Fast Matrix Factorization for Online Recommendation with Implicit Feedback

BOUVIER Oscar, DUPUY-ZINI Alexandre

February 1, 2022

Plan de la présentation

1. ALS générique
2. Fast eALS
3. Expérimentations en Offline et Online
4. Discussion
5. Conclusion

ALS Générique : définition du problème

- **Notations :**
  M utilisateurs, N items, R la matrice des interactions
  $\mathcal{R}$ l'ensemble des interactions
  K la dimension de l'espace de représentation
  p_u, q_i les vecteurs latents, regroupés dans P et Q

- **Objectif :**
  Apprendre les p_u et q_i pour reconstruire R^* avec $\hat{r}_{ui} = \langle p_u, q_i \rangle$

- **Fonction de perte :**

$$J = \sum_{u=1}^{M} \sum_{i=1}^{N} w_{ui}(r_{ui} - \hat{r}_{ui})^2 + \lambda \left(\sum_{u=1}^{M} ||p_u||^2 + \sum_{i=1}^{N} ||q_i||^2 \right)$$

ALS Générique : optimisation

- **Méthode ALS :**
  Optimisation itérative alternée. Pour un p_u,

$$J_u = ||W^u(r_u - Qp_u)||^2 + \lambda||p_u||^2$$

  avec $W^u = diag(w_{ui})$.

- **Optimisation :**
  Solution d'une régression Ridge :

$$p_u = \left(Q^T W^u Q + \lambda I\right)^{-1} Q^T W^u r_u$$

- **Complexité :**
  $O((M+N)K^3 + MNK^2)$

ALS Générique : poids uniforme

- **Pondération uniforme :**
  Poids w_0 pour les données implicites
  $Q^T W^u Q = w_0 Q^T Q + Q^T (W^u - W^0) Q$ avec $W^0 = diag(w_0)$

- **Complexité :**
  $O((M+N)K^3 + |\mathcal{R}|K^2)$

Fast eALS : element-wise learner

- **Optimisation coordonnée par coordonnée**
  Pour la f^{ieme} coordonnée de p_u,

$$p_{uf} = \frac{\sum_{i=1}^{N} w_{ui} q_{if} (r_{ui} - \hat{r}_{ui}^{f})}{\sum_{i=1}^{N} w_{ui} q_{if}^2 + \lambda}$$

  avec $\hat{r}_{ui}^{f} = \hat{r}_{ui} - p_{uf} q_{if}$

- **Complexité :**
  $O(MNK)$

Fast eALS : popularity-aware weighting strategy

- **Réécriture de la fonction de perte :**

$$J = \sum_{(u,i)\in\mathcal{R}} w_{ui}(r_{ui} - \hat{r}_{ui})^2 + \sum_{u=1}^{M} \sum_{i\notin\mathcal{R}_u} c_i \hat{r}_{ui}^2 + \lambda \left(\sum_{u=1}^{M} ||p_u||^2 + \sum_{i=1}^{N} ||q_i||^2 \right)$$

- **Pondération par la popularité :**

$$c_i = c_0 \frac{f_i^\alpha}{\sum_{j=1}^{N} f_j^\alpha}$$

Fast eALS : introduced method

- **Mutualisation des calculs :**
  Nouvelle solution pour la f^{ieme} coordonnée de p_u,

$$p_{uf} = \frac{\sum_{i \in \mathcal{R}_u} w_{ui} q_{if} (r_{ui} - \hat{r}^f_{ui}) - \sum_{i \notin \mathcal{R}_u} \hat{r}^f_{ui} c_i q_{if}}{\sum_{i \in \mathcal{R}_u} w_{ui} q^2_{if} + \sum_{i \notin \mathcal{R}_u} c_i q^2_{if} + \lambda}$$

