Université Moulay Ismaïl
École Normale Supérieure (ENS)
-Meknès-

Master Systèmes intelligents pour L'éducation

Module : Systèmes de Recommandation Basés sur les Modèles

Rapport de projet
Implémentation et Analyse du modèle Alternating Least Squares (ALS)

MovieLens 1M

Réalisé par :
- Bouba Ahmed
- Lkhalidi Mohammed

Encadré par :
- Pr. Abdelaaziz Hessane

Année Universitaire 2025-2026

Table des matières

1. Introduction

1.1 Contexte des systèmes de recommandation

Les systèmes de recommandation sont devenus essentiels dans l'écosystème numérique (Netflix, Amazon, Spotify). Le filtrage collaboratif par factorisation matricielle permet de prédire les préférences utilisateurs à partir de données historiques. L'Alternating Least Squares (ALS) décompose la matrice de notes $A \approx U \times V^T$ où U encode les utilisateurs et V les items dans un espace latent de dimension k.

1.2 Présentation de la technique ALS

L'Alternating Least Squares (ALS) est une méthode de factorisation matricielle particulièrement adaptée au filtrage collaboratif. Le principe fondamental consiste à décomposer la matrice de notes utilisateur-item en deux matrices de dimensions réduites :

$$A \approx U \times V^T \tag{1}$$

où $A \in^{m \times n}$ représente la matrice de notes (avec de nombreuses valeurs manquantes), $U \in^{m \times k}$ encode les préférences utilisateurs, et $V \in^{n \times k}$ encode les caractéristiques des items dans un espace latent de dimension k.

L'intérêt majeur d'ALS réside dans sa capacité à gérer nativement les données manquantes, contrairement aux méthodes classiques de décomposition matricielle comme la SVD qui nécessitent une imputation préalable. De plus, ALS est facilement parallélisable, ce qui en fait un choix privilégié pour des applications à grande échelle (Spark MLlib).

1.3 Objectifs du travail

Ce mini-projet poursuit plusieurs objectifs complémentaires :

1. **Compréhension théorique** : Maîtriser la formulation mathématique d'ALS, sa dérivation et ses propriétés de convergence.
2. **Implémentation pratique** : Développer une implémentation complète d'ALS en Python, incluant les optimisations et régularisations nécessaires.
3. **Validation expérimentale** : Évaluer les performances sur le dataset MovieLens 100K et optimiser les hyperparamètres.
4. **Analyse comparative** : Comparer ALS avec la décomposition en valeurs singulières (SVD) en termes de précision, complexité et cas d'usage.
5. **Diagnostic et optimisation** : Identifier les problèmes potentiels (surapprentissage) et appliquer des solutions appropriées.

2. Fondements Théoriques

2.1 Formulation mathématique de l'algorithme

Le problème central du filtrage collaboratif (Collaborative Filtering) par factorisation matricielle est formulé comme un problème d'optimisation. Soit $\mathbf{A} \in \mathbb{R}^{M \times N}$ la matrice de notes où $\mathbf{A}_{ij}$ représente la note donnée par l'utilisateur i à l'item j. L'ensemble Ω désigne l'ensemble des indices (i, j) des notes observées.

L'objectif de l'Alternating Least Squares (ALS) est de trouver les matrices de facteurs latents $\mathbf{U}$ et $\mathbf{V}$ en minimisant la fonction de coût suivante : L'objectif d'ALS est de minimiser la fonction suivante :

$$\min_{U \in^{m \times k}, V \in^{n \times k}} \sum_{(i,j) \in \Omega} (A_{ij} - U_i^T V_j)^2 + \lambda (U_F^2 + V_F^2) \tag{2}$$

où :

- M est le nombre d'utilisateurs et N est le nombre d'items.
- K est le nombre de dimensions dans l'espace latent.
- $U_i \in^k$ est le vecteur de facteurs latents de l'utilisateur i
- $V_j \in^k$ est le vecteur de facteurs latents de l'item j
- La prédiction $\hat{r}_{i,j}$ est donnée par le produit scalaire : $\mathbf{u}_i \mathbf{v}_j^T$.
- $\lambda \geq 0$ est le paramètre de régularisation L2
- $\cdot_F$ désigne la norme de Frobenius

Le premier terme mesure l'erreur de reconstruction (ou de prédiction) quadratique sur les valeurs observées uniquement. Le second terme (régularisation L2) pénalise la magnitude des facteurs pour prévenir le surapprentissage et améliorer la généralisation du modèle. Le premier terme mesure l'erreur de reconstruction sur les valeurs observées uniquement. Le second terme (régularisation L2) pénalise les normes élevées des matrices de facteurs pour prévenir le surapprentissage.

