Apprentissage et Factorisation pour la Recommandation

Julien Delporte*, Stéphane Canu*, Alexandros Karatzoglou**

*INSA de Rouen, LITIS
Avenue de l'Université, 76800 St Étienne du Rouvray
prenom.nom@insa-rouen.fr,
**Telefónica Research
Barcelone, Espagne
alexk@tid.es,

Résumé. De nombreux sites de l'internet proposent aujourd'hui des conseils personnalisés élaborés par un système de recommandation, un ensemble de logiciels et de méthodes permettant de suggérer automatiquement des articles pouvant être utiles à un utilisateur. L'arrivée des réseaux sociaux en ligne a ajouté une nouvelle dimension à la recommandation, dans laquelle la structure du graphe social peut être utilisée comme source d'informations. Cet article constitue un état de l'art de méthodes de recommandation orientées modèle, que sont les méthodes de factorisation dont l'intérêt a été démontré lors du challenge Netflix. Il présente également un travail montrant une approche qui prend en compte le réseau social et qui réalise de la prédiction sur des données d'appréciation implicite (*implicit feedback*). Nous avons testé ce modèle sur des données réelles volumineuses et observé l'amélioration des performances obtenues par d'autres méthodes de l'état de l'art.

1 Introduction

Qui n'a pas rêvé, pour l'aider à choisir un appareil photo ou les bons outils de bricolage, de pouvoir compter sur un ami compétent, disposé à lui prodiguer LE bon conseil. C'est cette forme d'expertise, adaptée à l'utilisateur, que tentent de reproduire de nombreux sites de l'internet comme Amazon pour les livres ou Facebook pour les personnes susceptibles de devenir nos « amis ». Un système de recommandation est un outil qui permet de proposer à l'utilisateur d'un service, un certain nombres d'articles [1] qui seraient les plus susceptibles de l'intéresser. La recommandation est la synthèse automatique d'un ensemble d'évaluations complexes éventuellement implicites, dépendantes du contexte et basées sur des activités passées.

Conçus initialement pour doper les ventes, les systèmes de recommandation sont aujourd'hui un domaine en pleine expansion au niveau du marché et des recherches associées. En effet, si certains systèmes de recommandation donnent des conseils utiles [2], d'autres affichent

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

2. Amazon rapporte en 2006 que de l'ordre de 35% de ses ventes seraient liées à des recommandations.

des performances modestes, voire ont un effet contre productif du fait de leurs mauvaises préconisations. Il existe donc une marge de progression et de nombreuses voies d'amélioration. Parmi les facteurs principaux liés à la qualité d'un système de recommandation, nous nous intéressons dans ce travail à celui de la pertinence des algorithmes en supposant disposer de suffisamment de données de qualité (ce qui n'est pas toujours le cas dans la réalité) et de pré et post traitement adaptés. Le but des algorithmes de recommandation est de convertir les données disponibles en un conseil personnalisé utile. La personnalisation étant la capacité de s'ajuster automatiquement et de présenter à chaque utilisateur des informations utiles en rapport avec ses intérêts et ses besoins (la bonne information pour la bonne personne au bon moment).

Une manière de traiter le problème consiste à extraire automatiquement, à partir des masses de données hétérogènes disponibles, des variables latentes (non directement observables) pertinentes en fonction de la question posée et des contraintes associées. Ces variables latentes correspondent à de nouveaux facteurs issus de l'analyse conjointe des informations disponibles sur les produits et les utilisateurs et de la réduction de leur dimension. Mais ce type d'approche est très gourmand en ressource informatique (d'où son introduction relativement récente) ce qui explique que, parmi les méthodes d'analyse des données existantes, cet article s'intéresse aux « méthodes de factorisation » les plus susceptibles de « passer à l'échelle ».

Ce terme « méthode de factorisation » de matrice (ou de blocks de données) pour l'analyse des données multivariées désigne un ensemble de problèmes et d'algorithmes liés à la détermination automatique de nouvelles variables (facteurs) sous-jacentes à des groupes de variables partiellement observées pour un objectif donné (et donc à une réduction de la dimensionalité). La méthode la plus utilisée est sans doute l'analyse en composante principale (ACP) introduite par Pearson en 1901.

Soulignons que le problème de construction de variables latentes par factorisation est un problème générique qui intéresse de nombreux domaines d'applications et pas seulement la recommandation. Cette question suscite en particulier un réel intérêt de la communauté de l'apprentissage statistique comme en témoignent les récents workshop proposés lors des conférences de référence comme NIPS et ICML ou les tutoriels proposés à KDD ou WWW.

Cet article est organisé de la manière suivante. De nombreux problèmes et difficultés inhérents aux systèmes de recommandation peuvent se poser, nous en ferons un rapide survol dans la section 2. Nous présenterons ensuite, dans la section 3, un état de l'art de certaines méthodes permettant la résolution de problèmes de recommandation, et plus particulièrement les méthodes basées sur le principe de factorisation. Nous proposerons enfin, dans la section 4, une méthode pour résoudre un problème d'ordonnancement (de *ranking*) d'articles sur des données d'appréciation implicite (*implicit feedback*) à travers un réseau social.

2 Les systèmes de 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 Tapstry et la création du laboratoire de recherche GroupLens qui travaille explicitement sur le problème de la recommandation automatique dans le cadre des forums de news de Usenet. 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. Puis en 1995 apparait Ringo, un

système de recommandation de musique basé sur les appréciations des utilisateurs et Bellcore un système de recommandation de vidéos. Ringo a été commercialisé sous le nom de Firefly et GroupLens a créé la société Net Perceptions dont un des premier client a été Amazon. C'est en 1996 qu'eut lieu le premier workshop dédié aux systèmes de recommandation qui est ensuite devenue la conférence RecSys [3], référence dans le domaine.

Aujourd'hui, il existe de nombreux logiciels de recommandation disponibles dont certains sous licence libre comme Myrix ou Duine, ou d'autres sous license propriétaire comme les solutions proposées par Kxen [4]. 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 » [5] qui était démonstrateur au sein du projet CADI 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 alors l'utilisateur à résister aux suggestions des systèmes de recommandation et même à les ignorer.

2.1 Les problèmes liés à la recommandation

Le problème de recommandation consiste, pour un utilisateur donné, à prédire son score d'appétence (ou d'utilité) pour une liste d'articles, ce qui permet 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 large) où les données de base sont des couples articles/utilisateur, et la recommandation d'amis à partir d'un graphe (social) pratiquée principalement par les réseaux sociaux (Aggarwal, 2011). Dans le premier cas, l'archétype du problème de recommandation peut se formaliser comme un problème de complétion de matrice illustré par le tableau 1. 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 vu. 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, ou de prédire un comportement de groupe. On appelle ces variables cachées des variables latentes.

