fr
http://www.lip6.fr

Résumé. La factorisation de matrices offre une grande qualité de prédiction pour les systèmes de recommandation. Mais sa nature statique empêche de tenir compte des nouvelles notes que les utilisateurs produisent en continu. Ainsi, la qualité des prédictions décroît entre deux factorisations lorsque de nombreuses notes ne sont pas prises en compte. La quantité de notes écartées est d'autant plus grande que la période entre deux factorisation est longue, ce qui accentue la baisse de qualité.

Nos travaux visent à améliorer la qualité des recommandations. Nous proposons une factorisation de matrices utilisant des groupes de produits et intégrant en ligne les nouvelles notes des utilisateurs. Nous attribuons à chaque utilisateur un biais pour chaque groupe de produits similaires que nous mettons à jour. Ainsi, nous améliorons significativement les prédictions entre deux factorisations. Nos expérimentations sur des jeux de données réels montrent l'efficacité de notre approche.

1 Introduction

L'objectif des systèmes de recommandation est de déterminer, parmi une grande quantité de produits, lesquels intéresseront le plus un utilisateur donné. Plus les produits recommandés intéressent effectivement l'utilisateur, plus la qualité de la recommandation est grande. La

recommandation a une valeur commerciale capitale pour tout type de e-commerce et concentre aujourd'hui beaucoup d'attention afin d'améliorer sa qualité (Schafer et al., 1999; Fleder et Hosanagar, 2007).

La factorisation de matrices est une technique de filtrage collaboratif apportant une qualité très satisfaisante (Su et Khoshgoftaar, 2009; Koren et al., 2009; Paterek, 2007; Takács et al., 2009; Koren, 2009). Elle consiste à construire des profils caractérisant les utilisateurs et les produits, au moyen de vecteurs de facteurs. Ces profils sont déduits des notes que les utilisateurs attribuent aux produits. Ainsi, il est possible d'estimer l'intérêt d'un utilisateur pour un produit en combinant le profil de l'utilisateur avec celui du produit (ex. : produit scalaire). Puis, les produits avec les estimations les plus grandes sont recommandés. Bien que très utilisée, la factorisation présente des limites. Un inconvénient majeur est que le modèle résultant de la factorisation, reste statique. Le modèle ne tient pas compte des nouvelles notes que les utilisateurs produisent continuellement. Ces nouvelles notes ne seront prises en compte qu'à une prochaine factorisation. Ainsi, le modèle a besoin d'être régénéré fréquemment, ce qui n'est pas toujours possible à cause du coût prohibitif de la factorisation. De ce fait, la qualité des recommandations décroît graduellement entre deux générations du modèle. Dans un contexte réel, les profils des utilisateurs évoluent dynamiquement. Prenons l'exemple d'un mélomane désirant acheter des chansons. Le marchand lui recommande une liste de chansons à partir du dernier modèle factorisé. L'utilisateur répond en notant les chansons qu'il connaît ou vient juste d'écouter. Cela renseigne le marchand en précisant les préférences actuelles de l'utilisateur. Le marchand souhaite tenir compte de ces toutes dernières notes de l'utilisateur, pour les prochaines recommandations, sinon les recommandations risquent d'être de faible qualité.

Nos travaux visent à améliorer la qualité des recommandations. Nous proposons une factorisation multi-biais supportant l'arrivée continuelle de nouvelles notes. Nous partons de l'observation que les utilisateurs ont tendance à noter différemment des produits de différents groupes. Nous attribuons à chaque utilisateur une liste de biais décrivant ces tendances. Ces biais sont continuellement mis à jour lorsque de nouvelles notes arrivent. Ceci permet de maintenir une qualité de recommandation satisfaisante plus longtemps et donc de différer la génération d'un nouveau modèle.

Notre approche améliore le passage à l'échelle en différent le besoin de recalculer le modèle. Les expérimentations que nous avons effectuées sur le jeu de données de Netflix et le plus grand de MovieLens confirment qu'elle est bien adaptée aux environnements dynamiques où de nouvelles notes arrivent continuellement (Wikipedia; GroupLens). Le coût d'intégration d'une nouvelle note est très faible et la qualité des recommandations ne décroît pas très rapidement entre deux factorisations successives. De plus, notre idée d'affiner les biais des utilisateurs est indépendante des modèles de factorisation. Elle peut être utilisée dans des modèles plus évolués réunissant, par exemple, l'aspect temporel et/ou avec des valeurs de confiance (Koren, 2010; Koren et al., 2009; Takács et al., 2008).

