Rapport Projet Sciences des données : Model Based collaborative filtering : Matrix Factorization

Nassim Ait Ali Braham, Jean Dupin, Vincent Gouteux

November 2019

1 Présentation du Problème

On s'intéresse dans ce projet à l'approche "collaborative filtering" appliquée à la recommandation de films. Plus précisément, nous explorerons les méthodes basées sur les facteurs latents [1]. Le principe général est de représenter les notes dans une matrice $Utilisateurs * Films$. Nous pouvons par la suite exploiter les corrélations entre les notes afin d'exprimer cette matrice comme le produit de deux matrices $U \in \mathbb{R}^{n \times k}$ et $V \in \mathbb{R}^{m \times k}$ de rang inférieur. Les matrices ainsi obtenues sont constituées de vecteurs que l'on appelle les facteurs latents.

Nous utiliserons les jeux de données MovieLens fournis par GroupLens : MovieLens Dataset. Le groupe de recherche GroupLens a rassemblé des évaluations de films auprès de volontaires depuis 1995 et a produit des jeux de données de différentes tailles. Nous nous concentrerons sur le jeu de données le plus petit (environ 100,000 notes)

On abordera ce problème par le biais d'une approche *model-based* où il s'agira de chercher une approximation de de la matrice $(utilisateurs, films)$ par le produit de deux matrices de faible rang U et V de la forme suivante $R = U \cdot V^T$ Le but va donc être de minimiser la fonction de coût $C(U, V)$ qui se compose de la norme de Frobenius de la différence de la matrice R et du produit des matrices U et V, à laquelle s'ajoutent des termes de régularisation. On a donc :

$$C(U, V) = ||R - UV^T||_{\mathcal{F}}^2 + \lambda ||U||_{\mathcal{F}}^2 + \mu ||V||_{\mathcal{F}}^2$$

Avec $U \in \mathbb{R}^{n,k}$, $V \in \mathbb{R}^{m,k}$, $||X||_{\mathcal{F}} = tr(X^T X)$

Afin de minimiser cette fonction, nous aurons recours à plusieurs algorithmes d'optimisation, notamment :

- La descente de gradient stochastique (Stochastic Gradient Descent)
- Les moindres carrés alternés (Alternating Least Squares)
- La descente par coordonnées (Coordinate Descent)

Nous comparerons les performances de ces algorithmes, leur sensibilité au paramétrage ainsi que la qualité des solutions trouvées.

2 Les données

Pour des raisons de temps de calcul, nous avons choisi le petit dataset qui contient environ 100 000 notes. Il y a 610 utilisateurs et 9742 films ce qui fait donc plus de 5 millions de notes a prédire. Nous avons donc stocké ces notes dans une matrice $R \in \mathcal{M}_{610*9742}(\mathbb{R})$.

Pour tester la validité et les performances de nos différents algorithmes, nous avons implémenté des fonctions nous permettant de diviser le dataset en 3 jeux de données : *Training Set, Validation Set, Testing Set*. Les utilisateurs n'ayant pas tous notés le même nombre de films, nous avons déterminé qu'il était plus judicieux de prendre une fraction des notes de chaque utilisateur plutôt qu'un sous-ensemble aléatoire de toutes les notes. Cette méthode nous permet d'entraîner notre modèle sur tous les utilisateurs et donc d'apprendre sur tout le monde, évitant ainsi le problème du *Cold Start* : passer toutes les notes d'un utilisateur dans le *Validation set ou Testing Set*.

3 Méthodes implémentées

Dans cette section, nous donnons un aperçu du principe général de chaque méthode d'optimisation implémentée. Nous discuterons également certaines modifications apportées au modèle pour de améliorer les performances.

3.1 Descente de Gradient Stochastique (SGD)

La descente de gradient est un algorithme d'optimisation itératif glouton extrêmement utilisé pour la minimisation de fonctions à valeurs réelles. Son principe est le suivant : à chaque itération, on prend la direction opposée du gradient de la fonction C à optimiser, jusqu'à convergence.