3. recsys.acm.org
4. myrrix.com, sourceforge.net/projects/duine/, www.kxen.com
5. www.b-a-o.fr

Spectateur \ Films											
	?	**1**	**4**	?	**1**	?	**1**	**1**	**5**	?	**5**
	2	?	?	?	**2**	**4**	**3**	?	?	**2**	**4**
	1	**5**	?	**5**	?	**2**	**3**	?	**5**	?	?
	1	**2**	**3**	**5**	**5**	?	?	**2**	**2**	**2**	**5**
	1	**4**	**3**	**2**	?	**5**	**3**	**2**	**4**	**2**	**3**
	?	?	**1**	?	**3**	**3**	?	?	**1**	?	?
	?	**5**	**3**	**4**	?	**4**	**4**	?	**1**	**3**	**4**

TAB. 1 – *Exemple d'une matrice spectateur/film de vote (*rating*). Les ? sont des valeurs manquantes à compléter à partir des données disponibles.*

Les systèmes de recommandation ne visent pas seulement à augmenter les ventes mais aussi à les diversifier pour augmenter la satisfaction et la fidélité des utilisateurs. Certains problèmes sont plus spécifiques et plus ciblés. C'est le cas notamment des problèmes de type longue traîne (*long tail*) qui se concentrent sur l'amélioration de prédictions liées aux articles n'ayant été que très peu vus/achetés par les utilisateurs. Quand moins d'informations sont disponibles, la recommandation est plus difficile. C'est aussi le cas des problèmes de type démarrage à froid (*cold start*) où l'on cherche à effectuer les meilleures prédictions possibles pour les nouveaux utilisateurs ou les utilisateurs sur lesquels nous n'avons que peu d'informations.

2.2 Le Challenge Netflix

La société Netflix a organisé en 2007 une compétition de recommandation mettant en jeu un million de dollars [6]. 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'affaire, ils voulurent améliorer leur moteur de recommandation. 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 dont la matrice contenant l'ensemble des votes des clients et un certain nombre de d'informations contextuelles sur les utilisateurs et sur les films. 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 RMSE 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 simple à mettre en œuvre. Malheureusement cet agrégat de modèles

6. www.netflixprize.com

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.

2.3 Les types de données : des volumes massifs

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 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. 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). Nous ne disposons que de peu d'informations, il peut donc être intéressant de se servir des données contextuelles disponibles, telles que 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…), utiliser ces données permettrait de prédire des changements de comportement d'utilisateur en améliorant ainsi la prédiction. L'utilisation du graphe social peut être un avantage, en se basant sur le principe d'affinité (*homophily*, Lazarsfeld et al., 1954), qui permet de prédire les préférences d'utilisateurs en observant les préférences de ses amis. Le regroupement de l'ensemble de ces différents types de données est appelé *Data Block* et est illustré par la figure 1. On y retrouve les trois caractéristiques des volumes massifs de données : leur volumétrie, leur variété (structurée, non structurée, multifacettes…) et leur vélocité puisque dans de nombreux systèmes de recommandation les données doivent être utilisées au fil de l'eau dans des délais fortement contraignants. En plus de la nature particulière des données dans les systèmes de recommandation, d'autres difficultés se posent. C'est le cas de l'évaluation. Il est en effet difficile d'évaluer la pertinence d'un système de recommandation. Pu et al. (2012); Ricci et al. (2011) se sont intéressés à ce problème.

2.4 Les différentes approches

Le problème de recommandation peut se formaliser comme l'estimation de la fonction d'utilité (score d'appétence) d'un article pour un utilisateur. Autour de ce problème s'est développée une certaine terminologie visant à distinguer différents types d'approches (voir Ricci

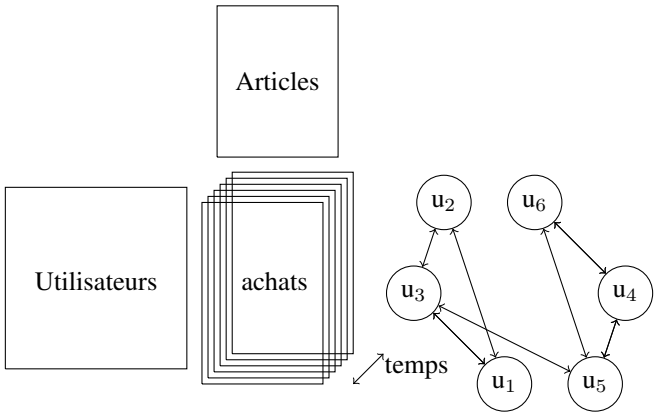

FIG. 1 – *Illustration d'un ensemble de données de type Data Block. Les matrices d'achat forment un tenseur dans le temps. Des données contextuelles sont disponibles sur les utilisateurs et sur les articles et forment des matrices supplémentaires. D'autres informations sont disponibles sous forme de graphe (réseaux sociaux).*

et al., 2011, pour des détails). On peut distinguer trois grandes classes d'approches selon la nature des données prises en compte.

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. Cette classe regroupe des approches communes avec la recherche d'information (IR) comme la classification (*Clustering*), les arbres de décision ou encore les réseaux de neurones.

La deuxième classe regroupe les approches 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 absolument 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 utilisateur ou articles. On compte également parmi ces approches les méthodes dites orientées modèle (*model-based*) comme les méthodes de factorisation.

La troisième est en fait un mélange des deux types d'approches précédemment définis. 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. Parmi ces informations, on peut citer les données contextuelles propres aux articles, les données contextuelles propres aux utilisateurs comme dans les approches orientées démographie, ou encore les données d'un réseau social comme dans les approches orientées communauté. C'est-à-dire toutes ou une partie des données représentées dans la figure 1.

2.5 L'éthique et les systèmes de recommandation

Il nous semble important d'évoquer les problèmes éthiques soulevés dans le monde de la recommandation. En effet, la personnalisation des recommandations est d'autant plus précise qu'elle utilise des données pertinentes sur l'utilisateur, ses gouts 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). Netflix a notamment 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 et 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 tel produit, ont également acheté ce produit » rassurantes pour l'utilisateur.

3 Factorisation et apprentissage

3.1 Factorisation et optimisation

On suppose que l'on dispose d'une matrice de données $Y \in \mathbb{R}^{n\times p}$ concernant n utilisateurs et p articles (*cf* figure 1). 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 $M \in \mathbb{R}^{d\times p}$ de sorte que la reconstruction $F_{ij} = U_i M_j$ entre un utilisateur i et un article j estime le terme Y_{ij} ou son utilité (score). U_i désigne la i-ème ligne de la matrice U et M_j désigne la j-ème colonne de la matrice M. Ainsi F_{ij} peut être utilisé pour fournir une recommandation en sélectionnant les N premiers articles obtenant le score le plus important pour un utilisateur donné, comme c'est le cas dans les problèmes d'ordonnancement (problème de *ranking*). Les facteurs latents U et M s'obtiennent généralement en minimisant une fonction objectif qui provient de fonctions de coût régularisées (Takacs et al., 2009) ou de modèles probabilistes (Krohn-Grimberghe et al., 2012). Dans les deux cas il s'agit d'un objectif multicritère puisque nous avons deux critères contradictoires à concilier, à savoir obtenir F proche de Y et garantir la faculté de généralisation du modèle sous-jacent en contrôlant la complexité de F. Le problème peut être formalisé par la minimisation du risque pénalisé :

$$\min_{F\in\mathbb{R}^{n\times p}} \mathcal{L}(F,Y) + \lambda\Omega(F) \tag{1}$$

où $\mathcal{L}(F,Y)$ est un terme d'attache aux données (comme la norme de Frobenius de l'erreur $\|F-Y\|_F^2$), $\Omega(F)$ un terme régularisant (une pénalité) permettant d'éviter le sur-apprentissage (comme le rang de la matrice F) et λ un paramètre d'équilibre dont le réglage établi le niveau de consensus entre les deux critères. Cette formulation de type lagrangienne est également appelée régularisation de Tikhonov.

