Représentations Gaussiennes pour le Filtrage Collaboratif

Hadrien Titeux [1], **Benjamin Piwowarski** [2], **Patrick Gallinari** [3]

1. *Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France*
   hadrien.titeux@lip6.fr
2. *CNRS, Sorbonne Université, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France*
   benjamin.piwowarski@lip6.fr
3. *Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France*
   patrick.gallinari.titeux@lip6.fr

RÉSUMÉ. *La plupart des systèmes de filtrage collaboratifs, comme par exemple la factorisation matricielle, utilisent des représentations vectorielles pour les articles et les utilisateurs. Ces représentations sont déterministes, et ne permettent pas de modéliser l'incertitude des représentations apprises, ce qui peut être utile quand un utilisateur a évalué un petit nombre d'articles (problème du démarrage à froid), ou quand le modèle est confronté à des informations contradictoires concernant le comportement d'un utilisateur ou les évaluations d'un utilisateur. Dans cet article, nous nous appuyons sur des représentations gaussiennes qui modélisent cette incertitude, et définissons un cadre d'apprentissage et d'inférence approprié à ce type de représentations. Nous montrons en outre que ce modèle fonctionne bien sur trois collections représentatives, et analysons les représentations apprises par le modèle.*

ABSTRACT. *Most collaborative filtering systems, such as matrix factorization, use vector representations for items and users. Those representations are deterministic, and do not allow modeling the uncertainty of the learned representation, which can be useful when a user has a small number of rated items (cold start), or when there is conflicting information about the behavior of a user or the ratings of an item. In this paper, we leverage recent works in learning Gaussian embeddings for the recommendation task. We show that this model performs well on three representative collections and analyze learned representations.*

MOTS-CLÉS : *systèmes de recommandation, représentations gaussiennes, apprentissage d'ordonnancement.*

KEYWORDS: *collaborative filtering, gaussian embeddings, learning to rank.*

1. Introduction

Les systèmes de recommandation aident les utilisateurs à trouver des items (livres, films, etc.) qu'ils pourraient apprécier dans des bases de données de grande taille. Les méthodes utilisées pour recommander des items peuvent être globalement divisées en trois catégories: le filtrage collaboratif, les méthodes basées sur le contenu, et les méthodes hybrides (Ricci *et al*., 2011). Parmi les systèmes de recommandation, le filtrage collaboratif est le plus largement utilisé. Cela s'explique par sa capacité à exploiter les notes données par les utilisateurs, alors que les approches basées sur le contenu reposent sur la construction de profils pour les utilisateurs (ou les items). Parmi les systèmes de recommandation collaboratifs, ceux qui nous intéressent tout particulièrement sont ceux qui représentent les utilisateurs et les items dans un espace latent commun. La factorisation matricielle et ses dérivés sont des exemples typiques de ce genre de modèles. L'hypothèse principale est que le produit scalaire entre la représentation latente d'un utilisateur et d'un item est corrélé avec la note que l'utilisateur donnerait à cet item s'il devait l'évaluer. L'intérêt principal de ces modèles est leur capacité à intégrer diverses sources d'information (Zhang *et al*., 2016), ainsi que leurs bonnes performances (tant en termes de rapidité que d'efficacité (Ricci *et al*., 2011)). Les méthodes les plus fructueuses sont basées sur les approches d'apprentissage d'ordonnancement (*learning-to-rank*) telles que (Rendle *et al*., 2009 ; Weimer *et al*., 2007).

Malgré leurs succès, une limitation importante de ces modèles est leur incapacité à traiter l'incertitude. Même quand ils sont basés sur des modèles probabilistes, comme (Salakhutdinov, Mnih, 2007 ; Rendle *et al*., 2009), ils ne font que l'hypothèse d'un à priori gaussien sur les représentations des utilisateurs et des items, et se limitent à apprendre une représentation déterministe. L'apriori est uniquement utilisé comme un terme de régularisation sur les représentations des utilisateurs et des items.

L'intégration de l'incertitude est motivée par plusieurs raisons. L'incertitude peut être liée au manque d'information (nouveaux utilisateurs ou items, ou des utilisateurs qui n'auraient noté aucun item dans un genre donné), soit à des contradictions entre les notes données. Pour illustrer cette dernière éventualité, considérons un utilisateur qui n'aime qu'une partie des films policiers, sans motif clair (sans sous-genre). Avec les approches habituelles, un tel utilisateur aurait une composante de sa représentation à 0 (correspondant à la dimension de l'espace latent qui modélise le gout pour les films policiers). Avec notre approche, nous avons toujours une moyenne à 0, mais avec une forte variance. Notre hypothèse est que ces deux facteurs - le démarrage à froid et les informations conflictuelles - auront pour conséquence des représentations avec une confiance qui peut différer, et que cette incertitude est importante pour la tâche de recommandation.

En outre, utiliser une densité plutôt qu'un point fixe ouvre la voie à des nouvelles approches pour la recommandation. Tout d'abord, à la place de calculer un score pour chaque item, le modèle peut calculer une distribution de probabilité, ainsi que la covariance entre les scores de deux items différents. Nous exploitons cela pour diversifier

la liste de résultats, à partir d'approches dites *Portfolio* (Wang, Zhu, 2009). Ensuite, ce genre de modèle est intéressant en ce qu'il peut servir de base pour intégrer différentes sources externes d'information. Finalement, une composante temporelle peut être intégrée en supposant que la variance de la représentation augmente avec le temps si aucune nouvelle information (c.-à-d. des évaluations utilisateur) n'est fournie.

Notre modèle utilise des représentations gaussiennes; celles-ci ont été utilisées récemment (et avec succès) pour apprendre des graphes de connaissance de mots (Vilnis, McCallum, 2014 ; He *et al*., 2015), ou les représentations de noeuds d'un graphe (Dos Santos *et al*., 2016a). Plus précisément, la représentation de chaque utilisateur et chaque item correspond à une distribution normale (gaussienne) où la moyenne et la matrice de covariance sont apprises : les utilisateurs (et les items) sont représentés par une densité, et non plus par un point fixe comme pour les autres modèles de recommandation. La variance est une mesure de l'incertitude associée à la représentation des utilisateurs/items.