Ce papier est organisé comme suit. A la section 2, nous introduisons des notions préliminaires sur la recommandation dans notre contexte. Puis, à la section 3, nous présentons notre solution de factorisation basée sur des groupes. De même, nos résultats expérimentaux sont présentés à la section 4. Enfin, à la section 5, nous exposons des travaux connexes, avant de conclure.

2 Préliminaires

Le but d'un système de recommandation est de découvrir les intérêts d'un utilisateur afin de lui proposer des produits susceptibles de l'intéresser. La plupart du temps, l'intérêt d'un utilisateur sur un produit est donné sous forme de valeurs numériques d'une plage donnée (ex : 1 à 5) et il ne note un produit qu'une fois. Plus la valeur donnée est élevée, plus l'utilisateur s'intéresse au produit.

Si on considère un ensemble d'utilisateurs U, un ensemble de produits I et une liste de notes (u, i, r_{ui}, t_{ui}) où chaque valeur r_{ui} représente l'intérêt de l'utilisateur u pour le produit i - t_{ui} étant le moment où la note a été soumise, la recommandation consiste à prédire les futures notes des utilisateurs telles que l'écart entre une note prédite et celle réellement donnée ultérieurement, soit le plus petit possible. Cela permet de proposer à l'utilisateur les produits présentant les plus grandes valeurs de prédiction. Ainsi, la qualité d'un système de recommandation peut-être rattachée à la précision de ses prédictions. En pratique, pour estimer cette précision (c-à-d les écarts), l'ensemble des notes existantes est subdivisé en deux parties : la plus grande pour l'apprentissage et la seconde pour l'évaluation. La mesure appelée RMSE est l'une des plus utilisées pour l'évaluation. RMSE est la racine carré de la moyenne des carrés des écarts (Herlocker et al., 2004; Su et Khoshgoftaar, 2009; Wikipedia). Nous l'utilisons par la suite.

$$RMSE = \sqrt{\frac{1}{n} \sum_{u,i} (r_{ui} - \hat{r}_{ui})^2} \tag{1}$$

n représente le nombre total de notes à prédire. Plus petit est le RMSE, meilleures sont les prédictions.

2.1 factorisation de matrices

Les systèmes de recommandation utilisant la factorisation de matrices représentent les notes des utilisateurs dans une matrice R creuse. Les colonnes représentent les utilisateurs et les lignes les produits. Ainsi la note $r_{ui} \in R$ est celle donnée par l'utilisateur u au produit i. R est généralement très creuse (Wikipedia). L'objectif de la factorisation est de prédire les valeurs manquantes dans R. Dans sa forme basique (FM basique), elle cherche à approximer R comme le produit de deux autres matrices

$$R = P \cdot Q \tag{2}$$

Les deux matrices P et Q contiennent respectivement les vecteurs de facteurs des utilisateurs et ceux des produits. Ce sont les matrices de facteurs. Pour prédire la note $\hat{r}_{ui}$ que l'utilisateur u donnerait au produit i, il suffit simplement d'appliquer la formule

$$\hat{r}_{ui} = p_u \cdot q_i^T \tag{3}$$

p_u et q_i étant respectivement les vecteurs de facteurs de l'utilisateur u et du produit i dans P et Q.

Le processus d'apprentissage qu'effectue la factorisation détermine les valeurs dans P et Q telles qu'on s'approche le plus des notes r_{ui} existantes dans R. Il utilise une descente de

gradient stochastique (DGS) qui calcule un minimum local tel que la somme des erreurs (c-à-d des écarts), $e_{ui} \stackrel{def}{=} r_{ui} - \hat{r}_{ui}$ entre les notes prédites $\hat{r}_{ui}$ et celles réelles r_{ui} données par les utilisateurs, soit la plus faible possible. DGS minimise la somme des erreurs quadratiques $\sum_{ui} e_{ui}^2$ en ajustant les facteurs dans P et Q jusqu'à ce que cette somme ne diminue plus :

$$\begin{aligned} p_{uk} &\leftarrow p_{uk} + \lambda \cdot (2 \cdot e_{ui} \cdot q_{ki} - \beta \cdot p_{uk}) \\ q_{ki} &\leftarrow q_{ki} + \lambda \cdot (2 \cdot e_{ui} \cdot p_{uk} - \beta \cdot q_{ki}) \end{aligned} \tag{4}$$

Ceci permet de diminuer les erreurs et par conséquent d'avoir une meilleure approximation des notes réelles. Le paramètre λ introduit dans l'ajustement des facteurs représente un taux d'apprentissage. β est un paramètre de régularisation.