3.1.1 Problèmes et modèles

Problème d'approximation : le problème d'approximation consiste à chercher une matrice F 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 1 en utilisant une fonction $\mathcal{L}$ de la forme :

$$\mathcal{L}(F, Y) = \sum_{i=1}^{n} \sum_{j=1}^{p} l(F_{ij}, Y_{ij}) \tag{2}$$

où $l : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ est un coût quelconque (distance, divergence...). Nous discutons du choix de cette fonction l dans le paragraphe 3.1.2. Le modèle implicite associé peut s'écrire

$$Y = F + R$$

où R est la matrice des résidus que l'on cherche à minimiser. Dans un cadre probabiliste bayésien, la fonction $\mathcal{L}$ peut être vue comme l'opposée du log de la probabilité a posteriori.

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). 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 utiliser une matrice de poids W qui jouera le rôle de filtre sur les données :

$$\mathcal{L}(F, Y) = \sum_{i=1}^{n} \sum_{j=1}^{p} W_{ij}\ l(F_{ij}, Y_{ij}). \tag{3}$$

On pourra par exemple choisir la matrice W de la forme suivante :

$$W_{ij} = \begin{cases} 0 & \text{si } (i,j) \notin \mathcal{Y} \\ \phi(Y_{ij}) & \text{si } (i,j) \in \mathcal{Y} \end{cases} \tag{4}$$

où $\phi : \mathbb{R} \longmapsto \mathbb{R}^+$ désigne une fonction positive quelconque. Ainsi ne seront prises en compte que les valeurs connues de la matrice Y qui seront pondérées. Ce modèle 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).

Problème de décomposition : Il existe également un type de problème que l'on peut appeler problème de décomposition, qui correspond plus à une façon de voir les choses qu'à un problème. Ce type de problème est décrit notamment par Candès et al. (2011) sous le terme d'ACP robuste. Dans ce cadre, la matrice de données Y est modélisée par :

$$Y = L + S + R$$

avec $F = L + S$ 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 critère propre).

3.1.2 Le critère d'attache aux données

Le terme d'attache aux données traduit la confiance que l'on accorde à la prédiction F ayant observé Y. Dans un cadre probabiliste, c'est l'opposé de la log vraisemblance des données. On peut citer parmi les plus utilisés le coût quadratique $l(y, f) = \frac{1}{2}(y-f)^2$. Si l'on choisit un coût logistique $l(y, f) = 1-e^{-yf}$ ou encore coût charnière $l(y, f) = \max(0, 1-yf)$, 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 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).

Il est également possible utiliser des divergences et en particulier les divergence 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 contrainte de positivité (Lee et Seung, 2001; Févotte et al., 2009).

3.1.3 Le terme de régularisation

Dans le cas de la factorisation le terme de régularisation Ω est généralement de la forme $\Omega(F) = \Omega(U, M) = \Omega_1(U) + \mu\Omega_2(M)$, où μ est un paramètre d'équilibrage et les Ω_i sont généralement des normes. Deux classes de normes sont classiquement 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 les tous éléments de la matrice.

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

$$\|F\|_{\rho\to q} = \sup_{u\in\mathbb{R}^p} \frac{\|Fu\|_q}{\|u\|_\rho} \tag{5}$$

où $(\rho, q) \in \{1, 2, \ldots, \infty\}^2$ dans la plupart des cas. Lorsque $\rho = q = 1$ il s'agit de la norme L_1 et $\|F\|_{1\to 1} = \max_j \sum_i |F_{ij}|$, lorsque $\rho = q = \infty$ il s'agit de la norme L_∞ et $\|F\|_{\infty\to\infty} = \max_i \sum_j |F_{ij}|$, et enfin lorsque $\rho = q = 2$ il s'agit de la norme euclidienne et $\|F\|_{2\to 2} = \max_i |\lambda_i|$ (λ_i désigne les valeurs singulières de F). On retrouve l'utilisation de normes de ce type dans Srebro et Shraibman (2005), où est défini la norme max de la façon suivante :

$$\|F\|_{\max} = \min_{U,M} \|U\|_{2\to\infty}\|M\|_{2\to\infty}. \tag{6}$$

Parmi l'autre classe de norme matricielle, la plus répandue est la norme de Frobenius : $\Omega(F) = \|F\|_F^2 = \sum_i \sum_j F_{ij}^2$ Un autre terme régularisant très utilisé est $\Omega(F) = rang(F)$ mais on pourra également trouver une relaxation que l'on appelle la norme nucléaire : $\Omega(F) = \|F\|_* = \sum \lambda_i$, où λ_i désigne la i-ème valeur singulière de F (Recht et al., 2010). D'autres normes permettront 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 (Lee et Seung, 2001). Cependant cette norme est non convexe. On utilisera alors des relaxations du problème, en utilisant entre autres, celle que nous désignerons comme étant la norme $\|.\|_1$ (et qui ne désignera pas la norme d'opérateur L_1

noté $\|.\|_{1\to 1}$) 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, l'approximation linéaire par morceaux ou l'approximation logarithmique. 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' :

$$\|F\|_{q,q'} = \sum_i \|F_{i\bullet}\|_q^{1/q'}. \tag{7}$$

3.1.4 La nature des problèmes

Parmi les problèmes d'optimisation posés par l'équation 1, certains sont convexes et d'autres non de même que certains sont différentiables alors que d'autres ne le sont pas. 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 obtenir une solution parcimonieuse), il n'est pas possible de calculer le gradient. Ainsi, pour un problème convexe, un point est solution si zéro appartient à la sous-différentielle du lagrangien en ce point. Pour un problème non-convexe il faudra trouver les solutions pour lesquelles zéro appartient à la sous-différentielle au sens de Clarke (Sra, 2011).

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 et Gordon (2008) (table 1 page 364), dans Koren et al. (2009), dans Rendle (2010) ou encore dans le tutoriel de KDD 2012 [7].

3.2 Quelques algorithmes pour la factorisation

3.2.1 Méthode de résolution « exacte » sur matrices creuses

Dans certains cas il est possible d'utiliser des algorithmes permettant un calcul direct de la solution du problème (1). C'est le cas notamment lorsque $\mathcal{L}(Y,F) = \|Y-F\|_F^2$ et $\Omega(F) = \text{rang}(F)$. La solution est alors donnée par la décomposition en valeurs singulières de Y. Il existe des méthodes efficaces permettent de calculer directement tout ou partie de cette décomposition. C'est par exemple le cas du programme PROPACK [8] qui permet d'obtenir cette décomposition en utilisant un processus de Lanczos. PROPACK gère, entre autres, les instabilités numériques liées à cet algorithme et est particulièrement optimisé pour la décomposition de matrices creuses de très grande taille.

3.2.2 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 (éventuellement stochastique), car elles tirent très bien avantage de l'aspect parcimonieux de la matrice.