Dans cet article, après avoir présenté un bref état de l'art, nous décrirons d'abord le modèle, puis montrerons expérimentalement qu'il donne de bons résultats en comparaison aux approches de l'état de l'art, et ceci sur trois jeux de données différentes. Ce travail reprend et étend les travaux présentés dans (Dos Santos *et al*., 2016b), dans lequel nos contributions étaient les suivantes : 1. L'utilisation d'un nouveau type de représentation pour les utilisateurs et les items en filtrage collaboratif ; 2. Un travail expérimental extensif qui démontre les bonnes performances de notre modèle 3. Une analyse qualitative qui explique comment la variance est utilisée dans le modèle . Dans cet article, nous exposons et étendons ce modèle en proposant un nouveau critère pour apprendre les représentations et en proposant d'utiliser les techniques d'optimisation de portfolio pour ordonner les items lors de l'évaluation du modèle. Nous montrons expérimentalement l'intérêt d'une telle approche.

2. État de l'art

Il existe deux familles de systèmes de recommandation (Adomavicius, Tuzhilin, 2005). D'une part, les systèmes de filtrage par contenu qui se basent sur la construction de profils explicites pour chaque utilisateur et item, comme par exemple l'âge et la profession des utilisateurs, ou le genre et la popularité des items. D'autre part, les systèmes de filtrage collaboratif qui construisent un profil pour les utilisateurs et les items selon l'ensemble des jugements. Le filtrage collaboratif donne d'excellents résultats lorsqu'il peut exploiter une très grande quantité de données, et tirer profit du voisinage explicite ou de toute mesure de similarité calculée à l'aide de l'historique des notes.

Parmi les systèmes de filtrage collaboratif, nous nous intéressons à ceux qui représentent les utilisateurs et les items dans un espace latent $\mathbb{R}^d$. Les modèles basés sur la factorisation matricielle sont un exemple typique de ce type de modèles. L'hypothèse principale est que le score d'un item est lié au produit scalaire entre la représentation d'un utilisateur et d'un item donnés. L'intérêt principal de ce type de modèle est leur

potentiel pour intégrer différentes sources d'information (Zhang *et al.*, 2016), ainsi que leurs bonnes performances (Ricci *et al.*, 2011).

Pour apprendre ces représentations, il est possible d'utiliser des modèles de régression, où le but est de prédire la note d'un item (Salakhutdinov, Mnih, 2007). Lorsque le but est de présenter les items que l'utilisateur est le plus susceptible d'aimer, il est plus intéressant d'utiliser des techniques d'apprentissage d'ordonnancement qui sont plus proches de mesures favorisant le haut du classement – et de la satisfaction réelle d'un utilisateur (Liu, 2011). Les méthodes d'apprentissage d'ordonnancement ont pour but d'apprendre la préférence d'un utilisateur entre deux items (*pairwise*) ou bien optimisent un substitut d'une métrique de RI (*listwise*).

En recommandation, ce type de techniques ont été explorées depuis une dizaine d'années (Karatzoglou *et al.*, 2013). Parmi celles-ci, nous pouvons citer le modèle Bayesian Personalized Ranking (BPR) qui vise à maximiser la probabilité a posteriori de préférence d'une paire d'éléments (Rendle *et al.*, 2009), où la probabilité qu'un utilisateur préfère un élément par rapport à un autre est égale à la sigmoïde de la différence des scores prédits. CliMF (Shi *et al.*, 2012) a exploité ce même principe de préférence, mais pour optimiser le rang réciproque au lieu d'un critère par paire. CliMF obtient de meilleures performances que BPR. Il est aussi possible d'optimiser des métriques plus complexes qui dépendent de la liste entière renvoyée par l'algorithme d'ordonnancement : en recommandation, CofiRank (Weimer *et al.*, 2007) optimise une borne inférieure du nDCG – normalized Discounted Cumulated Gain (Järvelin, Kekäläinen, 2002) – et obtient des performances similaires à CliMF. Dans notre modèle, nous exploitons BPR et SoftRank (Taylor *et al.*, 2008). Ce dernier optimise l'espérance du nDCG, mais n'a été utilisé que dans le contexte de l'apprentissage de classement pour la recherche d'information. Ce critère est plus adapté aux représentations gaussiennes car suppose que les scores ont une variance associée, ce qui est naturel dans notre approche.

Finalement, le problème de l'ordonnancement en inférence n'est pas trivial lorsqu'un modèle prédit une distribution de probabilité d'un score. C'est ce qui nous a poussé à étudier les modèles basés sur la théorie de l'optimisation de portfolio. Ceux-ci ont été utilisés en recherche d'information et en recommandation (Wang, Zhu, 2009), mais utilisent une covariance estimée en fonction des données ce qui pose des problèmes de complexité numérique. Dans notre approche, la covariance des scores découle directement de la représentation choisie, et permet donc d'utiliser de façon naturelle un tel formalisme.

3. Modèle

Dans notre modèle, les représentations des utilisateurs et des items sont des variables aléatoires, notées $\mathbf{X}_\bullet$, où $\bullet$ est soit un utilisateur u ou un item j. Contrairement aux autres modèles de recommandation, la représentation d'un utilisateur ou d'un item est une densité dans un espace vectoriel, et non un point fixe.

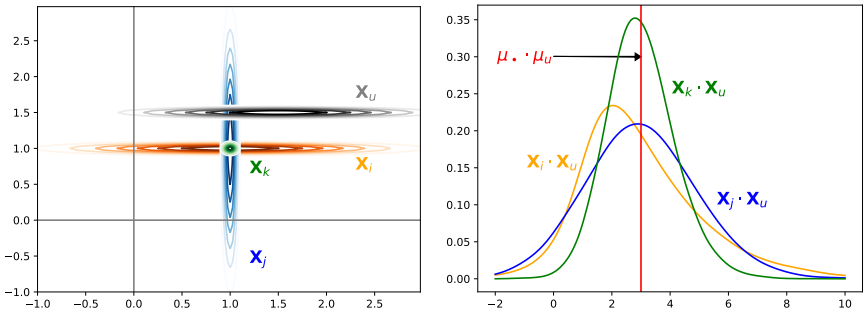

FIGURE *1. Exemple de représentations de trois items (i, j et k) ayant la même moyenne μ mais des variances différentes, et d'un utilisateur u, ainsi que la distribution de probabilité du produit scalaire (droite).*

Nous commençons par supposer que la représentation d'un utilisateur ou d'un item suit une distribution normale: pour tout utilisateur u et tout item i, $\mathbf{X}_u \sim \mathcal{N}(\mu_u, \Sigma_u)$ et $\mathbf{X}_i \sim \mathcal{N}(\mu_i, \Sigma_i)$. On suppose que $\Sigma_\bullet = \text{diag}(\sigma_{\bullet 1}, \ldots, \sigma_{\bullet d})$ est une matrice de covariance diagonale pour limiter la complexité du modèle (d est la dimension de l'espace latent): en pratique, nous avons $2d$ paramètres par utilisateur, et $2d + 1$ pour chaque item ($+1$ à cause du biais). L'ensemble des représentations (moyennes et variance) est noté Θ.