  Pré-calcul des caches $S^q = \sum_{i=1}^{N} c_i q_i^T q_i$ and $S^p = P^T P$

- **Complexité :**
  Réduite à $O((M+N)K^2 + K|\mathcal{R}|)$

Fast eALS : online procedure

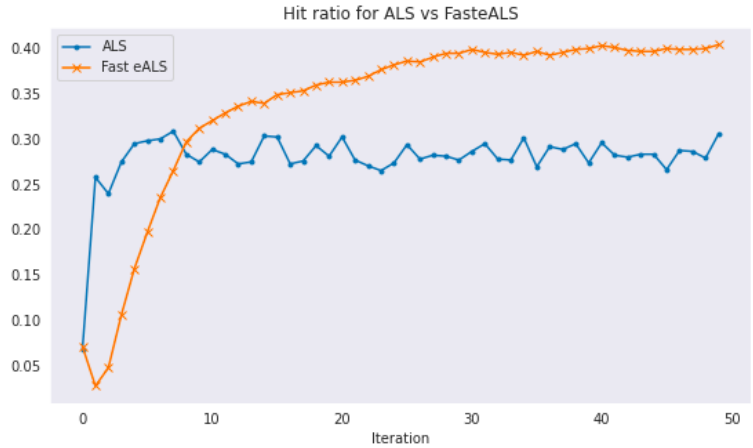
- **Méthode de rafraîchissement itérative :**
  Pour une nouvelle interaction (u, i), seuls q_i et p_u sont mis à jour
  Nécessite le choix d'un w_{new} et d'un nombre d'itérations

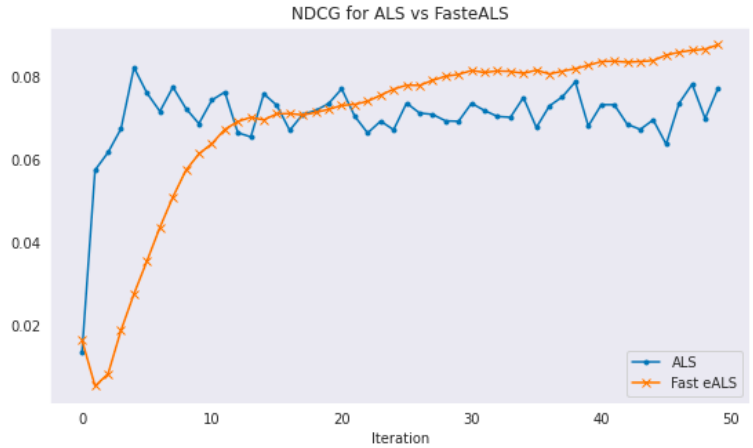
- **Complexité :**
  $O(K^2 + (|\mathcal{R}_u| + |\mathcal{R}_i|)K)$

Offline experiments

- **Données :**
  - MovieLens100k : 100k interactions, 1682 items, 943 utilisateurs, 93.7% de sparsité.
  - Yelp : 365k interactions, 22k items, 10k utilisateurs, 99,8% de sparsité.
- **Analyse de la performance de recommendation:** Leave-One-Out split pour chaque utilisateur, étude du *Hit ratio* et du *NDCG*.
- **Analyse du temps de calcul :** comparaison entre les deux algorithmes.