7. http ://www.ismll.uni-hildesheim.de/aktuelles/kdd_en.html
8. http ://soi.stanford.edu/ rmunk/PROPACK/

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

- $U_i^{NEW} = U_i^{OLD} - p(\nabla_{U_i}\mathcal{L}(Y,U,M) + \nabla_{U_i}\Omega(U,M))\ \forall i$
- $M_j^{NEW} = M_j^{OLD} - p(\nabla_{M_j}\mathcal{L}(Y,U,M) + \nabla_{M_j}\Omega(U,M)\ \forall j$

où p 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. L'initialisation des valeurs de U et M se fait de façon aléatoire selon une loi de probabilité quelconque. Dans les types de problèmes abordés dans cet article, la taille des données interdit souvent la recherche d'un pas et celui-ci est donc fixé après une validation croisée. Beaucoup d'algorithmes de factorisation sont basés sur ces règles de mise à jour comme la *Maximum Margin Matrix Factorisation* (MMMF) (Srebro et al., 2005; Weimer et al., 2008).

Mais la formulation la plus utilisée désormais est celle proposée par Funk (2006) [9]. 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.

L'algorithme RSMD met en œuvre cette règle de mise à jour [10] et c'est également cette règle qui est utilisée dans le logiciel CoFiRank [11].

3.2.3 Optimisation alternée

La méthode dite d'optimisation alternée peut être utilisé dans le même cadre que la descente de gradient stochastique. L'idée de cette méthode de résolution de factorisation est d'obtenir les règles de mise à jour des facteurs en calculant le gradient pour ce facteur, en l'annulant et en isolant le terme à mettre à jour pour connaître sa règle de mise à jour.

Prenons l'exemple du coût suivant : $\min_{U,M} J = \frac{1}{2}\sum_{ij}(U_iM_j - Y_{ij})^2 + \lambda(\|U\| + \|M\|)$. La dérivée partielle par rapport à U_i se calcule facilement : $\frac{\partial J}{\partial U_i} = (U_iM - Y_{i\bullet})M^\top + \lambda\mathbb{1}$. Ainsi en annulant la dérivée et en isolant U_i de l'équation nous obtenons de façon analytique la règle de mise à jour suivante : $U_i = (Y_iM^\top)\ (MM^\top + \lambda I)^{-1}$. La règle de mise à jour de M_j 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 et ensuite l'ensemble des vecteurs M_j jusqu'à convergence. Ce type d'algorithme est décrit dans Takane et al. (1977).

Des versions évoluées de cet algorithme ont été proposées dans la littérature (Hu et al., 2008). Ces méthodes sont optimisées pour la factorisation de matrices creuses et permettent de gérer les données d'appréciation implicite de très grandes tailles. Elles sont donc particulièrement bien adaptées aux problèmes de recommandation.

3.2.4 Algorithme multiplicatif

L'idée générale des algorithmes multiplicatifs est, contrairement aux méthodes additives telles que les descentes de gradient ou les algorithmes de Newton où l'on avance dans le processus d'optimisation par ajout, de progresser dans le processus d'optimisation en multipliant les paramètres à optimiser par un terme (qui tendra vers la valeur 1) à chaque itération. Lorsque l'on cherche à factoriser $Y \in \mathbb{R}^{n\times p}$ sous la forme UM (avec $U \in \mathbb{R}^{n\times d}$ et $M \in \mathbb{R}^{d\times p}$), le

9. http ://sifter.org/ simon/journal/20061211.html
10. http ://code.google.com/p/pyrsvd/
11. http ://www.cofirank.org/

processus d'optimisation suit les règles de mise à jour de U et M suivantes :

$$\begin{aligned} U &\leftarrow U \otimes \frac{\nabla_U^- J(Y, UM)}{\nabla_U^+ J(Y, UM)} \\ M &\leftarrow M \otimes \frac{\nabla_M^- J(Y, UM)}{\nabla_M^+ J(Y, UM)} \end{aligned} \tag{8}$$

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).

Ce genre de méthode est utilisé notamment dans des problèmes de factorisation avec contraintes de posivité, comme la factorisation non-négative (NMF). La NMF est utilisée dans plusieurs types d'applications, notamment pour la décomposition de signaux musicaux (Févotte et al., 2009). Elle a été également utilisée pour résoudre des problèmes de recommandation (Pessiot et al., 2006).

3.3 Factorisation dans les réseaux sociaux

Depuis le succès des réseaux sociaux en ligne et la masse de données qui en résulte, il est devenu important de disposer de systèmes de recommandation permettant d'aider les utilisateurs dans leurs choix. La recommandation dans les réseaux sociaux se concentre principalement sur trois aspects :

- 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 considèrera 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 une des centres d'intérêt de l'utilisateur pour qui la recommandation a été effectuée.

Tout comme pour la recommandation dans un contexte non social, il existe toute sorte de modèle de recommandation. Nous nous intéresserons ici aux méthodes de factorisation dans le cadre de la recommandation dans les réseaux sociaux.

Avant de rentrer dans les détails, posons quelques notations. 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 issue du graphe social et définie par $S_{ij} = 1$ si $i \in \mathcal{F}_j$, 0 sinon, ainsi que la matrice A issue d'un graphe de confiance et définie par $A_{ij} \in [0, 1]$ si $S_{ij} = 1$, et 0 sinon.

Ainsi, en plus des données d'appréciations de la matrice Y et des données contextuelles liées aux articles ou aux utilisateurs (si elles sont disponibles) dont nous disposons traditionnellement, nous avons maintenant le graphe social du réseau qui est une source d'information supplémentaires pour améliorer la recommandation d'article. Nous allons voir maintenant les différentes approches qui permettent de prendre en compte ce réseau social dans le processus lié à l'optimisation.

3.3.1 Ajout de contraintes au problème / de termes à J

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.

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.

$$\min_{f,h} \sum_{(i,j)\in\mathcal{Y}} l(Y_{ij}, F_{ij}) + \lambda_A \sum_{(i,k)\in\mathcal{F}} l(A_{ik}, H_{ij}) + \Gamma(f,h),$$

où Γ 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 : $f_{ij} = U_i M_j + x_i^\top W x_j$ et $h_{i,j} = U_i U_{i'}^\top + x_i^\top V x_{i'}$, où x_i est un ensemble de variables observables sur les utilisateurs i, et où x_j un ensemble de variables observables sur les articles j , c'est-à-dire des données contextuelles dans les deux cas, et où V et W sont des matrices carrées. Le problème consiste à optimiser les matrices U,M,W et V. 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 $x_j = 0\ \forall j$) hormis les données sociales. Le problème peut se formuler comme dans l'équation 9.

$$\min_{U,M} \sum_{(i,j)\in\mathcal{Y}} l(Y_{ij}, U_i M_j) + \lambda_A \sum_{(i,k)\in\mathcal{F}} l(A_{ik}, U_i U_j^\top) + \lambda_\Omega \Omega(U,M). \tag{9}$$

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 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, nous pourrions 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.

Dans le même esprit, Purushotham et al. (2012) ajoutent une contrainte différente et obtient un problème d'optimisation selon trois facteurs U, M et $T \in \mathbb{R}^{d\times n}$. Ce problème est de la forme suivante :

$$\min_{U,M,T} \sum_{i,j} c_{ij}(Y_{ij}, U_i M_j)^2 + \lambda_A \sum_{i,k} b_{ij}(A_{ik}, U_i T_k)^2 + \lambda_\Omega \Omega(U,M,T). \tag{10}$$

où les cœfficients c_{ij} et b_{ik} pondèrent les pénalités suivant l'état d'un utilisateur par rapport à un article (pour c), c'est-à-dire suivant l'appréciation d'un utilisateur pour un article, ou pondèrent l'état d'un utilisateur par rapport à un autre utilisateur (pour b), c'est-à-dire suivant le lien d'amitié éventuel entre deux utilisateurs. Les auteurs utilisent un processus d'optimisation alternée pour mettre à jour les trois différents facteurs.