Pour illustrer l'intérêt d'une telle représentation, dans la figure 1, nous avons représenté trois items (même moyenne μ, mais covariance différente) et un utilisateur. Nous voyons que : (i) Lorsque la variance de l'item est faible (k), alors le score (produit scalaire $\mathbf{X}_u \cdot \mathbf{X}_k$) a une variance faible ; (ii) lorsque la dimension où la variance est haute dans chaque dimension, soit pour l'item, soit pour l'utilisateur (item j), alors la variance est la plus forte

Finalement, il nous reste à définir la distribution à priori sur les paramètres du modèle Θ. Nous ne considérons que des matrices de covariance diagonales, et pouvons donc considérer des à priori indépendants pour chaque moyenne $\mu_{\bullet k}$ et chaque variance $\Sigma_{\bullet kk}$. Dans ce cas, la distribution conjuguée d'une loi normale :

$$\left(\mu_{\bullet k}, \Sigma_{\bullet kk}^{-1}\right) \sim \text{NormalGamma}\left(\nu, \lambda, \alpha, \beta\right) \tag{1}$$

avec $\nu = 0$, $\lambda = 1$, $\alpha = 2$ et $\beta = 2$ – ces paramètres n'affectent que peu la solution, et ont été choisi de façon ne pas trop contraindre les représentations des utilisateurs et des items. En l'absence de données, cela fixerait, pour chaque composante, la moyenne à 0 et la variance à 1, ce qui correspond au mode de la distribution normal-gamma.

Dans la suite, nous définissons deux méthodes pour apprendre des représentations GER (*Gaussian Embeddings for Recommendation*) : la première est basée sur BPR, et optimise la probabilité d'observer des préférences entre paires d'items (GER-P). La

seconde est basée sur SoftRank, et optimise l'espérance de la métrique nDCG (GER-L).

3.1. Apprentissage de préférences (GER-P)

Le premier critère d'apprentissage que nous utilisons est basé sur les préférences (*pairwise learning-to-rank*), et a été proposé par (Rendle *et al.*, 2009) pour le filtrage collaboratif (Bayesian Personalized Ranking, BPR). Formellement, le maximum a posteriori des observations $\mathcal{D}$ est optimisé, c.-à-d. la probabilité $p(\Theta|\mathcal{D}) \propto p(\mathcal{D}|\Theta)\, p(\Theta)$ est maximisée, où $\mathcal{D}$ représente l'ensemble des triplets ordonnés (u, i, j) utilisés pour l'entrainement – l'utilisateur $u \in \mathcal{U}$ préfère l'item $i \in \mathcal{I}$ à l'item $j \in \mathcal{I}$. En utilisant l'hypothèse standard de l'indépendance des échantillons, étant donné les paramètres du modèle, on a

$$p(\mathcal{D}|\Theta) = \prod_{(u,i,j)\in\mathcal{D}} p(i >_u j|\Theta) \tag{2}$$

Dans (Rendle *et al.*, 2009), la probabilité que l'utilisateur u préfère l'item i à l'item j, notée $i >_u j$, est donnée par la sigmoïde de la différence des produits scalaires, c.-à-d.

$$p(i >_u j|\Theta) = \sigma(x_i \cdot x_u + b_i - x_j \cdot x_u - b_j) \tag{3}$$

où b_i (resp. b_j) est le biais pour l'item i (resp. j).

Étant donné que $\mathbf{X}_i$ et $\mathbf{X}_u$ sont des variables aléatoires, leur produit scalaire est aussi une variable aléatoire, et nous n'avons pas besoin d'utiliser la fonction sigmoïde de l'équation (3) pour obtenir la probabilité que u préfère i à j. Nous pouvons utiliser directement les variables aléatoires :

$$\begin{aligned} p(i >_u j|\Theta) &= p(\mathbf{X}_i \cdot \mathbf{X}_u + b_i > \mathbf{X}_j \cdot \mathbf{X}_u + b_j|\Theta) \\ &= p\Big(\underbrace{\mathbf{X}_u \cdot (\mathbf{X}_j - \mathbf{X}_i)}_{\mathbf{Z}_{uij}} < b_i - b_j \Big| \Theta\Big) \end{aligned}$$

Pour calculer l'équation ci-dessus, nous supposons que les représentations sont indépendantes, ce qui est raisonnable puisqu'elles sont données par les paramètres du modèle. De plus, nous approximons la distribution issue du produit scalaire par une distribution gaussienne en faisant correspondre les deux premiers moments (l'espérance et la variance) et en utilisant les résultats de (Brown, Rutemiller, 1977) qui a défini les moments du produit scalaire de deux variables aléatoires Gaussiennes; avec nos notations :

$$\mathbb{E}[\mathbf{Z}_{uij}] = \mu_u^\top(\mu_j - \mu_i) \tag{4}$$

$$\mathrm{Var}[\mathbf{Z}_{uij}] = 2\mu_u^\top(\Sigma_i + \Sigma_j)\,\mu_u + (\mu_j - \mu_i)^\top \Sigma_u(\mu_j - \mu_i) + \mathrm{tr}(\Sigma_u(\Sigma_j + \Sigma_i))$$

ce qui nous permet d'écrire:

$$p\left(i >_u j|\Theta\right) \approx \int_{-\infty}^{b_i - b_j} \mathcal{N}\left(x; \mathbb{E}\left[\mathbf{Z}_{uij}\right], \mathrm{Var}\left[\mathbf{Z}_{uij}\right]\right) \mathrm{d}x \tag{5}$$

qui est la fonction de répartition de la loi normale. Cette fonction peut être aisément dérivée par rapport à $\mu_{\bullet k}$ et $\sigma_{\bullet k}$ ($\bullet$ est un utilisateur ou un item), ce qui permet d'apprendre les représentations par descente de gradient.

À partir des équations (4) et (5), et en ignorant les biais, on peut voir que la différence des produits scalaires $\mu_u \cdot \mu_i$ et $\mu_u \cdot \mu_j$ est modulée par la variance de l'utilisateur et de l'item (tout particulièrement pour les composantes de la moyenne qui ont une grande magnitude) – plus grande elle est, plus proche de 0.5 sera la probabilité. Ceci est la principale différence avec les autres approches basées sur la factorisation matricielle.