Après cette phase, les prédictions $\hat{r}_{ui}$ sont calculées comme les produits $p_u \cdot q_i^T$. Un tri est effectué par la suite pour trouver les produits les plus intéressants (ceux avec les plus grandes notes de prédiction) et les recommander à l'utilisateur concerné.

2.2 Factorisation biaisée de matrices (FBM)

Plusieurs améliorations possibles de la factorisation présentée ci-dessus ont été proposées dans la littérature.

L'une d'elle suppose que la plupart des variations observées sur les notes des utilisateurs sont dues principalement à des effets associés soit aux utilisateurs, soit aux produits (Takács et al., 2009; Koren et al., 2009; Paterek, 2007). Autrement dit, certains utilisateurs ont tendance à donner des notes plus élevées ou plus faibles que les autres utilisateurs. Et certains produits aussi sont plus ou moins appréciés que les autres. La factorisation basique (FM basique) présentée précédemment ne prend pas en compte ces tendances. La factorisation biaisée de matrices (FBM) introduit des biais pour tenir en compte ces variations de notation. Les biais reflètent les tendances des utilisateurs et des produits. On a la formule de prédiction suivante :

$$\hat{r}_{ui} = p_u \cdot q_i^T + \mu + b_u + b_i \tag{5}$$

où μ dénote la moyenne de toutes les notes confondues, b_u et b_i sont respectivement le biais de l'utilisateur et celui du produit (i.e, la tendance de l'utilisateur et la perception du produit par rapport à la moyenne). Une bonne approximation de ces biais est cruciale pour avoir des prédictions de bonne qualité (Paterek, 2007; Koren, 2009). Ainsi, ils doivent être ajustés durant la phase d'apprentissage en utilisant

$$\begin{aligned} b_i &\leftarrow b_i + \lambda \cdot (2 \cdot e_{ui} - \gamma \cdot b_i) \\ b_u &\leftarrow b_u + \lambda \cdot (2 \cdot e_{ui} - \gamma \cdot b_u) \end{aligned} \tag{6}$$

γ est un paramètre de régularisation. Il joue un rôle comparable à β dans l'équation 4.

3 Recommandation dynamique

Nos travaux considèrent les contextes dynamiques où de nouvelles notes sont continuellement soumises. Dans de tels contextes, il n'est pas possible d'avoir un modèle à jour à cause du

temps nécessaire pour le calculer. Au minimum, les notes soumises durant la génération d'un modèle ne sont pas prises en compte. Après la génération d'un modèle, la situation peut se dégrader assez rapidement puisque le nombre de notes non prises en compte augmente rapidement. De ce fait, une perte de qualité grandissante peut être observée dans les recommandations aussi longtemps qu'un nouveau modèle n'est pas généré. Pour y faire face, nous proposons un modèle combinant des biais globaux à des biais locaux. Les biais locaux sont calculés à partir des groupes de produits similaires. Leur coût de calcul faible permet de les ajuster à la volée lorsque de nouvelles notes arrivent. Ils assurent ainsi la robustesse du modèle en maintenant une meilleure qualité dans le temps.

3.1 Factorisation multi-biais de matrices (FMBM)

Nous nous basons sur l'observation que beaucoup d'utilisateurs tendent à sur-apprécier ou sous-apprécier les produits qu'ils notent. Une manière simple de quantifier cette tendance est d'assigner un biais global à chaque utilisateur comme avec la FBM (Paterek, 2007; Koren et al., 2009; Takács et al., 2008). Cependant, la tendance d'un utilisateur n'est généralement pas uniforme : elle peut changer d'un groupe de produits à un autre. Pour certains groupes de produits, un utilisateur peut avoir tendance à noter comme tout le monde alors qu'il surestime ou sous-estime d'autres groupes par manque d'objectivité. Cette tendance devient uniforme pour un ensemble de produits similaires ; ce que nous avons formalisé dans Gueye et al. (2012).

Pour prendre en compte cette diversité de notation, nous attribuons un biais local b_u^C à chaque utilisateur u pour chaque groupe C de produits jugés similaires ou proches. Cette multitude de biais par utilisateur permet d'avoir un modèle plus raffiné et visant une meilleure qualité de recommandation. Nous utilisons les techniques de clustering existantes pour grouper les produits et considérons ne pas disposer d'informations supplémentaires sur les produits. Nous construisons nos groupes en nous reposant uniquement sur les notes qui leur ont été attribuées.