Cet ajout d'une contrainte au problème d'optimisation se retrouve également dans Ma et al. (2011). Les auteurs 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 :

$$\min_{U,M} \sum_{(i,j)\in\mathcal{Y}} l(Y_{ij}, U_i M_j) + \lambda_A \sum_{i=1}^{n} \|U_i - \frac{1}{|\mathcal{F}_i|} \sum_{k\in\mathcal{F}_i} U_k\|_F^2 + \lambda_\Omega \Omega(U, M), \tag{11}$$

$$\min_{U,M} \sum_{(i,j)\in\mathcal{Y}} l(Y_{ij}, U_i M_j) + \lambda_A \sum_{i=1}^{n} \|U_i - \frac{\sum_{k\in\mathcal{F}_i} sim(i,k) U_k}{\sum_{k\in\mathcal{F}_i} sim(i,k)}\|_F^2 + \lambda_\Omega \Omega(U, M), \tag{12}$$

$$\min_{U,M} \sum_{(i,j)\in\mathcal{Y}} l(Y_{ij}, U_i M_j) + \lambda_A \sum_{i=1}^{n} \sum_{k\in\mathcal{F}_i} sim(i,k) \|U_i - U_k\|_F^2 + \lambda_\Omega \Omega(U, M). \tag{13}$$

Dans les deux derniers cas, $sim(i,k)$ est une fonction de similarité entre l'utilisateur i et l'utilisateur k, et plusieurs choix pour cette fonction sont proposés pour l'article. Cette similarité est calculée dans chaque cas à partir des informations présentes dans la matrice de préférence et non pas à partir de celles du graphe social. Ainsi résolvant les problèmes 12 et 13, les variables latentes d'utilisateurs qui sont amis auront tendance à avoir des valeurs proches. Dans l'équation 11, les variables latentes des utilisateurs auront tendance à ressembler à la moyenne des variables latentes de ses amis. Le processus d'optimisation se fait comme dans Yang et al. (2011) à travers un algorithme de descente de gradient stochastique.

On retrouve une fonction coût qui suit le même esprit dans Jamali et Ester (2010) où les auteurs choisissent d'optimiser le problème suivant :

$$\min_{U,M} \sum_{(i,j)\in\mathcal{Y}} \left(Y_{ij} - g(U_i M_j)\right)^2 + \lambda_A \sum_{i=1}^{n} \|U_i - \sum_{k\in\mathcal{F}_i} A_{ki} U_k\|^2 + \lambda_\Omega \Omega(U, M) \tag{14}$$

Où g est la fonction logistique. Ce problème ressemble à celui de l'équation 11, à ceci près qu'ici la moyenne des amis est pondérée par le niveau de confiance accordé aux amis par l'utilisateur. Les auteurs choisissent également d'appliquer une algorithme de descente de gradient pour résoudre ce problème.

3.3.2 Modification de la fonction de décision

D'autres travaux visent à modifier la fonction de décision pour faire intervenir les informations sociales plutôt que d'ajouter des contraintes. Ceci aura bien sûr une incidence sur la fonction coût associée.

C'est par exemple le cas de Ma et al. (2009) qui, dans le cadre de la recommandation d'articles à travers un réseau social, modifie la fonction de décision afin que la prédiction pour un utilisateur soit une moyenne des prédictions de ces amis : $\hat{Y}_{ij} = \sum_{k\in\mathcal{F}_i} \frac{U_k M_j}{|\mathcal{F}_i|}$. Le critère d'attache aux données dans la fonction à optimiser est alors différent. Les auteurs ont choisis le critère suivant :

$$\mathcal{L}(Y, U, M) = \sum_{(i,j)\in\mathcal{Y}} Y_{ij} - g\left(\gamma U_i M_j + (1-\gamma) \sum_{k\in\mathcal{F}_i} \frac{U_k M_j}{\mathcal{F}_i}\right), \tag{15}$$

où g est la fonction logistique et γ une constante d'équilibrage.

4 Recommandation avec préférences et liens d'amitié

4.1 Problème Tuenti

Dans la section "expériences", nous utilisons les données du service de lieux du réseau social en ligne Tuenti, 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.

4.1.1 Les données

Pour tester notre modèle, nous avons utilisé la matrice d'interaction lieu-utilisateur de *Tuenti*, comme matrice Y, 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 en moyenne 4 lieux sur son profil et 20 amis.

Nous repartons de la fonction objectif de l'équation 1 où F_{ij} désignait le score obtenu par le produit scalaire $F_{ij} = U_i M_j$. $U \in \mathbb{R}^{n \times d}$ et $M \in \mathbb{R}^{d \times p}$ sont les facteurs latents respectivement des utilisateurs et des articles.

4.2 L'influence des amis

Le principal défi de ce travail est d'inclure l'influence du graphe social dans le modèle de factorisation matricielle. Dans ce but, nous modifions la fonction de décision pour inclure l'influence du réseau social. Cette fonction de décision devient alors :

$$F_{ij} = U_i M_j + \sum_{k \in \mathcal{F}_i} \frac{\alpha_{ik}}{|\mathcal{F}_i|} U_k M_j \tag{16}$$

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' « affinité » (*homophily*) dans le réseau social, il est raisonnable de supposer que certaines des préférences d'un utilisateur ne sont pas exprimées dans les données mais peuvent en revanche être exprimées dans celles 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 16 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 particulier dans les réseaux sociaux en ligne où les utilisateurs auront tendance à avoir un nombre assez conséquent d'amis, et nous nous attendons à ce que ces utilisateurs aient des goûts similaires avec seulement une fraction de leurs 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.

À partir de cette fonction de décision et la fonction objectif de l'équation 1, nous posons une nouvelle fonction objectif dépendant des facteurs U et M, ainsi que du poids α_{ik}. Nous définissons la matrice A telle que $A_{ik} = \alpha_{ik}, \forall i, \forall k \in \mathcal{F}_i$, 0 ailleurs.

$$\min_{U,M,A} J = \sum_{(i,j)\in\mathcal{Y}} c_{ij}\big(U_i M_j + \sum_{k\in\mathcal{F}_i} \frac{\alpha_{ik} U_k M_j}{|\mathcal{F}_i|} - Y_{ij}\big)^2 + \Omega_{U,M,A} \tag{17}$$

où $\Omega_{U,M,A} = \lambda_1\|U\|_F^2 + \lambda_2\|M\|_F^2 + \lambda_3\|A\|_F^2$ est un terme régularisant et c_{ij} est une constante définie pour donner plus de poids dans le coût aux données observées $Y_{ij} = 1$ qu'aux valeurs inconnues $Y_{ij} = 0$.

4.3 Optimisation

Bien que l'équation 17 ne soit pas convexe selon U, M et A, elle l'est indépendamment selon chacun des trois paramètres à condition de fixer les deux autres. 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 s'il ne le connait 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 de faible *negative feedback*.