3.2. Apprentissage de liste (GER-L)

Notre second critère d'apprentissage est basé sur SoftRank (Taylor *et al.*, 2008). SoftRank est un critère de liste qui optimise l'espérance du nDCG, qui est la métrique d'évaluation principale pour la recommandation. SoftRank exploite une modélisation probabiliste des scores d'items (pertinence du document dans l'article original qui a pour cadre la recherche d'information), en introduisant une variance. Dans notre modèle, le score de l'utilisateur est déjà une variable aléatoire, et nous pouvons utiliser directement et naturellement SoftRank, sans avoir besoin d'introduire un nouvel hyperparamètre comme dans (Taylor *et al.*, 2008). De plus, cette variance dépend de la représentation de l'utilisateur et de l'item et n'est pas fixe comme dans l'article original.

Avec l'hypothèse simplificatrice que les rangs des items sont indépendants, SoftRank optimise le nDCG [1], pour chaque utilisateur u:

$$\mathbb{E}\left[nDCG\right] = \frac{1}{CG_{\max}} \sum_{i \in I_u} g(i) \sum_{r \geq 1} d(r) p(\mathbf{R} = r|\mathbf{u}, \mathbf{i}, \mathbf{I}_u)$$

où I_u est l'ensemble des items à classer pour l'utilisateur u, $d(r)$ le poids (discount) associé au rang rank r, et $\mathbf{R}$ est le rang d'un item pour un utilisateur donné, ici l'article i et l'utilisateur u parmi les items I_u, et $p(\mathbf{R} = r|\mathbf{u}, \mathbf{i}, \mathbf{I}_u)$ la probabilité que l'item i soit au rang r sachant que l'utilisateur est u et l'ensemble des items à ordonner I_u.

Dans notre cas, une représentation doit être apprise pour chaque utilisateur et item. Cela implique que SoftRank ne peut pas être directement utilisé, puisque dans l'apprentissage d'ordonnancement dans le cadre de la recherche d'information ad-hoc un modèle unique est appris à partir de caractéristiques du couple document-question : la

1. Nous avons adapté les notations au cadre de recommandation

régularisation des paramètres est donc nettement moins importante dans ce cadre que pour GER-L.

Nous introduisons donc un à priori sur les paramètres, comme pour GER-P, et nous utilisons un critère de maximum a posteriori pour optimiser les paramètres. Pour ce faire, nous montrons tout d'abord que l'espérance du nDCG peut être réinterprétée comme la probabilité que l'utilisateur soit satisfait par un classement produit par le modèle d'un ensemble d'items I_u:

$$p(\mathbf{S}|u, \mathbf{I}_u) = \sum_r \sum_i p(\mathbf{S}|r, \mathbf{u}, \mathbf{i}, \mathbf{I}_u) p(\mathbf{R} = r|\mathbf{u}, \mathbf{i}, \mathbf{I}_u) P(\mathbf{i}|\mathbf{u}, \mathbf{I}_u) \tag{6}$$

où $\mathbf{S}$ est la satisfaction de l'utilisateur, et $\mathbf{I}_u$ la liste d'item à ordonner. En identifiant la probabilité qu'un utilisateur soit satisfait par un item i au rang r à la proportion du gain total apporté par l'item i au rang r, c.-à-d.

$$p(\mathbf{S}|r, \mathbf{u}, \mathbf{i}, \mathbf{I}_u) = \frac{g(i)d(r)}{CG_{\max}}$$

et en supposant que la probabilité d'observer un item i est uniforme, c.-à-d. $P(\mathbf{i}|\mathbf{u}, \mathbf{I}_u)$ est constant, nous pouvons alors utiliser une optimisation du maximum a posteriori :

$$\begin{aligned}
\Theta^* &= \operatorname*{argmax}_\Theta \frac{p\left(\mathcal{D}|\Theta\right) p(\Theta)}{p(\mathcal{D})} \\
&= \operatorname*{argmax}_\Theta \log p(\Theta) + \sum_{(u,\mathbf{I}_u)\in\mathcal{D}} \log p\left(\mathbf{S}|\mathbf{u}, \mathbf{I}_u, \Theta\right) \\
&= \log p(\Theta) + \sum_{(u,\mathbf{I}_u)\in\mathcal{D}} \frac{1}{CG_{max}(u, I_u)} \log \sum_{i\in I_u} g(i) \sum_{r\geq 1} d(r) p(\mathbf{R} = r|\mathbf{u}, \mathbf{i}, \mathbf{I}_u)
\end{aligned} \tag{7}$$

où $p(\Theta)$ est l'a priori Normal-Gamma utilisé pour GER-P.

Pour calculer la probabilité $p(\mathbf{R} = r|\mathbf{u}, \mathbf{i}, \mathbf{I}_u)$ que l'item i soit classé au rang r pour l'utilisateur u et la liste d'items I_u, nous utilisons la méthode basée sur la programmation dynamique (Taylor *et al.*, 2008) – qui consiste à ajouter les items un par un, c.-à-d. de calculer $P(i|u, I = i_1)$ puis $P(i|u, I = i_1, i_2)$, etc. Avec cet algorithme de calcul, il est seulement nécessaire de calculer la probabilité qu'un item soit préféré à un autre, ce qui est donné par l'équation 5. Il est possible de calculer la dérivée par rapport aux paramètres de la même façon.

3.3. Inférence : ordonner les items

Une fois les paramètres appris, le modèle peut être utilisé pour ordonner les items pour chaque utilisateur u. Contrairement aux autres modèles de recommandation, établir un ordre n'est pas direct car le score $\mathbf{S}_{ui} = X_i \cdot X_u + b_i$ est une variable aléatoire. Nous proposons plusieurs méthodes dans la suite.

La première (**D**irect) est d'utiliser le fait que la relation $i >_u j \iff p(i >_u j|\Theta) > 0.5$ définit une relation d'ordre total sur les items, car cela dépend seulement du signe de l'espérance de la différence des scores $p(i >_u j|\Theta) > 0.5 \iff \mu_u \cdot (\mu_i - \mu_j) + b_i - b_j > 0$. Un défaut de cette approche est que le classement des items ne prend pas en compte la variance des scores.