On peut aisément montrer que les dérivées partielles de notre fonction coût $C(U, V)$ sont données par les formules suivantes :

$$\frac{\partial C}{\partial u_{iq}}(U,V) = 2 \cdot \sum_{j,r_{ij}\ observed} (r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js})(-v_{jq}) + 2\lambda u_{iq}$$

$$\frac{\partial C}{\partial v_{jq}}(U,V) = 2 \cdot \sum_{i,r_{ij}\ observed} (r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js})(-u_{iq}) + 2\mu v_{jq}$$

Il est possible d'effectuer une descente de gradient classique sur les matrices U et V. Néanmoins, le calcul d'un gradient étant bien trop coûteux, nous avons opté pour la variante stochastique (SGD) qui minimise l'erreur point par point. La descente de gradient stochastique (SGD) calcule une prédiction p_{ij} et son erreur associée $e_{ij} = p_{pi} - r_{ij}$. On calcule ensuite le gradient par rapport à ce terme d'erreur, pour mettre à jour les paramètres dans la direction opposée du gradient.

$$u_i = u_i + \alpha(e_{ij} \cdot v_j - \cdot u_i)$$

$$v_j = v_j + \alpha(e_{ij} \cdot u_i - \cdot v_j)$$

Où le terme α représente le *Learning Rate*.

3.2 Moindres Carrés Alternés (ALS)

L'algorithme des moindres carrés alternés exploite la convexité de la fonction $C(U, V)$ en U (resp. V) pour V (resp. U) fixée. Le principe est le suivant :

1. On fixe U et on choisit V de sorte à minimiser $C(U, V)$ en résolvant $\frac{\partial C}{\partial V}(U, V) = 0$
2. On fixe V et on choisit U de sorte à minimiser $C(U, V)$ en résolvant $\frac{\partial C}{\partial U}(U, V) = 0$

Ce processus est itéré jusqu'à convergence. On remarquera que le coût d'une itération de cet algorithme est bien plus élevé que pour SGD. Néanmoins, la section expérimentale montrera que la convergence est atteinte en un faible nombre d'itérations.

3.3 Descente par coordonnées (CD)

L'algorithme de *Coordinate Descent (CD)* repose sur un principe très similaire à celui de ALS. Il consiste à optimiser chaque variable de la fonction coût en fixant toutes les autres. De ce fait, ALS peut être vu comme une variante "en bloc" de CD.

3.4 Biais des utilisateurs et films

Le modèle précédemment décrit peut naturellement être enrichi avec des termes de biais pour plus d'expressivité. On pourra alors considérer :

- Un biais global μ : un intercepte qui représente la moyenne des notes observées
- Des biais bu_i : un vecteur de biais capturant la tendance générale des notes attribuées par chaque utilisateur (ex : utilisateur généreux)
- Des biais bm_j : un vecteur de biais capturant la tendance générale des notes attribuées à chaque film (ex : film globalement bien noté)

On se retrouve donc avec de nouvelles prédictions et une nouvelle fonction coût à minimiser :

$$p_{ij} =< u_i, v_j > + bu_i + bm_j$$

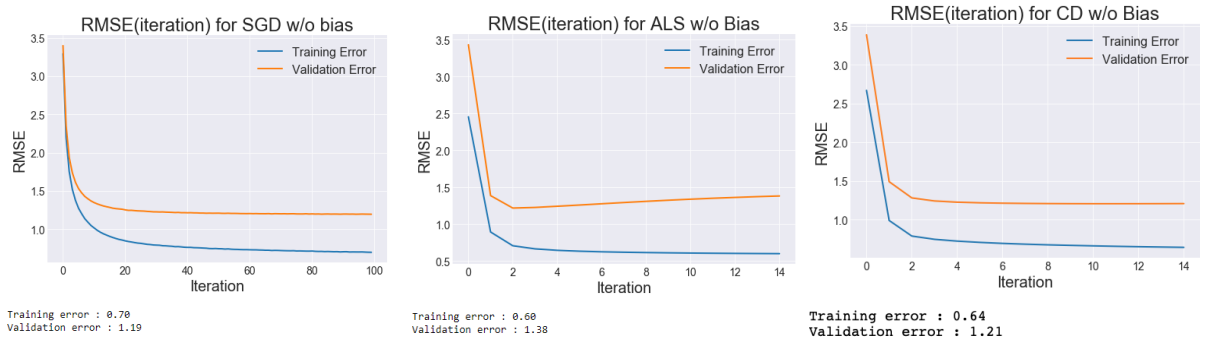
$$(u_i, v_j) = \min_{u,v} \sum_{j, r_{ij} observed} (p_{ij} - bu_i - bm_j - r_{ij})^2 + \lambda(||u_i||^2 + bu_i^2) + \mu(||v_j||^2 + bm_j^2)$$