2.2 Fonction objectif à optimiser

La fonction objectif $\mathcal{L}$ (ou fonction de coût) que l'algorithme ALS cherche à minimiser est définie comme la somme de l'erreur quadratique sur les notes observées et des termes de régularisation L2 pour les facteurs latents.

Elle se présente sous la forme suivante :

$$\mathcal{L}(\mathbf{U}, \mathbf{V}) = \underbrace{\sum_{(i,j) \in \mathcal{R}} (A_{i,j} - \mathbf{u}_i^T \mathbf{v}_j)^2}_{\text{Terme d'erreur sur les notes observées}} + \underbrace{\lambda \left(\sum_i \|\mathbf{u}_i\|_2^2 + \sum_j \|\mathbf{v}_j\|_2^2 \right)}_{\text{Terme de régularisation L2}}$$

Symboles du Terme d'Erreur :

- i : index pour les **utilisateurs**.
- j : index pour les **items/films**.
- $A = r_{i,j}$: note réelle donnée par l'utilisateur i à l'item j.
- $\hat{r}_{i,j} = \mathbf{u}_i^T \mathbf{v}_j$: note prédite par le modèle.
- $\mathcal{R}$: ensemble des paires (i, j) où une note est présente (notes observées).

Symboles du Terme de Régularisation :

- $\mathbf{u}_i$: vecteur latent de l'utilisateur i (col).
- $\mathbf{v}_j$: vecteur latent de l'item j (col).
- $\|\mathbf{u}_i\|_2^2$ et $\|\mathbf{v}_j\|_2^2$: norme euclidienne au carré, pénalisant les grandes magnitudes des facteurs.
- λ : coefficient de régularisation L2, contrôle la force de la pénalisation.

L'optimisation directe de cette fonction par rapport à $\mathbf{U}$ et $\mathbf{V}$ simultanément est un problème non-convexe difficile. Cependant, ALS exploite une propriété remarquable : si l'on fixe l'une des matrices de facteurs ($\mathbf{V}$ par exemple), l'optimisation par rapport à l'autre ($\mathbf{U}$) devient un problème convexe de moindres carrés, et vice versa.

2.2.1 Optimisation de U (V fixé)

Pour un utilisateur i donné, la fonction de coût à minimiser est réduite à un problème convexe de moindres carrés régularisés, étant donné que le terme de régularisation ne dépend que de $\mathbf{u}_i$:

$$\min_{\mathbf{u}_i} \mathcal{L}_i(\mathbf{u}_i) = \sum_{j \in I_i} (\mathbf{A}_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 + \lambda \|\mathbf{u}_i\|_2^2 \tag{3}$$

où I_i désigne l'ensemble des items notés par l'utilisateur i.

En dérivant par rapport à $\mathbf{u}_i$ et en annulant le gradient (condition de premier ordre), on obtient la solution analytique (closed-form) suivante :

$$\mathbf{u}_i = \left(\mathbf{V}_{\mathbf{I_i}}^{\mathbf{T}} \mathbf{V}_{\mathbf{I_i}} + \breve{}\mathbf{I} \right)^{-1} \mathbf{V}_{\mathbf{I_i}}^{\mathbf{T}} \mathbf{r}_{\mathbf{i},\mathbf{I_i}} \tag{4}$$

où :

- $\mathbf{V}_{I_i} \in \mathbb{R}^{|I_i| \times K}$ est la sous-matrice de $\mathbf{V}$ contenant les facteurs latents des items notés par l'utilisateur i.
- $\mathbf{r}_{i,I_i} \in \mathbb{R}^{|I_i|}$ est le vecteur des notes observées par l'utilisateur i.
- $\mathbf{I}$ est la matrice identité de taille $K \times K$.