Pour éviter ce menu problème, on peut ordonner les items par leur probabilité d'avoir un score positif (**P**ositif), c.-à-d. $s_{ui} = p(\mathbf{X}_u \cdot \mathbf{X}_i + b_i > 0)$ [2]. L'intérêt de ce score est qu'il prend en compte la variance : en supposant un biais nul, si la variance est forte, alors cette probabilité sera proche de zéro; lorsque elle est faible, elle se rapproche de 1 si l'espérance est supérieure à zéro (et 0 sinon).

Le problème de cette manière d'ordonner les items est que le seuil 0 est arbitraire. Nous proposons donc d'utiliser la théorie de l'optimisation du portfolio (**PF**, **P**ort**F**olio) pour ordonner les items. L'utilisation de cette méthode a été proposée dans (Wang, Zhu, 2009) pour diversifier les résultats. Celle-ci a pour but de maximiser un critère qui dépend de l'espérance du gain et de la covariance des gains associés aux items. Dans notre modèle, le gain est le score $\mathbf{S}_u i$.

Pour ordonner les items, l'optimisation de portfolio correspond à l'optimisation d'un critère qui dépend d'un seul paramètre α qui contrôle le risque toléré en sélectionnant un ensemble d'items. Formellement, cela correspond à trouver le sous-ensemble d'items $L \subseteq I$ tel que, pour un utilisateur donné u,

$$\mathbb{E}\left[\sum_{i \in L} \mathbf{S}_{ui}\right] - \alpha V\left(\sum_{i \in L} \mathbf{S}_{ui}\right)$$

est maximisé. Avec $\alpha > 0$, le risque est réduit car les items avec une variance haute, ou dont le gain est très corrélé à d'autres items, ne seront pas sélectionnés. Quand $\alpha < 0$, c'est l'inverse : les items avec des gains potentiellement grands seront favorisés.

Ce critère étant trop complexe pour être résolu directement, un algorithme glouton est couramment utilisé (Wang, Zhu, 2009) : à chaque itération de cet algorithme, un item est ajouté à une liste. En notant L_n la liste obtenue après que n items aient été ajoutés, l'algorithme sélectionne alors l'item i qui maximise

$$\mathbb{E}(\mathbf{S}_{ui}) - \alpha V(\mathbf{S}_{ui}) - 2\alpha \sum_{j \in L_n} \operatorname{cov}(\mathbf{S}_{ui}, \mathbf{S}_{uj}) \tag{8}$$

Wang et al. (Wang, Zhu, 2009) estime la variance et covariance des scores des items à partir des données, ce qui pose trois problèmes : (1) lorsque peu de données sont disponibles, l'estimation est peu fiable; (2) numériquement, il est très couteux d'estimer la covariance d'un ensemble de paires d'items; (3) cette (co)variance ne dépend pas de l'utilisateur.

2. Nous utilisons une approximation normale et (Brown, Rutemiller, 1977) pour calculer la distribution issue du produit scalaire

La force d'une représentation gaussienne des utilisateurs et items est que nous pouvons utiliser directement celles-ci pour estimer variance et co-variance, car les scores $\mathbf{S}_{ui}$ sont des variables aléatoires. Pour la variance, nous utilisons directement le résultat de (Brown, Rutemiller, 1977):

$$V\left(\mathbf{S}_{ui}\right) = \mu_u^\top \Sigma_i \mu_u + \mu_i^\top \Sigma_u \mu_i + tr(\Sigma_u \Sigma_i) \tag{9}$$

Pour la covariance, nous exploitons l'indépendance entre les représentations des items/utilisateurs (étant donné leurs moyennes et matrices de covariance, c.-à-d. les paramètres du modèle) : X_u, X_i an X_j sont indépendants. Cela donne :

$$\begin{aligned}
\operatorname{cov}(\mathbf{S}_{ui}, \mathbf{S}_{uj}) &= \mathbb{E}\left[\mathbf{X}_u^\top \mathbf{X}_i \mathbf{X}_u^\top \mathbf{X}_j\right] - \mathbb{E}\left[\mathbf{X}_u^\top \mathbf{X}_i\right] \mathbb{E}\left[\mathbf{X}_u^\top \mathbf{X}_j\right] \\
&= \sum_{p=1}^{d} \sum_{q=1}^{d} \left(\mathbb{E}\left[\mathbf{X}_{up}\mathbf{X}_{uq}\right] - \mathbb{E}\left[\mathbf{X}_{up}\right]\mathbb{E}\left[\mathbf{X}_{uq}\right]\right) \mathbb{E}\left[\mathbf{X}_{ip}\right]\mathbb{E}\left[\mathbf{X}_{jq}\right] \\
&= \mathbb{E}\left[\mathbf{X}_i\right]^\top \operatorname{cov}(\mathbf{X}_u)\mathbb{E}\left[\mathbf{X}_j\right] = \mu_i^\top \Sigma_u \mu_j
\end{aligned} \tag{10}$$

En combinant les équations (9) et (10), on peut calculer l'item i qui maximise l'équation (8) et constituer de cette manière la liste à renvoyer aux utilisateurs.

4. Expériences

4.1. Expériences avec GER-P

Nous avons testé notre modèle sur trois différents jeux de données extraits respectivement du *Yahoo! Music ratings for User Selected and Randomly Selected songs, version 1.0* [3], de MovieLens 100k [4] et de Yelp Dataset Challenge [5]. Le jeu de données **Yahoo!** contient des évaluations d'utilisateurs sur des chansons, pour 15,400 utilisateurs et 1,000 chansons, avec environ 350,000 évaluations. Le jeu de données **MovieLens** contient des évaluations sur des films. Les données concernent 943 utilisateurs et 1,682 films pour environ 100,000 évaluations. Le jeu de données **Yelp** contient des évaluations sur des commerces, pour 1,029,432 utilisateurs et 144,072 commerces, pour un total de 4,1 million évaluations.

Nous avons testé les modèles suivants, qui sont tous basés sur un critère d'apprentissage *Learning-to-Rank* (à l'exception de MP): **(MP)** *Most Popular* est un classement où les items sont ordonnés en fonction du nombre de fois où ils ont été noté dans le passé – bien que très rudimentaire, ce modèle est tout à fait compétitif. Cette procédure ne nécessite pas d'optimisation d'hyperparamètres; **(SM)** Fonction objectif

3. Disponible via le *Yahoo! Webscope data sharing program* http://webscope.sandbox.yahoo.com/.
4. http://grouplens.org/datasets/movielens/100k/.
5. https://www.yelp.com/dataset_challenge

Tableau 1. Statistiques des jeux de données en fonction du nombre d'exemples d'apprentissage (10, 20 ou 50 évaluations par utilisateur).