On peut aisément calculer les dérivées partielles de C par rapport aux variables de biais, et les intégrer dans les algorithmes d'optimisation. Une autre façon de procéder consiste à artificiellement étendre U et V avec des colonnes de 1 et de variables de biais. De cette façon, les biais peuvent être traités comme des variables régulières. Cette astuce est analogue au traitement des interceptes dans les modèles linéaires et réseaux de neurones par exemple.

4 Résultats expérimentaux

4.1 Résultats préliminaires

Dans cette sous-section, nous donnons les premiers résultats obtenus pour les méthodes implémentées. Il s'agit donc de s'assurer du fonctionnement correct des algorithmes, ainsi que leur comportement global.



(a) SGD (b) ALS (c) CD

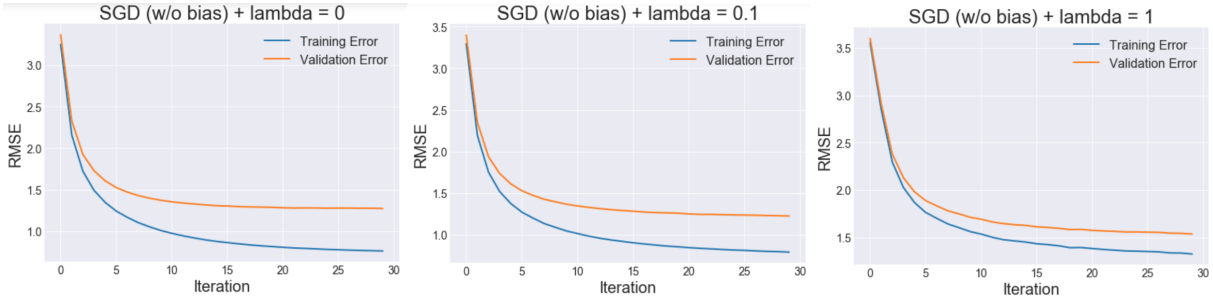
Figure 1: Convergence de ALS et SGD

La Figure 1 illustre l'évolution de l'erreur d'entraînement et de validation pendant la phase d'apprentissage. On remarque une convergence relativement rapide pour SGD (environ 20 itérations), avec une erreur de validation qui se stabilise à 1.19 et une erreur d'entraînement de 0.70. ALS donne des résultats assez similaires, mais converge en un nombre bien plus faible d'itérations. On remarquera par ailleurs un phénomène *d'overfitting* à partir de quelques itérations, avec une erreur d'entraînement qui baisse et une erreur de validation qui augmente.

Enfin, Coordinate Descent donne des résultats sensiblement proches aux deux autres algorithmes. À noter que ces graphiques ont été générés en utilisant les paramètres suivants : $\lambda = \mu = 0.1$, $\alpha = 0.01$ et $k = 5$.