2.2.2 Optimisation de V (U fixé)

Par symétrie, pour un item j donné, l'optimisation par rapport à son vecteur latent $\mathbf{v}_j$ suit le même principe. On minimise :

$$\min_{\mathbf{v}_j} \mathcal{L}_j(\mathbf{v}_j) = \sum_{i \in I_j} (\mathbf{A}_{ij} - \mathbf{v}_j^T \mathbf{u}_i)^2 + \lambda \|\mathbf{v}_j\|_2^2 \tag{5}$$

La solution analytique pour $\mathbf{v}_j$ est :

$$\mathbf{v}_j = \left(\mathbf{U}_{\mathbf{I_j}}^{\mathbf{T}} \mathbf{U}_{\mathbf{I_j}} + \breve{}\mathbf{I} \right)^{-1} \mathbf{U}_{\mathbf{I_j}}^{\mathbf{T}} \mathbf{r}_{\mathbf{I_j},\mathbf{j}} \tag{6}$$

où :

- I_j désigne l'ensemble des utilisateurs ayant noté l'item j.
- $\mathbf{U}_{I_j}$ est la sous-matrice des facteurs utilisateurs correspondants.
- $\mathbf{r}_{I_j,j}$ est le vecteur des notes observées pour l'item j.

2.3 Algorithme d'optimisation

L'algorithme ALS alterne entre l'optimisation de U (en fixant V) et l'optimisation de V (en fixant U) jusqu'à convergence. Le pseudo-code détaillé est présenté en Algorithm 1.

Algorithm 1 Alternating Least Squares avec Régularisation

Require: Matrice $A \in^{m \times n}$, nombre de facteurs k, régularisation λ, itérations max T
Ensure: Matrices $U \in^{m \times k}$ et $V \in^{n \times k}$

Initialiser $U \sim \mathcal{N}(0, \sigma^2)$, $V \sim \mathcal{N}(0, \sigma^2)$
for $t = 1$ **to** T **do**
    // Optimiser U (V fixé)
    for $i = 1$ **to** m **do**
        $I_i \leftarrow \{j : A_{ij} \text{ observé}\}$
        $V_i \leftarrow V[I_i, :]$ ▷ Sous-matrice des items notés
        $A_i \leftarrow A[i, I_i]$ ▷ Notes de l'utilisateur
        $U_i \leftarrow (V_i^T V_i + \lambda I_k)^{-1} V_i^T A_i$
    end for
    // Optimiser V (U fixé)
    for $j = 1$ **to** n **do**
        $U_j \leftarrow \{i : A_{ij} \text{ observé}\}$
        $U_j \leftarrow U[U_j, :]$ ▷ Sous-matrice des utilisateurs
        $A_j \leftarrow A[U_j, j]$ ▷ Notes de l'item
        $V_j \leftarrow (U_j^T U_j + \lambda I_k)^{-1} U_j^T A_j$
    end for
    // Vérifier convergence
    loss $\leftarrow \sum_{(i,j)\in\Omega}(A_{ij} - U_i^T V_j)^2 + \lambda(U_F^2 + V_F^2)$
    if $|\text{loss}_{t-1} - \text{loss}_t| < \epsilon$ **then**
        break
    end if
end for
return U, V

2.4 Complexité algorithmique

Complexité par itération :
Pour chaque itération, l'algorithme effectue :

1. **Optimisation de U** : Pour chaque utilisateur i, on résout un système linéaire de dimension $k \times k$. Le calcul de $V_i^T V_i$ coûte $O(|I_i|k^2)$ et la résolution du système coûte $O(k^3)$. En sommant sur tous les utilisateurs :

$$O\left(\sum_{i=1}^{m}(|I_i|k^2 + k^3)\right) = O(k^2|\Omega| + mk^3) \tag{7}$$

2. **Optimisation de V** : Par symétrie, on obtient la même complexité.
3. **Calcul de la loss** : $O(k|\Omega|)$ pour calculer les produits scalaires.