	Utilisateurs			items			Évaluations ($\times$ 1000)		
	10	20	50	10	20	50	10	20	50
Yahoo!	3386	1645	286	1000	1000	995	159	100	33
MovieLens	743	643	448	1336	1330	1307	95	91	81
Yelp	13775	8828	3388	44877	39355	27878	980	791	467

d'ordonnancement *Soft margin (hinge)*, en utilisant la descente de gradient stochastique (Weimer *et al*., 2008) et l'implémentation de (Gantner *et al*., 2011), en optimisant tout les hyperparamètres possibles; **(BPRMF)** *Bayesian Personalized Ranking* (Rendle *et al*., 2009), sur lequel notre fonction de coût est basée, en utilisant l'implémentation de (Gantner *et al*., 2011) en optimisant tout les hyperparamètres possibles; **(CofiRank)** Un algorithme d'ordonnancement avec des performances de l'état de l'art pour les systèmes de recommandation (Weimer *et al*., 2007), qui optimise une borne supérieure de la nDCG [6]. Nous avons utilisé les hyperparamètres utilisés dans (Weimer *et al*., 2007); **GER-P** Notre modèle, nommé *Gaussien Embeddings Ranking*, en utilisant l'apprentissage par ordonnancement et l'ordonnancement donné par $p(\mathbf{S}_{ui} > 0)$. Le nombre de dimensions pour l'espace latent de représentation est de 5, 10, 20, 50, 100, 200, 500 et 1000.

Nous nous servons principalement de la procédure expérimentale explicitée par (Weimer *et al*., 2007). Les jeux de données ont été pré-traités comme suit: Pour chaque jeu de données, et chaque utilisateur, un nombre fixe (10, 20, ou 50) d'items est mis de côté pour l'entrainement du modèle, 10 autres sont utilisés pour la validation, et le reste est attribué au jeu de test. Les utilisateurs avec moins de 30, 40, 70 évaluations ont été enlevés, de façon à ce qu'on ait toujours au moins 10 évaluations pour la phase de test. Nous avons aussi éliminé les items qui n'étaient pas notés par au moins 5 utilisateurs. Les statistiques sur les bases d'entrainement sont données dans la table 1. La base de validation est utilisée pour sélectionner les meilleurs hyper-paramètres pour GER-P [7], SM et BPRMF. Finalement, pour l'évaluation, nous avons ré-ordonné les items jugés en utilisant le modèle évalué, et nous avons utilisé la métrique nDCG aux rangs 1, 5 et 10. Les résultats rapportés ici sont une moyenne sur 5 expériences différentes, avec 5 divisions entrainement/validation/test différentes.

Les résultats sont donnés dans la Table 2. Il y a approximativement trois types de modèles: (i) Soft Margin (SM) qui fonctionne le moins bien (ii) BPRMF et *Most Popular* (MP) (iii) GER et CofiRank (CR) . *Most Popular* fonctionne plutôt bien en partie parce que nos expériences portent uniquement sur les items *vus* par les utilisateurs.

6. https://github.com/markusweimer/cofirank

7. Nous avons observé qu'une dimension de 50 donne en règle générale des bons résultats indépendamment du dataset, et c (régularisation du biais) $\approx 10^{-5}$ donne des bons résultats sur Movielens et Yelp, mais nécessite d'être plus élevé ($\approx 10^{-3}$) sur le jeu de données Yahoo pour obtenir des résultats satisfaisants.

Tableau 2. Résultats du classment collaboratif. Les valeurs de la nDCG (×100) à différents niveau de troncation sont montrées dans les colonnes principales, qui sont divisées selon le nombre d'items dans l'ensemble d'apprentissage.

Train Size →	10			20			50		
Model ↓	N@1	N@5	N@10	N@1	N@5	N@10	N@1	N@5	N@10
Yahoo!									
MP	53.0	59.1	67.3	52.5	58.3	66.4	53.6	57.8	64.0
BPRMF	52.8	59.0	67.2	52.2	58.3	66.4	52.2	57.7	63.5
SM	50.9	56.7	65.4	49.7	55.6	64.2	49.9	54.1	60.3
CR	**53.5**	**60.3**	68.2	**57.8**	**61.7**	**68.9**	**56.0**	**60.0**	**65.6**
GER	**53.5**	**60.3**	**68.3**	53.8	60.7	68.2	54.3	59.6	65.3
MovieLens									
MP	66.0	64.7	65.8	68.4	65.3	66.3	69.1	67.4	67.5
BPRMF	66.1	64.6	65.7	66.3	64.3	65.8	66.9	65.0	66.2
SM	55.9	57.5	60.3	58.3	59.6	61.6	58.6	60.4	62.5
CR	69.0	67.3	68.6	69.7	68.5	69.5	71.4	69.4	70.6
GER	**70.3**	**67.7**	**70.0**	**72.0**	**69.5**	**71.1**	**72.5**	**71.3**	**71.5**
Yelp									
MP	40.7	41.5	46.9	39.5	39.9	44.7	37.4	37.6	41.4
BPRMF	40.8	41.3	46.8	39.6	39.8	44.6	37.3	37.2	40.9
SM	37.3	38.3	44.4	35.8	36.9	41.9	33.4	34.1	38.0
CR	47.1	46.9	51.1	46.5	46.6	50.4	46.2	45.8	48.6
GER	**55.2**	**52.2**	**56.2**	**57.4**	**53.5**	**56.4**	**58.2**	**53.7**	**55.3**

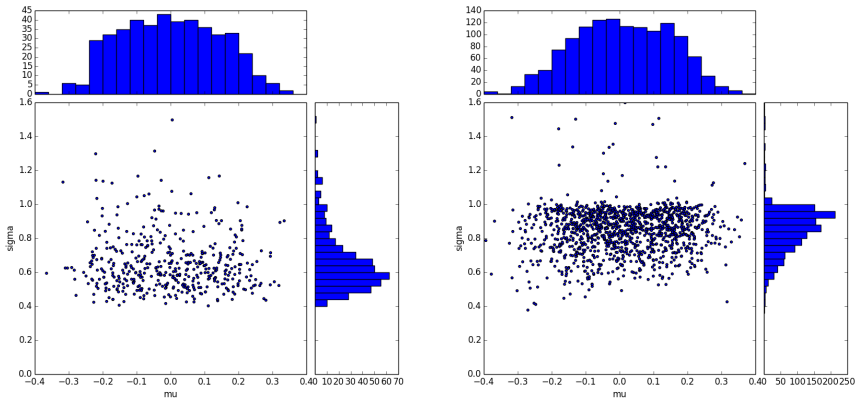

FIGURE 2. *Les premiers composants des représentations des utilisateurs (à gauche) et des items (à droite), dimension 50 et 50.*