4.2 Influence de la régularisation

Dans cette section, nous exposons les résultats d'une série d'expérience dont l'objectif était d'analyser l'effet des paramètres de régularisation sur la fonction coût et la solution obtenue. Le but étant de vérifier expérimentalement des faits établis, et évaluer les paramètres de régularisation les plus adéquats à notre problème. Pour simplifier, nous posons $\lambda = \mu$



(a) $\lambda = 0$ (b) $\lambda = 0.1$ (c) $\lambda = 1$

Figure 2: Influence du paramètre de régularisation sur la convergence des algorithmes

La figure 2 illustre l'influence d'un paramètre de régularisation croissant sur l'allure des erreurs d'entraînement et validation durant l'apprentissage. On constate sans surprise qu'augmenter λ réduit l'écart entre l'erreur d'entraînement et celle de la validation (moins *d'overfitting*). Par contre, pour des valeurs trop grandes de λ, on tombe dans le phénomène inverse *d'underfitting*. C'est à dire que les performances du modèle sur les données d'entrainement et validations sont très proches, mais globalement plus mauvaises.

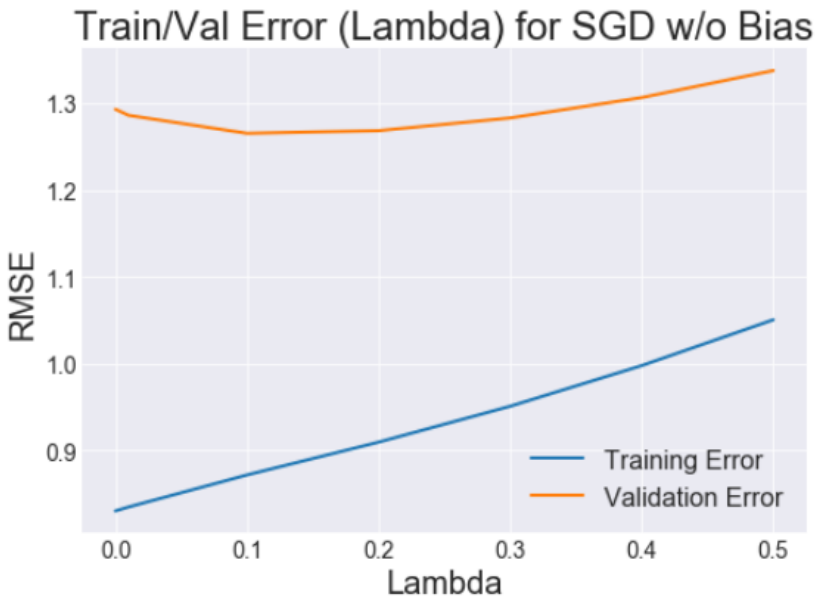


Figure 3: Évolution de l'erreur d'entrainement et validation en fonction de λ

Il est donc nécessaire de trouver un compromis entre underfitting et overfitting. Pour ce faire, on peut tracer l'évolution des erreurs pour différentes valeurs de λ, comme le montre la figure 3. La courbe montre une erreur d'entraînement croissante, ce qui est totalement attendu puisque l'on impose des contraintes de plus en plus fortes sur l'espace des solutions. D'un autre côté, on constate une légère forme en U pour la courbe de l'erreur de validation. Ceci suggère que les valeurs optimales se situent quelque part dans l'intervalle $[0.1, 0.2]$.

4.3 Influence de l'initialisation

L'initialisation des coefficients des matrices U et V a une grande influence sur les résultats et la performance du modèle. L'initialisation qui nous a paru la plus naturelle est de générer les coefficients en faisait des tirages de loi normale $\mathcal{N}(\mu, \sigma^2)$. Nous avons donc dû effectuer divers tests sur les paramètres de ces loi pour trouver la moyenne et l'écart-type qui minimisaient notre erreur de validation.

Nous avons remarqué à travers un certain nombre d'exécutions (fonction *best_initialization*()) que peu importe la moyenne, l'erreur de validation augmente systématiquement avec la variance. Ce résultat est illustré par la Figure 4. Il est donc préférable de prendre une variance faible. Pour ce qui est de la moyenne, après plusieurs tests, il semblerait que prendre une moyenne $\mu = 1$ nous donne une erreur de validation minimale.