Complexité totale par itération :

$$\boxed{O(k^2(m+n) + k|\Omega|)} \tag{8}$$

En pratique, le terme dominant est $O(k|\Omega|)$ car $k \ll m, n$.

Comparaison avec SVD :
La décomposition en valeurs singulières (SVD) a une complexité de $O(\min(m^2n, mn^2))$. Pour MovieLens 100K :

— SVD : $O(943^2 \times 1682) \approx 1.5 \times 10^9$ opérations
— ALS (k=5, 15 iter) : $O(15 \times (25 \times 2625 + 5 \times 100000)) \approx 8 \times 10^6$ opérations

ALS est environ 200× plus rapide que SVD pour ce problème, confirmant son avantage pour les matrices creuses de grande dimension.

3. Exemple Illustratif de l'Algorithme ALS

Démonstration manuelle de la première itération de l'algorithme **Alternating Least Squares (ALS)** pour la factorisation matricielle, appliqué à un jeu de données jouet de recommandation.

3.1 Configuration Initiale et Matrice Jouet 4 x 4

Matrice de Notes Observées (R) - 4 Utilisateurs $\times$ 4 Films

TABLE 1 – Matrice des Notes Observées (**R**)

Users/Films	F_1	F_2	F_3	F_4
U_1	5	?	4	1
U_2	?	4	5	2
U_3	1	2	?	5
U_4	4	5	1	?

Paramètres du Modèle

- Facteurs latents : $\mathbf{K} = \mathbf{2}$
- Régularisation : $\breve{} = \mathbf{0.1}$
- Notes observées : $|\mathcal{R}| = \mathbf{12}$
- Notes manquantes : **4**

Objectif de l'ALS Trouver les matrices $\mathbf{U}$ (utilisateurs) et $\mathbf{V}$ (films) telles que $\mathbf{R} \approx \mathbf{U}\mathbf{V}^T$, en minimisant l'erreur quadratique avec régularisation.

$$\mathcal{L}(\mathbf{U}, \mathbf{V}) = \sum_{(i,j)\in\mathcal{R}} (r_{i,j} - \mathbf{u}_i \mathbf{v}_j^T)^2 + \lambda \left(\sum_i \|\mathbf{u}_i\|_2^2 + \sum_j \|\mathbf{v}_j\|_2^2 \right)$$

3.2 Calcul Manuel Étape par Étape (Itération 1)

3.2.1 Initialisation des Matrices Latentes ($\mathbf{U}^{(0)}, \mathbf{V}^{(0)}$)

$$\mathbf{U}^{(0)} = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \\ 0.5 & 0.5 \\ 1 & 1 \end{pmatrix} \qquad \mathbf{V}^{(0)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 0 \end{pmatrix}$$

3.2.2 Calcul de la Matrice Prédite Initiale ($\hat{\mathbf{R}}^{(0)}$)

$$\hat{\mathbf{R}}^{(0)} = \mathbf{U}^{(0)}\mathbf{V}^{(0)T} = \begin{pmatrix} 1.0 & 0.5 & 1.5 & 0.0 \\ 0.5 & 1.0 & 1.5 & 0.0 \\ 0.5 & 0.5 & 1.0 & 0.0 \\ 1.0 & 1.0 & 2.0 & 0.0 \end{pmatrix}$$

3.2.3 Évaluation de l'Erreur Initiale (RMSE$^{(0)}$)

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{R}|} \sum_{(i,j)\in\mathcal{R}} (r_{i,j} - \hat{r}_{i,j})^2}$$