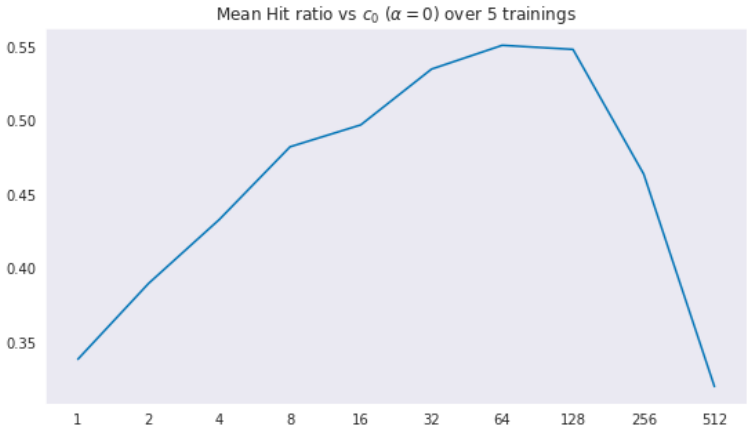
Offline - MovieLens100k

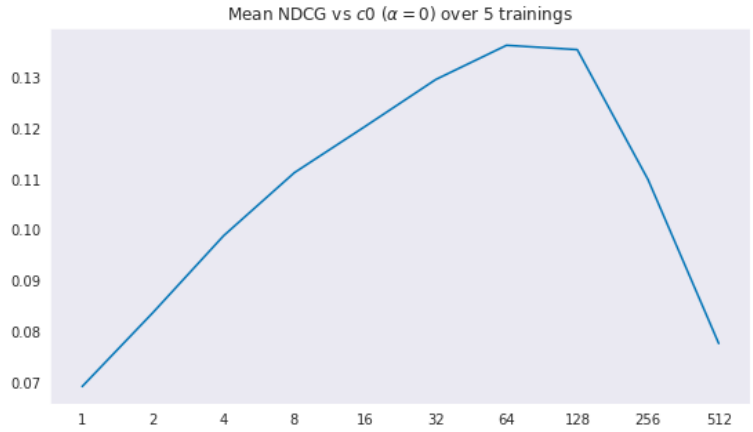




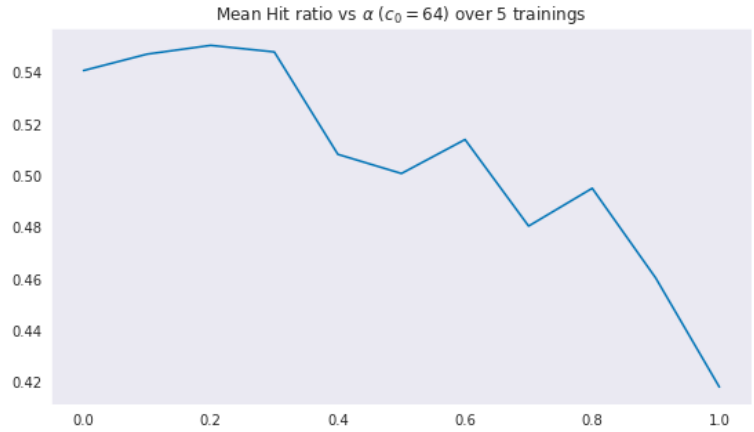
Hit ratio et *NDCG* pour ALS et FasteALS, $K = 20$, $c_0 = 4$, $\alpha = 0.8$