Le biais b_u^C d'un utilisateur u pour un groupe de produits C est déduit de ses notes reçues par les produits de ce groupe. Pour chaque produit $j \in C$ notée par l'utilisateur u, nous définissons la déviation b_u^j de l'utilisateur pour ce produit comme la différence entre sa note r_{uj} et la moyenne des notes μ^C des produits du groupe C : $b_u^j = r_{uj} - \mu^C$. Le biais local b_u^C de l'utilisateur, au niveau du groupe C, est pris comme ma moyenne de ses déviations

$$b_u^C = \frac{1}{|C|} \sum_{j \in C} r_{uj} - \mu^C \quad \forall j \in C \tag{7}$$

Nous tenons compte aussi du nombre relatif de notes existant dans chaque groupe pour atténuer le décalage entre les biais locaux. Ainsi, nous définissons δ_u^C comme étant l'écart pondéré entre le biais local b_u^C de l'utilisateur u dans le groupe de produits C et son biais global b_u. Dans l'équation 8, n_u^C est le nombre de notes que l'utilisateur u a dans le groupe de produits C, et n_u le nombre total de notes qu'il a eues à soumettre.

$$\delta_u^C = \frac{n_u^C}{n_u} \cdot (b_u^C - b_u) \tag{8}$$

Notre formule de prédiction devient la suivante

$$\hat{r}_{ui} = p_u \cdot q_i^T + \mu^{c(i)} + \delta_u^{c(i)} + b_u + b_i \tag{9}$$

$c(i)$ représente le groupe auquel appartient le produit i et b_i le biais de ce produit (c-à-d. sa perception par les utilisateurs).

Finalement, nous affinons aussi les biais locaux $b_u^{c(i)}$ (à travers leurs écarts pondérés $\delta_u^{c(i)}$) durant la phase d'apprentissage en utilisant :

$$\delta_u^{c(i)} \leftarrow \delta_u^{c(i)} + \lambda \cdot (2 \cdot e_{ui} - \gamma \cdot \delta_u^{c(i)}) \tag{10}$$

L'algorithme 1 détaille les étapes de la génération de modèle par la FMBM. A la ligne 1, un clustering est effectué afin de déterminer les groupes de produits. La ligne 2 calcule les valeurs initiales des biais des produits, celles des biais globaux et locaux des utilisateurs. Les écarts pondérés $\left\{\delta_u^C\right\}$ entre les biais locaux et ceux globaux sont aussi calculés. La ligne 3 initialise les matrices de facteurs P et Q par des valeurs aléatoires faibles. Les lignes 4 à 11 constituent la partie principale de l'apprentissage. A chaque itération (lignes 5 à 10), l'erreur de prédiction e_{ui} est calculée pour chaque note. Puis les matrices de facteurs, les biais (globaux et locaux) sont ajustés en conséquence (lignes 7 à 11) en utilisant les équations 4, 6 et 10. La ligne 13 calcule la somme globale des erreurs quadratiques. L'apprentissage s'arrête lorsque cette somme ne décroît plus ou lorsqu'un nombre spécifié d'itérations a été atteint.

Algorithme 1: Algorithme de la FMBM.

Data : N_c : nombre de groupes, K : nombre de facteurs, $\mathbf{R}$, λ, β et γ
Result : P, Q, $\mu = \left\{\mu^C\right\}$, b_i, b_u et $\left\{\delta_u^C\right\},\ C \in C_1, C_2, ...C_{N_c}$

1 Créer les groupes $C_1, C_2, ...C_{N_c}$ à partir de $\mathbf{R}$;
2 Pour chaque produit i, utilisateur u, calculer b_i, b_u et $\left\{\delta_u^C\right\},\ C \in C_1, C_2, ...C_{N_c}$;
3 Initialiser les matrices P et Q;
4 **repeat**
5 **foreach** $r_{ui} \in \mathbf{R}$ **do**
6 Calculer e_{ui};
7 **for** $k \leftarrow 1$ **to** K **do**
8 Ajuster $p_{uk} \in P$, $q_{ki} \in Q$;
9 **end**
10 Ajuster b_i, b_u et $\delta_u^{c(i)}$;
11 **end**
12 Calculer la somme globale des erreurs $\sum_{r_{ui}>0} e_{ui}^2$;
13 **until** *condition d'arrêt*;
14 **return** P, Q, $\mu = \left\{\mu^C\right\}$, b_i, b_u, $\left\{\delta_u^C\right\},\ C \in C_1, C_2, ...C_{N_c}$

3.2 Intégration des nouvelles notes des utilisateurs

Après la génération d'un modèle, de nouvelles notes arrivent continuellement ; elles sont insérées dans la matrice R. Nous les intégrons dans le modèle en ajustant simplement les biais locaux des utilisateurs ayant émis ces nouvelles notes. L'ajustement des biais locaux a un impact important sur la recommandation lorsque seuls les K premiers produits (top-K) sont recommandés et que ces K premiers produits appartiennent à plusieurs groupes. Dans cette

situation, l'ajustement des biais locaux réordonne les groupes les uns par rapport aux autres. Des produits d'un groupe peuvent ainsi remplacer ceux d'un autre groupe dans la liste des K produits recommandés. Dans nos expérimentations, nous avons effectivement observé cette situation (pour trois groupes de produits définis, 58% des utilisateurs de Netflix ont au moins deux groupes de produits présents dans leurs top-5, et 55% pour MovieLens).