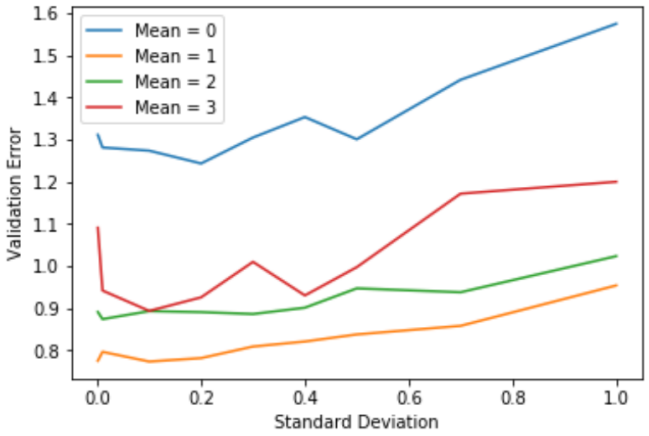
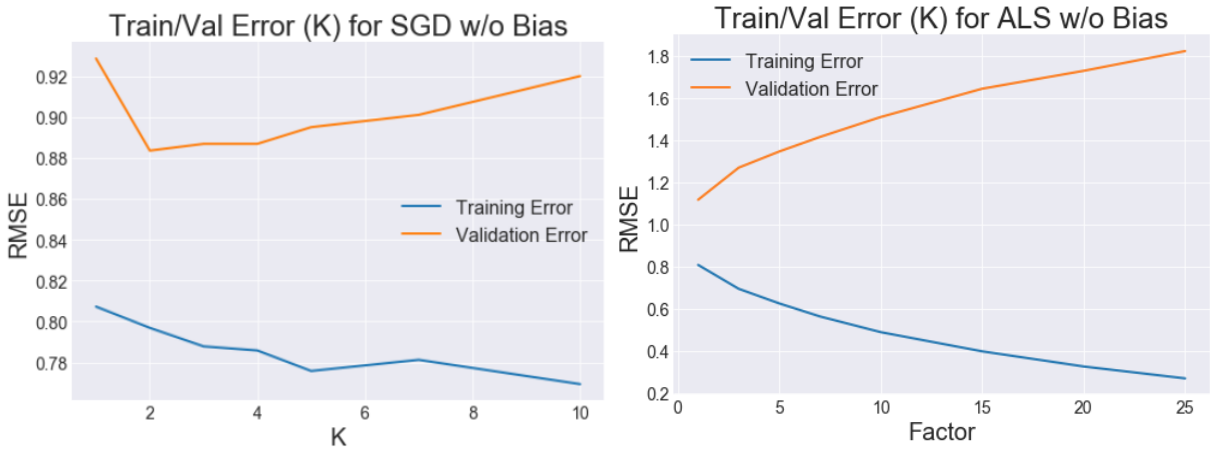


Figure 4: Erreur de validation obtenue pour diverses initialisations des matrices

4.4 Sélection du nombre de facteurs

Le nombre de facteurs latents k est sans doute l'hyper-paramètre le plus important du modèle. En plus d'influencer les performances, il peut potentiellement permettre une meilleure compréhension des données.



(a) SGD (b) ALS

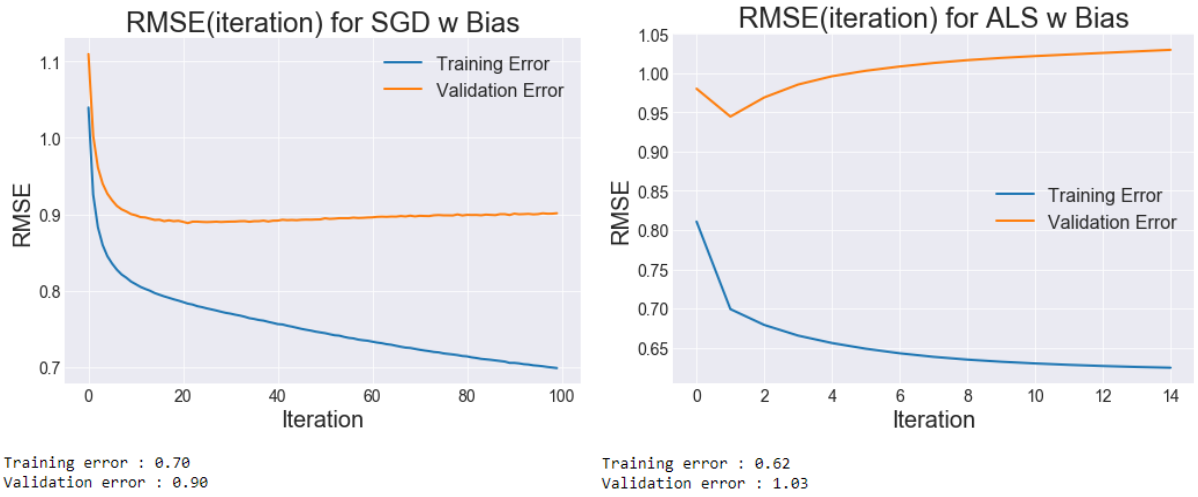
Figure 5: Évolution des erreurs en fonction de K pour SGD et ALS