Notre modèle (GER-P) surclasse les autres modèles sur les jeux de données Movielens et Yelp. Il est, en général, plus performant que CofiRank (avec des différences de nDCG oscillant entre 0.01 et 0.08), exception faite sur le jeu de données Yahoo avec un ensemble d'entrainement de 20 et 50 (avec des différences sur la nDCG allant de 0,001 à 0,02). De manière générale, GER fonctionne très bien sur les trois jeux de données avec des caractéristiques différentes en termes de comportement d'ordonnancement et du nombre d'évaluation.

4.2. Analyse

Dans cette section, nous analysons les résultats, et étudions les représentations latentes apprises, et en particulier l'utilisation de la variance, pour le jeu de données Yahoo. Nous avons utilisé les hyper-paramètres qui ont été sélectionnés sur le jeu de validation, puis observé l'ensemble des couples moyenne-variance, c.-à-d. $(\mu_{\bullet k}, \Sigma_{\bullet k})$ pour $k \in \{1, \ldots, N\}$ et $\bullet \in \mathcal{U} \cup \mathcal{I}$.

Dans la figure 2, nous avons représenté différents couples $(\mu_{\bullet k}, \Sigma_{\bullet k})$, ainsi que des histogrammes des moyennes et variances. Le modèle exploite, pour les moyennes, l'intervalle $[-0.3, 0.3]$. De manière plus importante, les variances se regroupent dans l'intervalle $[0.4, 1.2]$, autour de l'apriori. Il n'était pas tout à fait évident que la variance allait être exploitée correctement par le modèle, ces résultats sont donc positifs.

Pour mieux comprendre l'utilisation de la variance, dans la Figure 3 nous avons tracé un graphique "en violon" (densité) de la variance des utilisateurs et des items, en fonction de la dimension de l'espace latent (5 à 1000), et en fonction de la taille du jeu d'entrainement (10 à 50). On peut observer que, quand la quantité d'exemples en entrainement augmente, la médiane des $\sigma_{\bullet k}$ décroit alors que la variance des $\sigma_{\bullet k}$ croît. Cela rentre en adéquation avec nos hypothèses : la décroissance de la médiane montre que plus d'information réduit l'incertitude sur les représentations, alors que la croissance de la variance montre que l'augmentation du nombre d'exemples d'entrainement augmente l'incertitude pour certains éléments (donc leur variance) – ceci est surtout confirmé dans le cas des items qui reçoivent beaucoup d'évaluations. Finalement, nous pouvons observer que quand la dimension de l'espace latent est trop élevée, la plus grande partie de la densité des $\sigma_{\bullet k}$ est centrée autour de l'à priori – ce qui pourrait impliquer que la plupart des dimensions ne sont pas utilisées.

4.3. Expériences avec GER-L et PortFolio

Dans cette section, nous présentons des résultats préliminaires obtenus pour évaluer GER-L et les différentes formes d'ordonnancement. Nous avons utilisé des jeux de données plus importants et contenant une information sur le genre (pour les mesures de diversification). Nous avons ainsi pris comme base expérimentale la taille du jeu de donnée *Movielens 1M*[8] qui contient 1 million de notes. Le prétraitement

8. Disponible ici : https://grouplens.org/datasets/movielens/1m/.

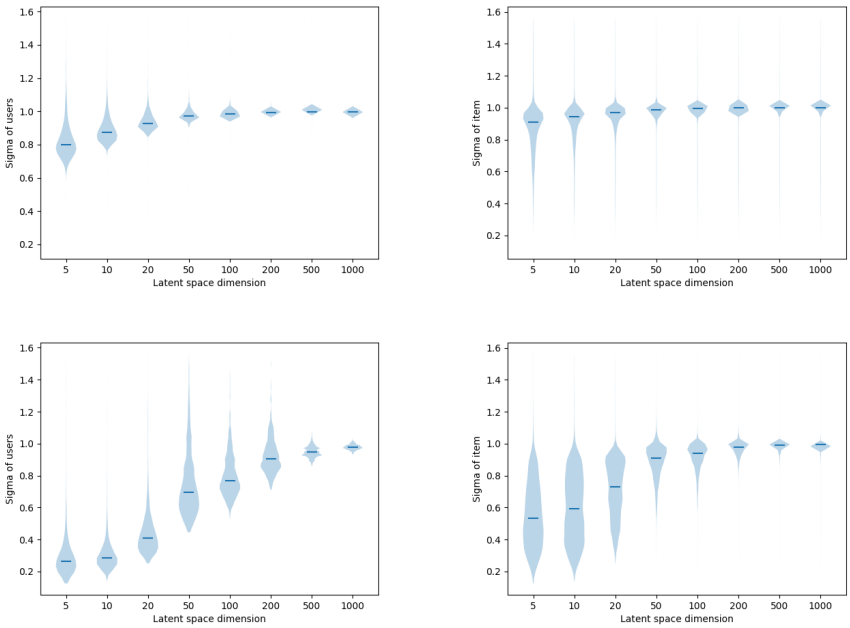

FIGURE *3. Graphique en violon (densité) pour la variance des utilisateurs (à gauche) et des items (à droite) pour la dimension de l'espace latent T10 (en haut) et T50 (en bas).*

sur ces jeux de données est identique aux expériences sur GER-P. Dans cette série d'expériences, nous évaluons GER-P et GER-L, ainsi que les différentes façons d'ordonner les items présentées dans la section 3.3. L'évaluation se fait ainsi toujours sur l'ensemble d'évaluation du jeu de données (10 items par utilisateur).

Nous avons testé ici 5 types de modèles différents, avec les deux critères d'apprentissage basés sur BPR et Softrank: (i) BPRMF ; (ii) GER-P ; (iii) GER-L (notre modèle, avec un critère basé sur SoftRank) ; (iv) BPR-L, que l'on optimise en utilisant le critère Softrank, mais en utilisant l'équation ((3)) pour estimer la préférence ; (v) SOFTRANK-L, où les représentations sont encore basées sur des points déterministes, mais la préférence est ici calculée selon la formule originelle donnée par (Taylor *et al.*, 2008) : $p\left(i >_u j|\Theta\right) = \int_0^\infty \mathcal{N}\left(s; x_i \cdot x_u + b_i - x_j \cdot x_u - b_j, 2\sigma_s^2\right) ds$ où σ est un hyperparamètre .