De plus, lorsqu'une nouvelle note r_{ui} arrive, l'ajustement du biais local b_u^C a un coût de calcul très faible car on considère seulement les notes que l'utilisateur u a déjà attribuées au sein du groupe $c(i)$. Le nombre moyen de notes à considérer est d'autant plus faible que le nombre de groupes est grand. Une descente de gradient est effectuée pour mettre à jour le biais local en utilisant l'équation 10. L'algorithme 2 présente les étapes de l'intégration. Comme dans l'algorithme 1, le processus d'ajustement s'arrête lorsque la somme globale des erreurs quadratiques ne décroit plus ou qu'un certain nombre d'itérations a été fait.

Algorithme 2: Algorithme d'intégration de nouvelles notes.

Data : P, Q, $V(u, c(i))$, b_i, b_u, $\delta_u^{c(i)}$, λ, β et γ

repeat
  foreach $r_{uj} \in V(u, c(i))$ **do**
    Calculer e_{uj} et ajuster $\delta_u^{c(i)}$;
  end
  Calculer la somme globale d'erreurs $\sum_{r_{uj}>0} e_{uj}^2$;
until *condition d'arrêt*;

4 Évaluation expérimentale

Nos expérimentations ont deux objectifs : (i) démontrer que la qualité initiale du modèle que nous générons est grande, (ii) montrer que l'intégration des nouvelles notes maintient le modèle à un bon niveau de qualité, sans factoriser à nouveau. Cela permet d'avoir un système réactif et peu coûteux en ressources puisque la génération des nouveaux modèles est retardée. Nous validons ces deux points séparément. Nous montrons d'abord qu'en termes de qualité notre solution est supérieure aux deux autres précédemment citées. Puis nous démontrons la perte de qualité que subit le système durant le temps lorsque les nouvelles notes des utilisateurs ne sont pas prises en compte. Et enfin, nous validons le gain obtenu par notre intégration.

Nous avons effectué nos expérimentations sur les jeux de données de Netflix et MovieLens dont les tailles respectives sont de 100M et 10M de notes (Wikipedia; GroupLens). Ces jeux de données sont très utilisés dans la littérature (Su et Khoshgoftaar, 2009).

Notre implémentation a été faite sous C++ avec une machine Linux (64 bits, Intel/Xeon x 8 threads, 2,66 Ghz, 16 GB RAM). Nous avons effectué un calibrage préliminaire pour fixer nos paramètres : $\lambda = 0,001$, $\beta = 0,02$, $\gamma = 0,05$, $N_c = 3$. Ces valeurs de λ, β, et γ sont proches de celles dans (Paterek, 2007). Nous avons limité le nombre d'itérations à 120 au plus et utilisons 40 facteurs pour P et Q. Pour le clustering des produits, nous avons effectué une FM basique de 30 itérations suivie d'un K-Means.

4.1 Qualité initiale

L'objectif de cette expérimentation est de comparer les qualités initiales des trois modèles. Nous trions les données par date croissante puis nous les séparons en deux parties : la première pour l'apprentissage (98% des notes, les plus anciennes) et le reste (les 2% de notes les plus récentes) pour le test. Ainsi, Notre jeu de test de Netflix contient 1,88M notes, une taille proche de celle du Netflix Prize dont le jeu de test contenait 1,4M de notes (Wikipedia). La table 1 rapporte les différentes erreurs RMSE obtenues avec les trois modèles FM basique, FBM et FMBM. FMBM dépasse les deux autres en termes de qualité. Il atteint 1,12% de qualité de plus que FBM. Notons à ce point qu'une amélioration aussi petite que 1% se traduit par une différence significative dans la plupart des "top-K" produits recommandés aux utilisateurs (Koren, 2007; Dror et al., 2011).