Nous optimisons la fonction objectif de l'équation 17 en utilisant le processus de Gauss-Seidel suivant : nous fixons alternativement deux des trois paramètres (U, M 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 M et la matrice poids A. Pour obtenir les bonnes règles de mise à jour pour chacun des trois paramètres (U_i, M_j et $\alpha_{ii'}$), nous devons calculer les dérivées partielles de la fonction objectif par rapport aux différents paramètres.

4.3.1 Mise à jour de la matrice U

Pour trouver la règle de mise à jour d'un vecteur facteur pour un unique utilisateur i, U_i, nous calculons la dérivée de la fonction objectif par rapport au facteur utilisateur et l'annulons.

Nous pouvons ensuite analytiquement résoudre cette expression par rapport à U_i. 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 $C^i_{jj} = c_{ij}$.

$$U_i = \left(Y_{i\bullet} C^i M^T - \frac{A_i U M C^i M^T}{|\mathcal{F}_i|} \right) (M C^i M^T + \lambda_1 I)^{-1} \tag{18}$$

Dans cette règle de mise à jour, le vrai problème n'est pas l'inversion de la matrice $d \times d$ (qui a une complexité en $O(d^3)$) mais le calcul de la matrice MC^iM^T (qui semble à première vue être de complexité en $O(p \times d^2)$).

Ce produit est trop coûteux même pour de petites données comme il faut le calculer pour chaque utilisateur. Dans l'esprit de Hu et al. (2008), nous remplaçons MC^iM^T par $MM^T + M(C^i - I)M^T$. Comme le produit MM^T 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 intelligement c_{ij}, le produit $M(C^i - I)M^T$ peut être calculé efficacement. Fixer le poids à $c_{ij} = 1 + \beta y_{ij}$ est un choix intéressant (β est une constante donnant plus ou moins de poids aux données connues). En effet, les termes diagonaux de $C^i - I$ seront zéro pour chaque j où $y_{ij} = 0$. Ainsi il ne sera pas nécessaire de calculer $M(C^i - I)M^T$, mais seulement $M_{\mathcal{Y}_i}(C^i - I)_{\mathcal{Y}_i} M^T_{\mathcal{Y}_i}$, où $\mathcal{Y}_i$ est l'ensemble des articles qu'un utilisateur i apprécie. Étant donné que la matrice Y est parcimonieuse, nous avons $|\mathcal{Y}_i| \ll p$. Ceci donne une complexité en $O(|\mathcal{Y}_i|\ d^2)$ linéaire par rapport au nombre d'articles que l'utilisateur i apprécie.

4.3.2 Mise à jour de la matrice M

Pour mettre à jour la matrice M, 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 :

$$L(U, M, A) = \sum_{i,j} c_{ij} (U'_i M_j - Y_{ij})^2$$

Le calcul de la dérivée partielle par rapport à M_j est alors direct et la règle de mise à jour peut facilement s'écrire sous forme matricielle en définissant la matrice diagonale C^j par $C^j_{ii} = c_{ij}$ [12]. Ainsi la règle de mise à jour de M_j est la suivante :

$$M_j = (U'^T C^j U' + \lambda_2 I)^{-1} U'^T C^j Y_{\bullet j} \tag{19}$$

Pour calculer le produit coûteux, nous proposons de réutiliser l'astuce décrite plus haut en réécrivant $U'^T C^j U' = U'^T U' + U'^T_{\mathcal{Y}_j} (C^j - I)_{\mathcal{Y}_j} U'_{\mathcal{Y}_j}$, 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 $U'^T_{\mathcal{Y}_j} (C^j - I)_{\mathcal{Y}_j} U'_{\mathcal{Y}_j}$ est en $O(|\mathcal{Y}_j| \times d^2)$.

4.3.3 Mise à jour de la matrice A

Nous allons proposer dans cette partie deux approches pour mettre à jour la matrice A.

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

Une approche pour mettre à jour A consiste à travailler ligne par ligne, c'est-à-dire mettre à jour $A_{i\bullet}$ pour chaque utilisateur i. Dans notre modèle, mettre à jour A consiste à modifier les poids des amitiés déjà connues, et non découvrir de nouveaux liens d'amitié éventuels. Ainsi la mise à jour de $A_{i\bullet}$ peut être réduite à la mise à jour de $A_{i\mathcal{F}_i}$. Nous annulons la dérivée partielle comme précédemment pour obtenir la règle de mise à jour suivante sous forme matricielle :

$$A_{i\mathcal{F}_i} = (Y_{i\bullet}C^iM^TU_{\mathcal{F}_i}^T - U_iMC^iM^TU_{\mathcal{F}_i}^T) \left(\frac{U_{\mathcal{F}_i}MC^iM^TU_{\mathcal{F}_i}^T}{|\mathcal{F}_i|} + \lambda_3\right)^{-1} \tag{20}$$

Notons à nouveau que le coût de calcul du produit $U_iMC^iM^TU_{\mathcal{F}_i}^T$, peut être réduit si nous appliquons la même astuce que celle des règles de mise à jour de U_i et de M_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 $O(|\mathcal{F}_i|^3)$, 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 problématique.

Une approche pour mettre à jour A consiste non pas à calculer les termes de A utilisateur par utilisateur mais terme à terme, c'est-à-dire mettre à jour $\alpha_{ii'}$ pour deux utilisateurs i et i' donnés. À nouveau, nous calculons 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 :

$$\alpha_{ii'} = \left(Y_{i\bullet} - U_iM - \sum_{\substack{k\in\mathcal{F}_i \\ k\neq i'}} \frac{\alpha_{ik}U_kM}{|\mathcal{F}_i|}\right) C^iM^TU_{i'}^T \left(\frac{U_{i'}MC^iM^TU_{i'}^T}{|\mathcal{F}_i|} + \lambda_3\right)^{-1} \tag{21}$$

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 MC^iM^T. Celui-ci peut en effet être réécrit comme $MC^iM^T = MM^T + M_{\mathcal{Y}_i}(C^i - I)_{\mathcal{Y}_i}M_{\mathcal{Y}_i}^T$, 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 21 croît linéairement par rapport au nombre d'amis de i, tandis que la complexité dans l'équation 20 est polynomiale par rapport au nombre d'amis de i. Nous en concluons que la règle de mise à jour de l'équation 21 est clairement plus performante que celle de l'équation 20 en terme de complexité. Notons finalement que les paramètres α fournissent une mesure relative de l'influence (ou de la confiance) d'un utilisateur sur ses amis.

4.3.4 Algorithme

À partir de ces règles de mise à jour, nous posons l'algorithme d'optimisation de la fonction objectif 17 selon U, M et A. Nous discuterons de l'initialisation des différents paramètres dans la partie expérimentation.