Afin de déterminer la valeur optimale du paramètre k, nous avons réalisé un ensemble de tests sur ALS et SGD, en faisant varier k sur un ensemble de valeurs candidates. Les graphiques obtenus sont donnés dans la Figure 5.

Les résultats obtenus sont sont plutôt surprenants. En effet, on constate d'abord qu'un nombre très faible de facteurs suffit à avoir des performances décentes. Plus encore, à partir de 3 facteurs, le modèle commence à overfitter. Autrement dit, on obtient les meilleures performances pour un petit nombre de facteurs (typiquement k = 1, 2). Les phénomène est encore plus accentué pour ALS, où ne constate aucune baisse de l'erreur de validation en augmentant les facteurs.

4.5 Ajout du biais

Les paramètres de biais devraient intuitivement permettre de capturer une partie de la régularité dans les données, permettant ainsi de mieux apprendre les spécificités de chaque utilisateur/film. On s'attend ainsi à obtenir de meilleurs résultats une fois les biais inclus.



(a) SGD avec biais

(b) ALS avec bias

Figure 6: Résultats obtenus en incluant des termes de bias

La Figure 6 confirme bien nos attentes. Pour les mêmes paramètres que ceux utilisés en section 4.1 (résultats préliminéaires), on obtient des erreurs de validation plus faibles. On donc peut voir à travers ces résultats que les biais ont un rôle assez important dans les performances des différents modèles.

Les biais expliquent même a eux seuls une grande partie des données et peuvent représenter une prédiction assez fiable malgré leur simplicité. En effet, pour voir le réel impact des biais, nous avons exécuté ALS et SGD sans facteurs latents. Les prédictions étaient alors uniquement constituées des biais. Les résultats obtenus se sont avérés être étonnement convaincants, comme le montre le tableau 1.

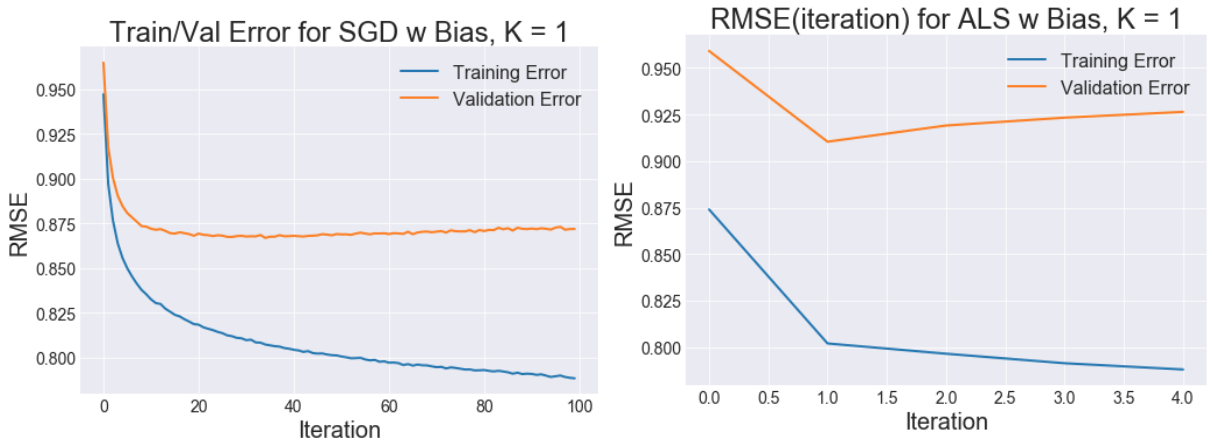
K=0	SGD	ALS
Training Error	0.80	0.83
Validation Error	0.87	0.93