Au niveau de l'ordonnancement en inférence, nous avons utilisé les ordonnancements suivants : **D** Basé sur le score s_{ui} associé à la probabilité $p(\mathbf{X}_u \cdot \mathbf{X}_i + b_i > 0)$ qui prend en compte la variance ; **P** Basé sur le score s_{ui} associé à la relation $i >_u j \iff p\left(i >_u j|\Theta\right) > 0.5$ qui ne prend donc pas en compte la variance ; **PF** Basé

sur l'algorithme d'optimisation de Portfolio (Wang, Zhu, 2009). Pour les représentations déterministes nous avons expérimenté avec les deux ordonnancements suivants: (i) Basé sur le score $s_u i = x_i \cdot x_u + b_i$; (ii) Basé sur l'algorithme de portfolio, en évaluant la covariance et la variance à partir des données présentes dans le jeu d'entrainement. .

Tableau 3. Résultats du classment collaboratif. Les valeurs de la nDCG ($\times 100$) à différents niveau de troncation sont montrées dans les colonnes principales, qui sont divisées selon le nombre d'items dans l'ensemble d'apprentissage.

Train Size →	10			20			50		
Model ↓	N@1	N@5	N@10	N@1	N@5	N@10	N@1	N@5	N@10
Movielens 1M									
BPRMF	56.9	54.0	54.9	57.5	54.2	54.6	55.9	53.1	53.2
BPR-L	65.5	59.7	56.8	66.0	60.5	57.0	65.9	59.4	55.6
SOFTRANK-L	64.5	59.2	57.5	66.0	60.2	56.9	65.3	59.3	55.6
GER-P	**68.5**	**62.3**	**58.1**	67.3	61.4	56.9	68.0	61.1	56.2
GER-L	67.2	61.5	57.8	**67.3**	**61.7**	**57.3**	**68.4**	**61.7**	**56.5**

Nous présentons les résultats des expériences préliminaires dans la table 3 pour l'ordonnancement P (les autres ayant donné des résultats similaires). La première constatation est que les modèles GER ont de meilleurs résultats que BPR et SoftRank; la seconde est que le modèle basé sur l'apprentissage de liste (GER-L) a de meilleures performances seulement lorsque il y a suffisament de données pour apprendre – et la différence reste faible. Nous conduisons actuellement des expériences sur d'autres jeux de données et en variant plus les hyper-paramètres des différents modèles afin de quantifier leurs différences.

5. Conclusion

Dans cet article, nous avons proposé un système de recommandation collaboratif où les utilisateurs et les items correspondent à des distributions dans un espace latent, et non à des points déterministes comme dans d'autres modèles basés sur des représentations latentes. Cela permet de gérer avec l'incertitude inhérente au manque d'information, ou à des informations conflictuelles. Nous montrons que notre modèle surclasse d'autres modèles de l'état de l'art, et présentons des pistes à explorer au niveau des modalités d'apprentissage et d'inférence. En outre, ce type de modèle a divers avantages que nous n'avons pas encore exploités: (i) l'intégration de sources externes et variées d'information, en exploitant potentiellement l'incertitude de chaque source (ii) l'utilisation de la variance pour gérer la composante temporelle (la représentation devient de plus en plus incertaine avec le temps si aucune nouvelle information n'est donnée)

Remerciements

Ces travaux ont été partiellement financés par: FUI PULSAR (BPI France, Région Ile de France) et LOCUST (ANR-15-CE23-0027-01).

Bibliographie

Adomavicius G., Tuzhilin A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, n° 6.

Brown G. G., Rutemiller H. C. (1977, October). Means and Variances of Stochastic Vector Products with Applications to Random Linear Models. *Management Science*, vol. 24, n° 2.

Dos Santos L., Piwowarski B., Gallinari P. (2016a). Multilabel Classification on Heterogeneous Graphs with Gaussian Embeddings. In *ECML*. Springer International Publishing.

Dos Santos L., Piwowarski B., Gallinari P. (2016b). Multilabel Classification on Heterogeneous Graphs with Gaussian Embeddings. In *ECML*.

Gantner Z., Rendle S., Freudenthaler C., Schmidt-Thieme L. (2011). MyMediaLite: A free recommender system library. In *Recsys*.

He S., Liu K., Ji G., Zhao J. (2015). Learning to Represent Knowledge Graphs with Gaussian Embedding. In *CIKM*.

Järvelin K., Kekäläinen J. (2002, octobre). Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.*, vol. 20, n° 4.

Karatzoglou A., Baltrunas L., Shi Y. (2013). *Learning to rank for recommender systems*.

Liu T.-Y. (2011). *Learning to Rank for Information Retrieval*. Springer.

Rendle S., Freudenthaler C., Gantner Z., Schmidt-Thieme L. (2009, June). BPR: Bayesian personalized ranking from implicit feedback. In *Uncertainty in artificial intelligence*. AUAI Press.

Ricci F., Rokach L., Shapira B., Paul B. K. (2011). *Recommender Systems Handbook* (vol. 532).

Salakhutdinov R., Mnih A. (2007). Probabilistic Matrix Factorization. In *Proceedings of advances in neural information processing systems*, vol. 20.

Shi Y., Karatzoglou A., Baltrunas L., Larson M., Oliver N., Hanjalic A. (2012). CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-more Filtering. In *RecSys*.

Taylor M., Guiver J., Robertson S., Minka T. (2008). SoftRank: Optimizing non-smooth rank metrics. In *WSDM*.

Vilnis L., McCallum A. (2014). Word Representations via Gaussian Embedding. In *ICLR*.

Wang J., Zhu J. (2009). Portfolio theory of information retrieval. In *SIGIR*. ACM Press.

Weimer M., Karatzoglou A., Le Q. V., Smola A. (2007). CofiRank Maximum Margin Matrix Factorization for Collaborative Ranking. In *NIPS*.

Weimer M., Karatzoglou A., Smola A. (2008). Improving maximum margin matrix factorization. *Machine Learning*, vol. 72, n° 3.

Zhang F., Yuan N., Lian D., Xie X., Ma W.-Y. (2016). Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*.