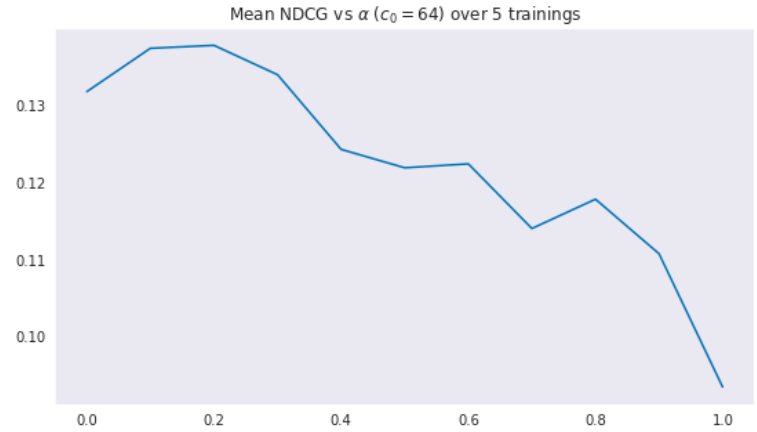
Offline - choix des hyperparamètres





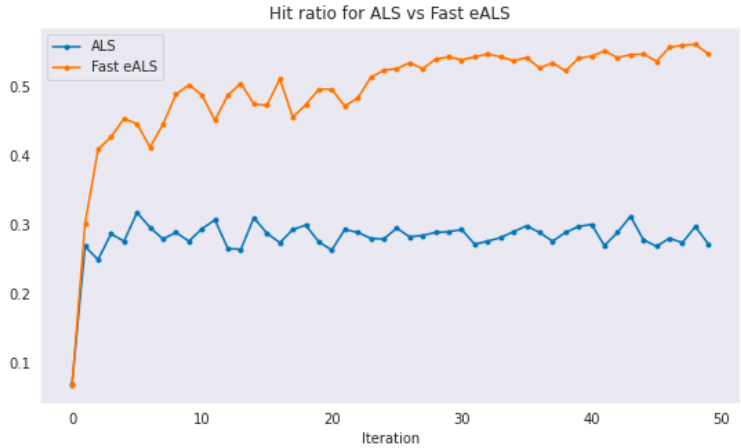
Hit ratio et *NDCG* en fonction de c_0, $\alpha = 0$

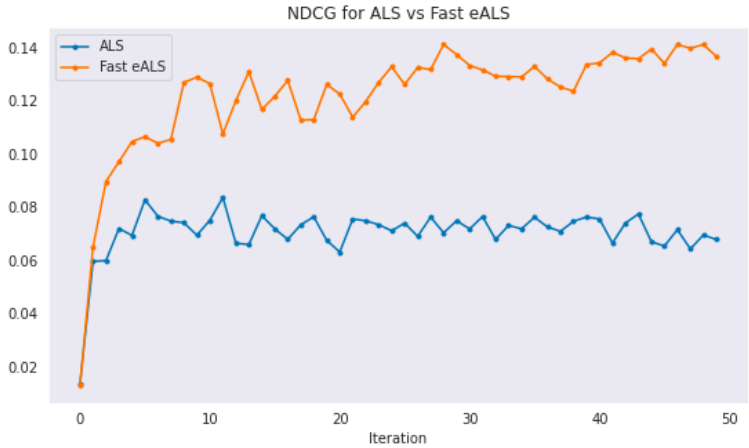




Hit ratio et *NDCG* en fonction de α, $c_0 = 64$

Offline - FasteALS optimisée



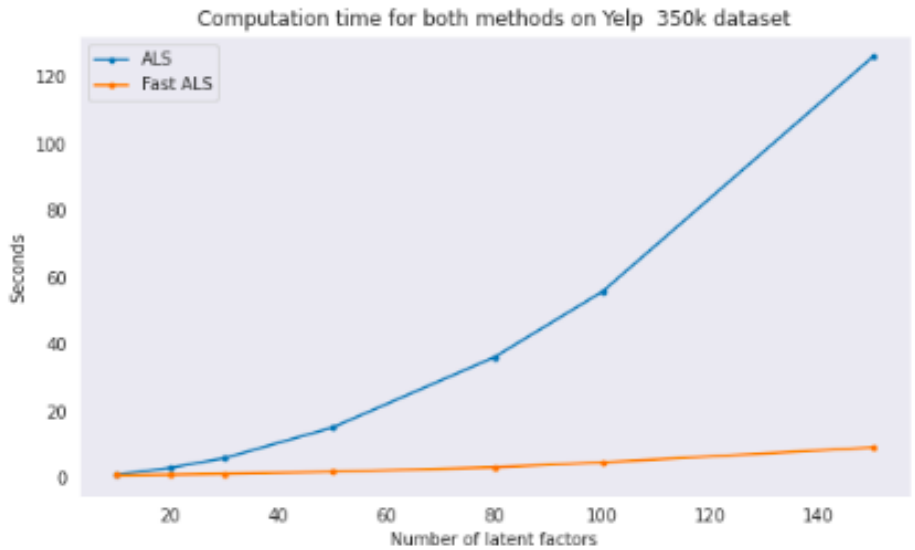


Hit ratio et *NDCG* pour ALS et FasteALS, $K = 20$, $c_0 = 64$, $\alpha = 0.3$

Offline - Analyse de la complexité temporelle

Complexités théoriques :