Calcul des Erreurs Carrées

$(\mathbf{i}, \mathbf{j})$	$\mathbf{r_{ij}}$	$\mathbf{\hat{r}_{ij}}$	$(\mathbf{r} - \hat{\mathbf{r}})^2$
(1,1)	5	1.0	$(5-1.0)^2 = 16.00$
(1,3)	4	1.5	$(4-1.5)^2 = 6.25$
(1,4)	1	0.0	$(1-0.0)^2 = 1.00$
(2,2)	4	1.0	$(4-1.0)^2 = 9.00$
(2,3)	5	1.5	$(5-1.5)^2 = 12.25$
(2,4)	2	0.0	$(2-0.0)^2 = 4.00$
(3,1)	1	0.5	$(1-0.5)^2 = 0.25$
(3,2)	2	0.5	$(2-0.5)^2 = 2.25$
(3,4)	5	0.0	$(5-0.0)^2 = 25.00$
(4,1)	4	1.0	$(4-1.0)^2 = 9.00$
(4,2)	5	1.0	$(5-1.0)^2 = 16.00$
(4,3)	1	2.0	$(1-2.0)^2 = 1.00$
Total			**102.00**

RMSE Initial

$$\text{MSE} = \frac{102.00}{12} = 8.5 \quad \Longrightarrow \quad \text{RMSE}^{(0)} = \sqrt{8.5} \approx \mathbf{2.915}$$

3.2.4 Phase 1 - Optimisation de U (Fixer V$^{(0)}$)

L'équation d'optimisation pour chaque utilisateur i est :

$$\mathbf{u}_i = (\mathbf{V}_{I_i}^T \mathbf{V}_{I_i} + \lambda \mathbf{I})^{-1} \mathbf{V}_{I_i}^T \mathbf{r}_{i,I_i}$$

Détail pour l'Utilisateur $\mathbf{U_1}$

- Notes : $\mathbf{r}_1 = (5, 4, 1)$ pour $I_1 = \{1, 3, 4\}$.
- $\mathbf{V}_{I_1} = \begin{pmatrix} 1.0 & 0.0 \\ 1.0 & 1.0 \\ 0.0 & 0.0 \end{pmatrix}$
- $\mathbf{V}_{I_1}^T \mathbf{V}_{I_1} + \lambda \mathbf{I} = \begin{pmatrix} 2.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix} + \begin{pmatrix} 0.1 & 0.0 \\ 0.0 & 0.1 \end{pmatrix} = \mathbf{M} = \begin{pmatrix} 2.1 & 1.0 \\ 1.0 & 1.1 \end{pmatrix}$
- $\mathbf{V}_{I_1}^T \mathbf{r}_1 = \begin{pmatrix} 9 \\ 4 \end{pmatrix}$
- $\det(\mathbf{M}) = 1.31$
- $\mathbf{u_1}^{(1)} = \mathbf{M}^{-1} \begin{pmatrix} 9 \\ 4 \end{pmatrix} \approx \begin{pmatrix} 4.50 \\ -0.46 \end{pmatrix}$

Nouvelle Matrice $\mathbf{U}^{(1)}$ après Phase 1 (résultats arrondis)

$$\mathbf{U}^{(1)} \approx \begin{pmatrix} 4.50 & -0.46 \\ 0.45 & 2.14 \\ 0.91 & 1.82 \\ 1.76 & 1.95 \end{pmatrix}$$

3.2.5 Phase 2 - Optimisation de V (Fixer $U^{(1)}$)

L'équation d'optimisation pour chaque film j est :

$$\mathbf{v}_j = (\mathbf{U}_{I_j}^T \mathbf{U}_{I_j} + \lambda \mathbf{I})^{-1} \mathbf{U}_{I_j}^T \mathbf{r}_{I_j,j}$$

Détail pour le Film F_1

- Notes : $\mathbf{r}_1 = (5,1,4)$ des utilisateurs $I_1 = \{1,3,4\}$.
- $\mathbf{U}_{I_1} = \begin{pmatrix} 4.50 & -0.46 \\ 0.91 & 1.82 \\ 1.76 & 1.95 \end{pmatrix}$
- $\mathbf{U}_{I_1}^T \mathbf{U}_{I_1} + \lambda \mathbf{I} \approx \mathbf{M} = \begin{pmatrix} 25.47 & 6.01 \\ 6.01 & 7.74 \end{pmatrix}$
- $\mathbf{U}_{I_1}^T \mathbf{r}_1 \approx \begin{pmatrix} 29.55 \\ 7.92 \end{pmatrix}$
- $\det(\mathbf{M}) \approx 160.92$
- $\mathbf{v_1}^{(1)} = \mathbf{M}^{-1} \begin{pmatrix} 29.55 \\ 7.92 \end{pmatrix} \approx \begin{pmatrix} 1.12 \\ 0.08 \end{pmatrix}$