Jeu de données	FM basique	FBM	FMBM
Movielens	0,7743	0,7608	**0,7578**
Netflix	0,9599	0,9312	**0,9208**

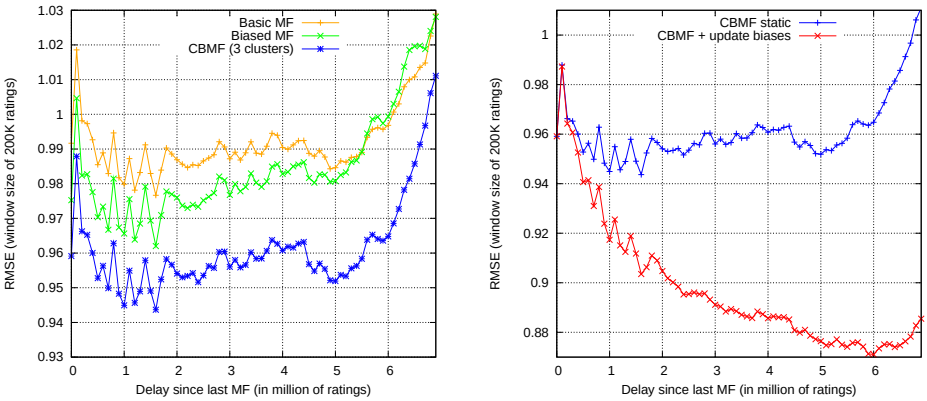
TAB. 1 – *Qualités initiales des trois modèles.*

4.2 Quantifier le besoin d'intégration en ligne

Le fait d'ignorer les nouvelles notes arrivées après la factorisation a-t-il un impact sur la qualité de la recommandation ? Nous répondons à cette question dans le contexte où les utilisateurs soumettent une quantité importante de notes (i.e. plusieurs millions de notes par jour). La compagnie Netflix, par exemple, reçoit jusqu'à 4M de notes par jour (Amatriain et Basilico, 2012). Pour mettre en place cette situation, nous devons utiliser un jeu de données de grande taille : seul celui de Netflix contient suffisamment de notes. Celui de MovieLens est assez petit.

Nous souhaitons observer l'impact global des nouvelles notes provenant de tous les utilisateurs sans exception. Pour cela, nous construisons le jeu de test de telle sorte que chaque utilisateur soit présent dans le test. Ainsi, le jeu de test contient les notes les plus récentes de chaque utilisateur à hauteur de 10%. Le reste (i.e., 90%) sert pour l'apprentissage. Plus précisément, nous alignons dans l'échantillon de test, la séquence d'arrivée des notes : à la date D_i, i notes sont déjà arrivées pour chaque utilisateur.

Nous mesurons ensuite l'évolution de la qualité des prédictions pour des dates D_i successives lorsqu'on progresse à travers le jeu de test. Nous utilisons une fenêtre glissante de 200.000 notes (dont la moitié est partagée avec la fenêtre précédente afin de lisser les résultats). La figure 1(a) montre l'évolution de la qualité pour les trois modèles : MF basique, FBM et FMBM. L'erreur de prédiction (i.e., le RMSE) augmente, cela confirme la perte de qualité au fil du temps. Nous observons une augmentation de 5% du RMSE lorsque 5M à 7M de nouvelles notes n'ont pas été prises en compte. En pratique, cela signifie que les trois modèles deviennent rapidement obsolètes.



(a) Croissance de l'erreur RMSE lorsque le nombre de nouvelles notes non prises en compte augmente.

(b) Évolution du RMSE avec l'intégration en ligne.

FIG. 1 – *Évolution de la qualité avec ou sans intégration.*

4.3 Robustesse dans le temps de notre modèle d'intégration en ligne

Le but ici est de montrer la robustesse de notre modèle dans le temps. Autrement dit qu'il maintient une bonne qualité dans le temps. Utilisant toujours les jeux d'apprentissage et de test de l'expérimentation précédente, nous prenons maintenant en compte l'intégration en ligne des nouvelles notes des utilisateurs en ajustant leurs biais locaux (cf. Algorithme 2).

Nous parcourons un à un les nouvelles notes dans le jeu de test. Chaque nouvelle note est comparée par rapport à la prédiction qui aurait été faite (on calcul l'écart), puis elle est automatiquement intégrée afin d'améliorer les futures prédictions. Le temps moyen d'intégration est de 1,24 millisecondes. L'intégration est rapide et ne constitue qu'un léger calcul.

La figure 1(b) présente la nouvelle évolution de la qualité des prédictions pour le modèle FMBM lorsqu'on intègre les nouvelles notes des utilisateurs. On y voit l'importance de la prise en compte de ces nouvelles notes avec un bénéfice de 13,97% lorsqu'on atteint 7M de nouvelles notes intégrées. C'est une amélioration significative pour des recommandations. Ce qui prouve que notre solution est robuste.