Table 1: Erreur d'entraînement et de validations obtenues

Enfin, nous avons également constaté que l'ajout des biais minimisait l'impact de l'initialisation sur la solution finale. En exécutant les mêmes expériences qu'en section 4.3, nous avons observé une forte baisse de la sensibilité dans la solution obtenue.

4.6 Comparaison des méthodes

Globalement, nous avons (étonnement) obtenu les meilleures performances en utilisant SGD, qui nous a permis d'atteindre une RMSE d'environ 0.87, en utilisant les paramètres "optimaux". Quant à ALS, la solution optimale obtenue possède une RMSE d'environ 0.91. On notera par ailleurs le phénomène récurrent d'overfitting dans ALS au bout de quelques itérations.



(a) SGD (b) ALS

Figure 7: Erreur de validation et d'entraînement pour SGD et ALS avec les "meilleurs" paramètres

Enfin, l'erreur obtenue avec coordinate descent était légèrement supérieures à celle d'ALS (de l'ordre de 0.95). Néanmoins, la méthode a été moins testées que SGD et ALS faute de temps. Il se pourrait donc que de meilleures performances puissent être atteintes avec d'autres paramètres.

5 Conclusion

Au cours de ce projet, nous avons eu l'occasion d'étudier et implémenter une sous-classe des algorithmes les plus utilisés pour la construction de systèmes de recommandations. Nous avons implémenté différents algorithmes d'optimisation (SGD, ALS, CD) pour minimiser notre fonction de coût, et avons expérimentalement étudié l'influence des paramètres sur les solutions obtenus.

Le résultat le plus surprenant auquel nous sommes arrivés, et qui semble à priori rejoindre des constats précédemment établis dans l'état de l'art [2], est l'efficacité avec laquelle les termes de biais arrivent à expliquer les notes attribuées par les utilisateurs, ainsi que le nombre très faible de facteurs requis pour obtenir un bon score. Ces résultats contre intuitifs semblent difficiles à expliquer, mais pourraient provenir de la régularité des données. Après tout, les bons films ont majoritairement de bonnes notes, et la variabilité des notes attribuées par les utilisateurs devrait être relativement faible puisque les individus ont tendance à regarder (et donc noter) les films qu'ils sont susceptibles d'apprécier. Il se pourrait également que cet effet soit atténué sur des jeux de données plus grands.

Une direction que nous avions commencé à explorer, mais que nous n'avons pu mener à bout faute de temps, est l'incorporation dans le modèle de feedbacks implicites. L'idée, qui rejoint le constat précédent, consiste à exploiter les zéros dans la matrice de ratings. En effet, le fait de ne pas noter un film peut révéler que ce dernier ne correspond pas aux goûts de l'utilisateur (autrement, il l'aurait *peut-être* regardé). Autrement dit, avoir deux utilisateurs qui ont regardé les mêmes films peut suggérer qu'ils ont facteurs latents similaires, et ce *indépendemment* [1] des

[1] Il n'est pas raisonnable d'ignorer la valeur attribuée par les utilisateurs dans ce cas, même s'il existe des modèles simples qui le font. Néanmoins, la plupart des modèles intègrent ces 2 informations.

notes.

References

[1] C. C. Aggarwal *et al.*, *Recommender systems*. Springer, 2016.

[2] R. M. Bell and Y. Koren, "Lessons from the netflix prize challenge.," *SiGKDD Explorations*, vol. 9, no. 2, pp. 75–79, 2007.