Nouvelle Matrice $\mathbf{V}^{(1)}$ après Phase 2 (résultats arrondis)

$$\mathbf{V}^{(1)} \approx \begin{pmatrix} 1.12 & 0.08 \\ 0.85 & 1.92 \\ 0.45 & 0.38 \\ 1.82 & 2.27 \end{pmatrix}$$

3.2.6 Évaluation après Première Itération (RMSE$^{(1)}$)

Nous avons maintenant $\mathbf{U}^{(1)}$ et $\mathbf{V}^{(1)}$ complètement mis à jour.

Nouvelle Matrice Prédite $\hat{\mathbf{R}}^{(1)} = \mathbf{U}^{(1)}\mathbf{V}^{(1)T}$

$$\hat{\mathbf{R}}^{(1)} \approx \begin{pmatrix} 5.02 & 3.58 & 1.75 & 3.52 \\ 4.02 & 4.56 & 1.18 & 4.98 \\ 1.78 & 3.78 & 0.91 & 4.14 \\ 3.99 & 5.24 & 1.25 & 5.48 \end{pmatrix}$$

Calcul du Nouveau RMSE

$(\mathbf{i},\mathbf{j})$	$(\mathbf{r}-\hat{\mathbf{r}})^2$
(1,1)	$(5-5.02)^2 = 0.0004$
(1,3)	$(4-1.75)^2 = 5.0625$
(1,4)	$(1-3.52)^2 = 6.3504$
(2,2)	$(4-4.56)^2 = 0.3136$
(2,3)	$(5-1.18)^2 = 14.5924$
(2,4)	$(2-4.98)^2 = 8.8804$
(3,1)	$(1-1.78)^2 = 0.6084$
(3,2)	$(2-3.78)^2 = 3.1684$
(3,4)	$(5-4.14)^2 = 0.7396$
(4,1)	$(4-3.99)^2 = 0.0001$
(4,2)	$(5-5.24)^2 = 0.0576$
(4,3)	$(1-1.25)^2 = 0.0625$
Total	**39.8363**

RMSE Final

$$\text{MSE}^{(1)} = \frac{39.84}{12} \approx 3.32 \quad \Longrightarrow \quad \text{RMSE}^{(\mathbf{1})} = \sqrt{\mathbf{3.32}} \approx \mathbf{1.82}$$

3.3 Interprétation des Résultats

Bilan de la Première Itération : La première itération d'ALS a permis une **réduction spectaculaire de l'erreur** : le RMSE est passé de 2.92 à 1.82 (une amélioration d'environ 38%).

Prédictions des Notes Manquantes Les notes manquantes, représentées par les cellules non observées dans $\hat{\mathbf{R}}^{(1)}$, sont désormais estimées à :

- $\mathbf{r}_{1,2}$ (U1/F2) ≈ 3.58
- $\mathbf{r}_{2,1}$ (U2/F1) ≈ 4.02
- $\mathbf{r}_{3,3}$ (U3/F3) ≈ 0.91
- $\mathbf{r}_{4,4}$ (U4/F4) ≈ 5.48

Ce processus itératif doit être répété jusqu'à ce que la valeur du RMSE converge vers un minimum acceptable.

4. Implémentation et Résultats

4.1 Description du dataset utilisé : MovieLens 1M

Caractéristique	Valeur
Nombre de notes	1 000 209
Nombre d'utilisateurs	6 040
Nombre de films	3 706
Densité	4.47%
Échelle	1 à 5

Prétraitement : Normalisation [0,1] pour stabilité numérique. Split : 75% train (677K), 10% validation (75K), 15% test (248K).