Nous avons effectué des expériences supplémentaires démontrant le bénéfice de mettre à jour les biais plutôt que les facteurs. Voir les résultats présentés dans Gueye et al. (2012).

5 Travaux connexes

Les contributions de ce papier sont (i) l'utilisation de plusieurs biais avec la factorisation (FMBM) et (ii) l'intégration en ligne des nouvelles notes des utilisateurs. Ce dernier point a fait l'objet de plusieurs travaux alors que pour le premier on en compte pas autant. Cependant, à notre connaissance, les précédents travaux qui utilisaient la factorisation comme technique de recommandation n'utilisaient pas l'ajustement des biais pour l'intégration des nouvelles notes.

Chakraborty (2009); Sarwar et al. (2002) se concentrent uniquement sur l'intégration des nouveaux utilisateurs avec leurs notes. Ils ne se focalisent pas sur les nouvelles notes des

utilisateurs déjà existants alors que dans (Rendle et Schmidt-Thieme, 2008), Rendle et al. les prennent en compte en mettant à jour uniquement les facteurs de l'utilisateur concerné ou ceux du produit. Ils utilisent une fonction d'estimation pour le choix de la modification des facteurs de l'utilisateur ou ceux du produit. Dans le même ordre idée, Cao et al. attaquent le problème autrement (Cao et al., 2007). Ils utilisent une factorisation non-négative de matrices (nonnegative matrix factorization, en anglais) et mettent à jour l'ensemble des facteurs de tous les utilisateurs. Ce qui donne un temps d'intégration plus élevé que le nôtre. En effet notre solution ne met à jour que le biais local de l'utilisateur concerné. D'où l'intérêt de l'utilisation de biais multiples.

En ce qui concerne l'utilisation de biais multiples, on ne compte pas beaucoup d'attention dans la littérature. Nous comptons juste quelques travaux proches du nôtre (Koren, 2010; Koenigstein et al., 2011). Koren (2010) utilise des sessions de temps pour suivre l'évolution des tendances des utilisateurs et des produits. Ainsi, il attribue des biais par session aux utilisateurs et aux produits. Ce qui conduit à de meilleures recommandations. Contrairement à eux, nous avons opté pour des biais par groupe de produit et non par session. Dans (Koenigstein et al., 2011), les auteurs sont assez proches de notre approche. En plus d'utiliser des biais par session de temps, ils prennent en compte les types de produits selon leurs attributs. Ce qui est dans le même ordre d'idée que notre clustering à l'exception près que nous n'avons pas besoin d'informations sur les produits pour les grouper. Nous nous basons uniquement sur les notes qu'ils ont reçues.

Nos expérimentations montrent le besoin de tenir en compte les nouvelles notes des utilisateurs le plutôt possible afin de maintenir la qualité des recommandations à un bon niveau. Évidemment on peut utiliser la distribution pour réduire le temps de la factorisation comme étudiée dans (Gemulla et al., 2011; Bickson, 2011). Ceci permettrait de recalculer le modèle plus fréquemment. Cependant, le besoin de solutions d'intégration en ligne demeure nécessaire pour les applications à large échelle avec des milliards de notes d'utilisateur et des millions de nouvelles notes chaque jour. On peut citer le cas de Netflix qui a plus de 5 milliards de notes d'utilisateur dans sa base et reçoit chaque jour 4 millions de nouvelles notes (Amatriain et Basilico, 2012). Pour ces types d'application, une balance entre le recalcul (avec un coût élevé) et l'intégration en ligne (sans une perte significative de qualité) est probablement la meilleure solution.

6 Conclusion

Nous nous sommes intéressés au problème de la recommandation lorsque de nouvelles notes sont sans cesse soumises. Nous avons proposé une solution de factorisation pour un tel contexte très dynamique.

En effet, la factorisation bien que faisant partie des meilleures techniques de recommandation, souffre du fait qu'elle ignore toute nouvelle note ajoutée par un utilisateur jusqu'à ce qu'une nouvelle factorisation ne soit calculée. Plus la période entre deux factorisations s'avère longue, plus la quantité d'informations non prise en compte est grande, ce qui dégrade la qualité des recommandations.

Notre solution introduit des biais personnalisés pour les utilisateurs afin de mieux capturer leur subjectivité selon les groupes de produits (les produits étant groupés à partir des notes qu'elles ont reçues). Elle met à jour ces biais lorsque de nouvelles notes arrivent.