Algorithme 1 Socially-Enabled Collaborative Filtering

Entrées : Y and $\mathcal{F}$
initialiser U,M et A
répéter
  pour tout utilisateur $i \in \mathcal{U}$ **faire**
    mettre à jour U_i comme spécifié dans l'équation (18)
  fin pour
  pour tout article $j \in \mathcal{M}$ **faire**
    mettre à jour M_j comme spécifié dans l'équation (19)
  fin pour
  pour tout utilisateur $i \in \mathcal{U}$ **faire**
    pour tout utilisateur $k \in \mathcal{F}_i$ **faire**
      mettre à jour α_{ik} comme spécifié dans l'équation (21)
    fin pour
  fin pour
jusqu'à convergence

4.3.5 Prédiction

Notons qu'une fois le modèle appris, les prédictions peuvent être réalisées en utilisant la fonction de décision de l'équation 16. Ceci nécessite qu'au moment de la prédiction l'ensemble du réseau social soit en mémoire, et de nombreux accès mémoire ralentiront le calcul du score. Les systèmes de recommandation commerciaux modernes n'ont généralement que quelques millisecondes pour fournir des recommandations. Pour accélerer cette opération nous pouvons simplement pré-calculer 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 simplement $F_{ij} = U'_i M_j$, et le modèle est simplement composé de la matrice U' et la matrice M et non plus de U, M et A.

4.4 Expérimentations

4.4.1 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 derniers 25% des lieux appréciés par 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.

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'ordonnancement (de *ranking*). Une métrique d'ordonnancement populaire pour ce type de données est la métrique *Mean Average Precision* (MAP) qui est particulièrement adaptée à l'ordonnancement de recommandation parce qu'elle accorde de l'importance au fait d'avoir en tête de liste des articles pertinents.

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. Comme nous n'avons pas de données de *rating*, la métrique RANK peut s'écrire, dans notre cas, comme dans l'équation 22.

$$\text{RANK} = \frac{\sum_{i,j} Y_{ij} rank_{ij}}{|\mathcal{Y}|} \tag{22}$$

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

4.4.2 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. Nous pourrons 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*.

Nous avons également comparé les performances de notre méthodes avec quelques méthodes présentées dans la section 3.3. Nous nous comparons aux travaux de Yang et al. (2011), de Ma et al. (2011) et de Ma et al. (2009).

Yang et al. (2011) a testé sa méthode avec différentes fonctions coût. Nous avons choisi celle qui donnait les meilleurs résultats sur leur problème : un coût logistique et une norme $l2$ pour le terme régularisant. Pour la suite, nous nommerons cette méthode *LLA*.

Nous avons choisi la méthode de Ma et al. (2011) donnant les meilleures performances, c'est-à-dire celle qui implique une régularisation individuelle. Ici aussi une descente de gradient stochastique est utilisée pour optimiser la fonction objectif. Dans la suite de l'article, nous appelerons cette méthode *RSR*.

La dernière méthode 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 d'appréciation explicite 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 M 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 l'astuce de 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 également comparé notre modèle à un point de comparaison : le prédicteur moyen, qui recommande les lieux les plus populaires aux utilisateurs.

4.5 Résultats d'expérience

Nous avons réalisé une validation croisée pour la sélection de modèle. Nous initialisons aléatoirement U et M selon une loi uniforme entre 0 et 1. Pour l'initialisation du poids des amis α_{ij}, nous avons déterminé par validation croisée, sur un sous-ensemble de l'ordre de 10% des données, que les meilleures performances sont atteintes en initialisant à $\alpha_{ij} = 1$. Nous avons estimé la valeur optimale pour le paramètre β (utilisé dans le coefficient c_{ij}) de la même façon. Cette valeur optimale est $\beta = 30$, en accord avec les métriques MAP et RANK (cf. figure 2). Nous avons utilisé cette valeur de β pour toutes les expériences.

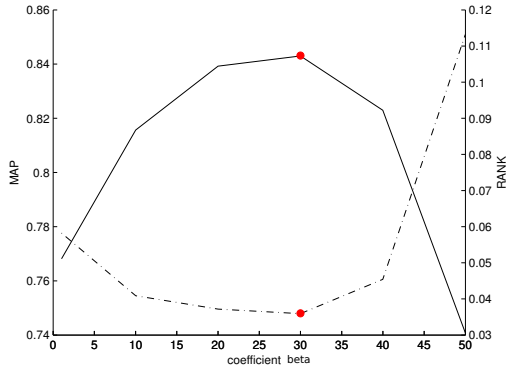

FIG. 2 – *la MAP et RANK par rapport à la valeur du coefficient β*

Nous avons également légèrement modifié le processus d'apprentissage. Au lieu de mettre à jour alternativement U, M et A, nous fixons A et optimisons alternativement U et M jusqu'à convergence. Une fois la convergence atteinte, nous mettons A à jour pour U et M. nous itérons ce processus jusqu'à une convergence globale. Nous avons constaté que ce processus conduisait à un taux de convergence plus rapide.

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 écarts types. Ainsi nous avons testé les différentes méthodes avec plusieurs valeurs de d et tracé la courbe des résultats sur la figure 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* in 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*. Étonnement *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.

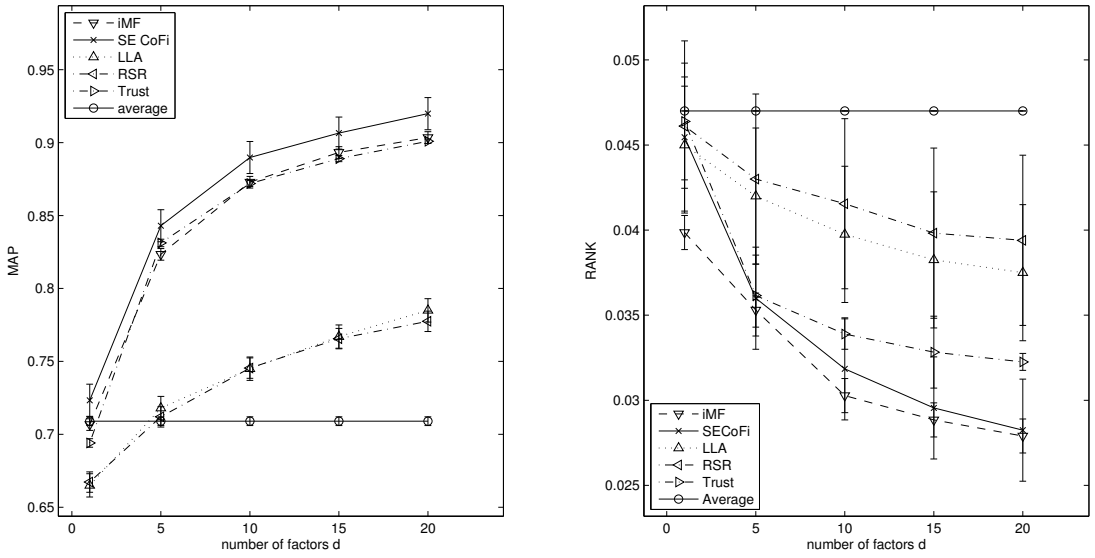

FIG. 3 – *la MAP et le RANK des différentes méthodes par rapport au nombre de facteurs* d

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 *SE CoFi* 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 *SE CoFi* 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 l'optimal du paramètre de régularisation de *SE CoFi* était inchangé, 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 l'optimal de λ trouvé pour une valeur donnée de d, nous pouvons fixer λ à cette valeur et optimiser par rapport à d. Ce n'est pas le cas pour les méthodes de type SGD.

Les performances de notre méthode justifient l'utilisation du graphe social, particulièrement en terme de MAP. 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.