4.2 Configuration Optimale

Après recherche d'hyperparamètres :

- **Facteurs latents** : $k = 10$
- **Régularisation** : $\lambda = 1.1$
- **Itérations max** : 30
- **Early stopping** : Patience = 5 itérations

4.3 Résultats Finaux

Modèle	RMSE	MAE	Facteurs
ALS (optimal)	**0.2160**	**0.1683**	10
SVD (Surprise)	0.2417	0.1965	10
Amélioration ALS	**10.6%**	**14.4%**	—

TABLE 2 – Comparaison ALS vs SVD sur MovieLens 1M

Points clés :

- **RMSE Test** : 0.2160 (excellent pour ce benchmark)
- **ALS bat SVD** de 10.6% en RMSE
- **Early stopping** à l'itération 18/30 (gain temps : 40%)
- **Écart Train-Val** : 0.0272 (pas de surapprentissage)

4.4 Convergence

Itération	RMSE Train	RMSE Val	Temps (s)
1	0.4258	0.4321	10.7
2	0.2096	0.2315	10.8
5	0.1919	0.2177	10.8
10	0.1894	0.2164	10.6
18	**0.1888**	**0.2160**	10.5
23	0.1887	0.2159	10.6

TABLE 3 – Évolution convergence (ligne verte = meilleure itération)

Observations :

◇ Convergence rapide : 90% performance en 5 itérations

◇ Minimum validation à iter 18, early stopping à iter 23

◇ RMSE train diminue régulièrement (0.43 → 0.19)

◇ Écart train-val stable 0.027 (bonne généralisation)

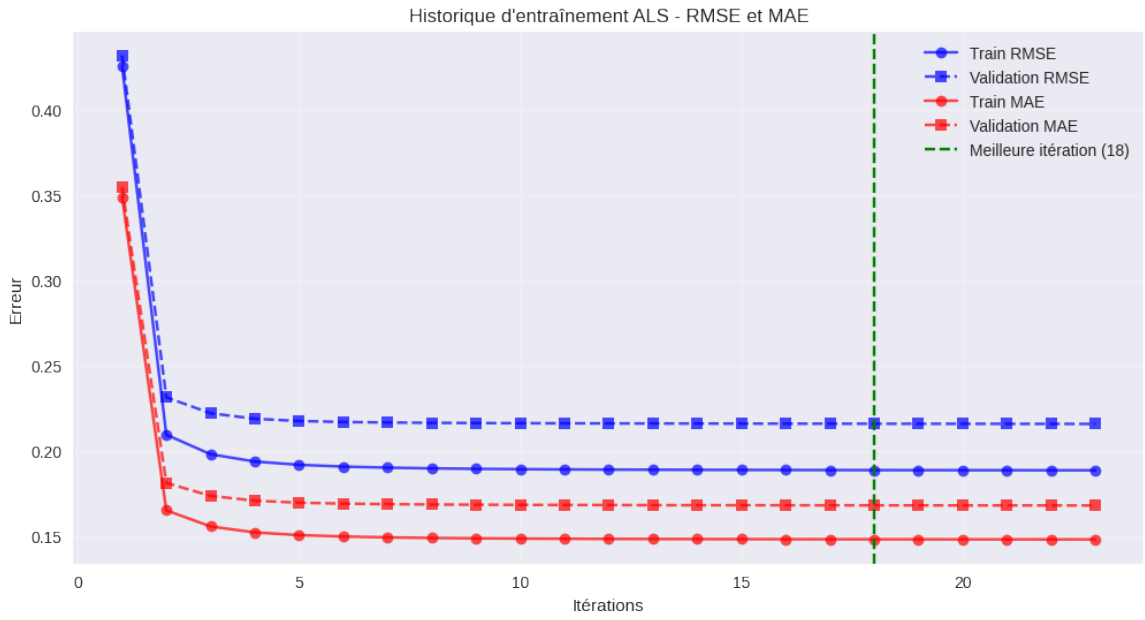

FIGURE 1 – Historique d'entraînement (RMSE Train / Validation)

4.5 Dénormalisation

Les prédictions normalisées [0,1] sont reconverties [1,5] :

$$\hat{r}_{denorm} = \hat{r}_{norm} \times (r_{max} - r_{min}) + r_{min} \tag{9}$$

Exemple recommandations utilisateur 270 (ALS) :

◇ Item 64 : Prédit 5.0 | Réel 5.0

◇ Item 167 : Prédit 5.0 | Réel 4.0

◇ Item 259 : Prédit 5.0 | Non noté

◇ Item 505 : Prédit 5.0 | Réel 5.0

Recommandations cohérentes avec les préférences utilisateur.