Nos résultats expérimentaux montrent que notre solution surpasse celle utilisant un seul biais global par utilisateur et celle n'en ayant pas. De plus, dans un contexte dynamique où de nouvelles notes sont continuellement soumises, elle apporte une amélioration jusqu'à 13,97% sur sa qualité initiale en intégrant ces nouvelles notes. Le coût très faible de ces intégrations confirme sa viabilité pour le passage à l'échelle des systèmes de recommandation dans des contextes très dynamiques où les notes des utilisateurs arrivent sans cesse (Amatriain et Basilico, 2012).

Références

Amatriain, X. et J. Basilico (2012). Netflix recommendations : Beyond the 5 stars. The Netflix Tech Blog.

Bickson, D. (2011). Large scale matrix factorization - yahoo ! kdd cup. Large Scale Machine Learning and Other Animals.

Cao, B., D. Shen, J.-T. Sun, X. Wang, Q. Yang, et Z. Chen (2007). Detect and track latent factors with online nonnegative matrix factorization. In *Proceedings of the 20th international joint conference on Artifical intelligence*, San Francisco, CA, USA, pp. 2689–2694. Morgan Kaufmann Publishers Inc.

Chakraborty, P. (2009). A scalable collaborative filtering based recommender system using incremental clustering. In *Advance Computing Conference, IACC 2009. IEEE International*, pp. 1526–1529.

Dror, G., N. Koenigstein, Y. Koren, et M. Weimer (2011). The yahoo ! music dataset and kdd-cup'11. In *Proceedings of KDDCup 2011*.

Fleder, D. M. et K. Hosanagar (2007). Recommender systems and their impact on sales diversity. In *Proceedings of the 8th ACM conference on Electronic commerce*, EC '07, New York, NY, USA, pp. 192–199. ACM.

Gemulla, R., E. Nijkamp, P. J. Haas, et Y. Sismanis (2011). Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, New York, NY, USA, pp. 69–77. ACM.

GroupLens. http ://www.grouplens.org/node/73.

Gueye, M., T. Abdessalem, et H. Naacke (2012). Dynamic recommender system : using cluster-based biases to improve the accuracy of the predictions. Technical report, arXiv.org.

Herlocker, J. L., J. A. Konstan, L. G. Terveen, et J. T. Riedl (2004). Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst. 22*, 5–53.

Koenigstein, N., G. Dror, et Y. Koren (2011). Yahoo ! music recommendations : modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys '11, New York, NY, USA, pp. 165–172. ACM.

Koren, Y. (2007). How useful is a lower rmse ? Netflix Prize Forum.

Koren, Y. (2009). The bellkor solution to the netflix grand prize.

Koren, Y. (2010). Collaborative filtering with temporal dynamics. *Commun. ACM 53*(4), 89–97.

Koren, Y., R. Bell, et C. Volinsky (2009). Matrix factorization techniques for recommender systems. *Computer 42*, 30–37.

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In *Proc. KDD Cup Workshop at SIGKDD'07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pp. 39–42.

Rendle, S. et L. Schmidt-Thieme (2008). Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In P. Pu, D. G. Bridge, B. Mobasher, et F. Ricci (Eds.), *RecSys*, pp. 251–258. ACM.

Sarwar, B., G. Karypis, J. Konstan, et J. Riedl (2002). Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Proceedings of the 5th International Conference in Computers and Information Technology*.

Schafer, J. B., J. Konstan, et J. Riedi (1999). Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, EC '99, New York, NY, USA, pp. 158–166. ACM.

Su, X. et T. M. Khoshgoftaar (2009). A survey of collaborative filtering techniques. *Adv. in Artif. Intell. 2009*, 4 :2–4 :2.

Takács, G., I. Pilászy, B. Németh, et D. Tikk (2008). Investigation of various matrix factorization methods for large recommender systems. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, NETFLIX '08, New York, NY, USA, pp. 6 :1–6 :8. ACM.

Takács, G., I. Pilászy, B. Németh, et D. Tikk (2009). Scalable collaborative filtering approaches for large recommender systems. *J. Mach. Learn. Res. 10*, 623–656.

Wikipedia. http ://en.wikipedia.org/wiki/netflix_prize.

Summary

It is today accepted that matrix factorization models allow a high quality of rating prediction in recommender systems. However, a major drawback of matrix factorization is its static nature that results in a progressive declining of the accuracy of the predictions after each factorization. This is due to the fact that the new obtained ratings are not taken into account until a new factorization is computed, which can not be done very often because of the high cost of matrix factorization.

In this paper, aiming at improving the accuracy of recommender systems, we propose a cluster-based matrix factorization technique that enables online integration of new ratings. Thus, we significantly enhance the obtained predictions between two matrix factorizations. We use finer-grained user biases by clustering similar items into groups, and allocating in these groups a bias to each user. The experiments we did on large datasets demonstrated the efficiency of our approach.