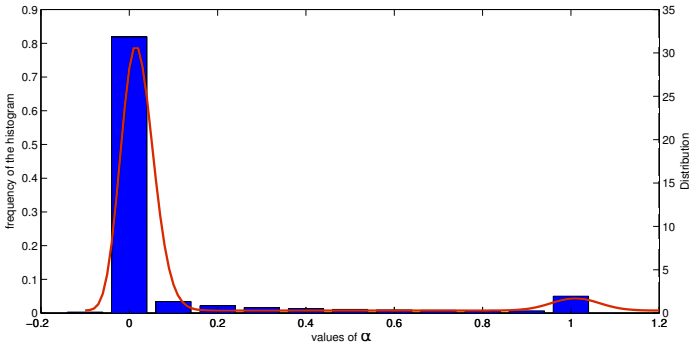

FIG. 4 – *Distribution des valeurs de α. La distribution est bimodale avec une grande majorité de valeurs positives*

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 confiance négative 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.

Nous avons tracé la distribution des valeurs de α sur la figure 4. Nous observons que la distribution est bimodale. Comme on peut le voir dans la figure 4, 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, c'est-à-dire qu'elles pourraient indiquer l'absence d'influence d'un utilisateur sur un de ses amis. Nous pouvons conclure que le principe d'affinité est confirmé pour notre modèle, c'est-à-dire que 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).

5 Conclusion

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. Nous avons établi une méthode d'optimisation simple mais efficace et l'avons testé sur un ensemble de données réelles de grande échelle. 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 en terme de MAP.

Dans de futurs travaux, nous envisageons d'améliorer notre méthode en utilisant les données contextuelles et 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 pourrons construire un graphe mettant en

relation les lieux (qui pourrait être vu comme une matrice d'"amitié" entre articles) et l'utiliser pour améliorer les performances. Nous pourrions également utiliser les valeurs calculées des α_{ij} pour réaliser de la recommandation d'amis, en appuyant le fait que ces valeurs représentent une forme de mesure relative de confiance (*Trust*) entre un utilisateur et ses amis.

Références

Aggarwal, C. (2011). An introduction to social network data analytics. *Social Network Data Analytics 1*, 1–15.

Candès, E., X. Li, Y. Ma, et J. Wright (2011). Robust principal component analysis ? *J. ACM 58*(3), 11 :1–11 :37.

Cravo, G. (2009). Matrix completion problems. *Linear Algebra and Its Applications 430*(8), 2511–2540.

Cremonesi, P., Y. Koren, et R. Turrin (2010). Performance of recommender algorithms on top-n recommendation tasks. In *Proc. of RecSys'10)*, pp. 39–46.

Févotte, C., N. Bertin, et J. Durrieu (2009). Nonnegative matrix factorization with the itakura-saito divergence : With application to music analysis. *Neural Computation 21*(3), 793–830.

Hu, Y., Y. Koren, et C. Volinsky (2008). Collaborative filtering for implicit feedback datasets. In *Proc. of ICDM '08*, pp. 263–272. Ieee.

Jamali, M. et M. Ester (2010). A matrix factorization technique with trust propagation for recommendation in social networks. In *Proc. of RecSys '10*, New York, NY, USA, pp. 135–142. ACM.

Kaptein, M. et D. Eckles (2010). Selecting effective means to any end : futures and ethics of persuasion profiling. In *Proc. of PERSUASIVE '10*, Berlin, Heidelberg, pp. 82–93. Springer-Verlag.

Koren, Y., R. Bell, et C. Volinsky (2009). Matrix factorization techniques for recommender systems. *Computer 42*(8), 30–37.

Krohn-Grimberghe, A., L. Drumond, C. Freudenthaler, et L. Schmidt-Thieme (2012). Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *Proc. of WSDM '12*, pp. 173–182.

Lazarsfeld, P., R. Merton, et al. (1954). Friendship as a social process : A substantive and methodological analysis. *Freedom and control in modern society 18*(1), 18–66.

Lee, L. et D. Seung (2001). Algorithms for non-negative matrix factorization. *Advances in neural information processing systems 13*, 556–562.

Ma, H., I. King, et M. R. Lyu (2009). Learning to recommend with social trust ensemble. In *Proc. of SIGIR '09*, New York, NY, USA, pp. 203–210. ACM.

Ma, H., D. Zhou, C. Liu, M. R. Lyu, et I. King (2011). Recommender systems with social regularization. In *Proc. of WSDM '11*, New York, NY, USA, pp. 287–296. ACM.

Pessiot, J., N. Usunier, M. Amini, P. Gallinari, et al. (2006). Factorisation en matrices non-négatives pour le filtrage collaboratif. In *CORIA*, pp. 315–326.

Pu, P., L. Chen, et R. Hu (2012). Evaluating recommender systems from the user's perspective : survey of the state of the art. *User Modeling and User-Adapted Interaction 22*, 317–355.

Purushotham, S., Y. Liu, et C.-C. J. Kuo (2012). Collaborative topic regression with social matrix factorization for recommendation systems.

Recht, B., M. Fazel, et P. Parrilo (2010). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review 52*(3), 471–501.

Rendle, S. (2010). *Context-aware ranking with factorization models*, Volume 330. Springer.

Ricci, F., L. Rokach, et B. Shapira (2011). Introduction to recommender systems handbook. *Recommender Systems Handbook 1*, 1–35.

Singh, A. et G. Gordon (2008). A unified view of matrix factorization models. In *Proc. of the ECML PKDD '08*, pp. 358–373. Springer-Verlag.

Sra, S. (2011). Nonconvex proximal splitting : batch and incremental algorithms.

Srebro, N., J. Rennie, et T. Jaakkola (2005). Maximum-margin matrix factorization. In L. K. Saul, Y. Weiss, et L. Bottou (Eds.), *Proc. of NIPS '17*, Cambridge, MA. MIT Press.

Srebro, N. et A. Shraibman (2005). Rank, trace-norm and max-norm. In P. Auer et R. Meir (Eds.), *Proc. of Annual Conf. Computational Learning Theory*, Number 3559 in Lecture Notes in Artificial Intelligence, pp. 545–560. Springer-Verlag.

Takacs, G., I. Pilaszy, B. Nemeth, et D. Tikk (2009). Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research 10*, 623–656.

Takane, Y., F. Young, et J. De Leeuw (1977). Nonmetric individual differences multidimensional scaling : an alternating least squares method with optimal scaling features. *Psychometrika 42*(1), 7–67.

Valizadegan, H., R. Jin, R. Zhang, et J. Mao (2009). Learning to rank by optimizing ndcg measure. *Advances in Neural Information Processing Systems 22*, 1883–1891.

Weimer, M., A. Karatzoglou, et A. Smola (2008). Improving maximum margin matrix factorization. *Machine Learning 72*(3), 263–276.

Yang, S.-H., B. Long, A. Smola, N. Sadagopan, Z. Zheng, et H. Zha (2011). Like like alike : joint friendship and interest propagation in social networks. In *Proc. of WWW '11*, New York, NY, USA, pp. 537–546. ACM.

Summary

Nowadays a lot of web sites provide elaborated and personalized advices using recommender systems, which are softwares and methods automatically suggesting items that could be useful for a user. The advent of Online Social Networks has added a new approach to recommendation whereby the structure of the social network is used as a source of information. This article consists in a rapid survey on the state-of-the-art model-based recommendation methods, and in particular factorization methods for recommendation. It also consists in a presentation of a approach which takes advantage of the social network and performs predictions on implicit feedback datas. We tested this model on a real life data set where it outperformed alternative state-of-the-art methods.