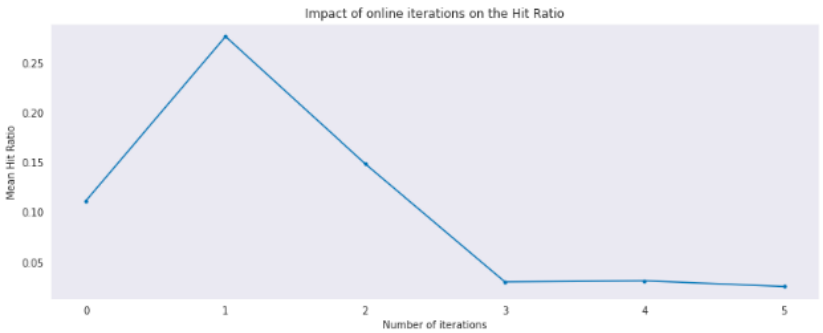
- ALS : $O((M+N)K^3 + |\mathcal{R}|K^2)$
- Fast eALS : $O((M+N)K^2 + |\mathcal{R}|K)$



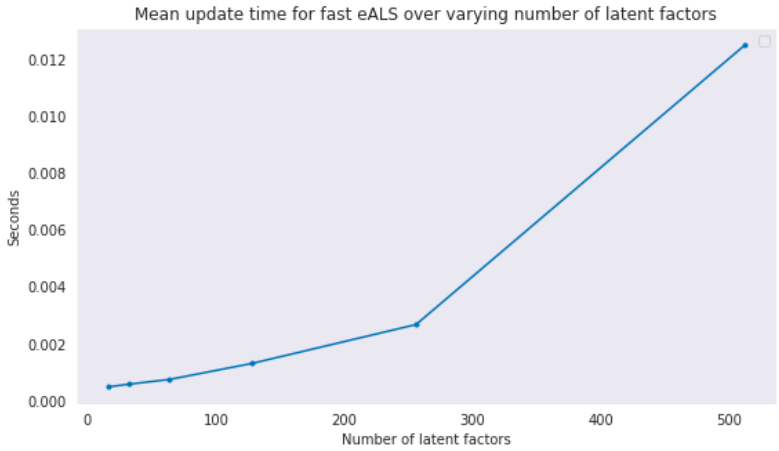
Temps de calcul pour ALS et Fast eALS - Yelp dataset

Online - MovieLens100k

- **Méthodologie :** on considère 90% des données d'entraînement (dans leur ordre d'apparition) et on update le modèle à chaque nouveau couple d'interaction item/utilisateur sur les 10% de données *online*.
- Étude de l'impact du nombre d'itérations optimale à chaque *update* et du temps de calcul *online*.



Hit ratio en fonction du nombre d'itérations *online*



Temps de calcul par itération *online*

Discussion

- **Commentaires sur les expérimentations** :
  on retrouve globalement les mêmes conclusions que le papier, Fast eALS performe mieux que l'ALS et est plus rapide.
- **Continuation du papier :**
  Truncated SVD pour permettre toute stratégie de pondération
- **Systèmes de recommendations par réseau de neurones** :
  méthodes MF : $r_{u,i} =< p_u, q_i >$, réseau de neurones $r_{u,i} = f(p_u, q_i)$
- **Parallèle avec le NLP :**
  méthodes d'embedding, negative sampling, utilisation des modèles d'attention pour modéliser les similarités entre utilisateurs et entre items ?

Conclusion

- Méthode très optimisée, plus flexible et moins contraignante
- Les résultats sont retrouvés même si notre implémentation pourrait être meilleure
- ALS populaire même si fast eALS plus performante. D'autres méthodes plus accessibles pourraient être préférées en pratique.