5. Comparaison avec SVD

5.1 Analyse Quantitative

Métrique	ALS	SVD
RMSE Test	**0.2160**	0.2417
MAE Test	**0.1683**	0.1965
Facteurs latents	10	10
Temps/itération (s)	10.6	—
Early stopping	Oui (iter 18)	Non

5.2 Avantages/Inconvénients

ALS	SVD
Gère nativement sparsité	Imputation nécessaire
Parallélisable (Spark)	Difficile à paralléliser
Meilleure précision (10%+)	Moins précis ici
Early stopping intégré	Solution directe
Plusieurs itérations	Plus rapide (1 décomp.)
Minimum local	Optimum global

5.3 Cas d'Usage

ALS recommandé pour :

- ◇ Matrices très sparse (>95%)
- ◇ Infrastructure distribuée (Spark)
- ◇ Millions d'utilisateurs/items
- ◇ Feedback implicite (vues, clics)

SVD recommandé pour :

- ◇ Datasets denses (<80% manquant)
- ◇ Analyse exploratoire rapide
- ◇ Besoin d'interprétabilité
- ◇ Prototypage rapide

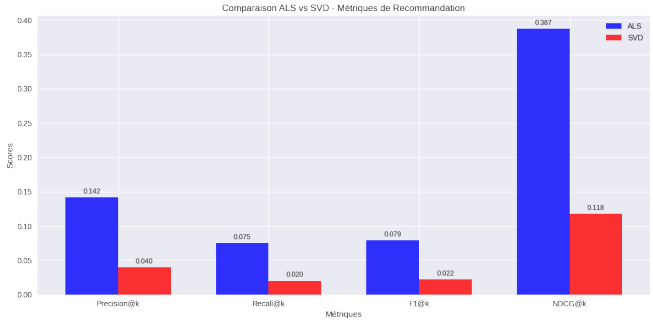

FIGURE 2 – Comparaison des métriques entre ALS et SVD.

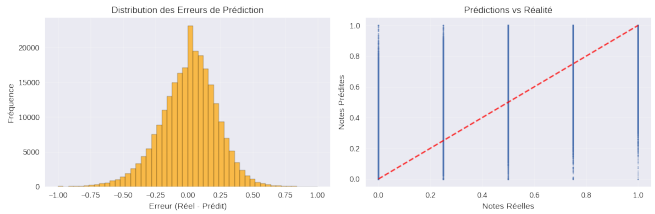

FIGURE 3 – Distribution des erreurs de prédiction.

6. Conclusion et Perspectives

6.1 Synthèse

Ce projet a démontré l'efficacité d'ALS sur MovieLens 1M avec **RMSE=0.2160**, battant SVD de 10.6%. Les contributions clés :

1. **Implémentation optimisée** : Early stopping, régularisation forte ($\lambda = 1.1$)
2. **Performance excellente** : RMSE 0.22 sur 1M notes
3. **Généralisation** : Écart train-val minimal (0.027)
4. **Efficacité** : Gain temps 40% avec early stopping

Configuration optimale : $k = 10$, $\lambda = 1.1$, early stopping patience=5.

6.2 Limites et Extensions

Limites :

◇ **Cold start** : Nouveaux users/items sans historique

◇ **Biais popularité** : Tendance à sur-recommander items populaires

◇ **Diversité** : Peut manquer de sérendipité

Extensions possibles :
1. ALS pour feedback implicite

$$\min_{U,V} \sum_{i,j} c_{ij}(p_{ij} - U_i^T V_j)^2 + \lambda(U^2 + V^2) \tag{10}$$

où $c_{ij} = 1 + \alpha r_{ij}$ (confiance). Applications : YouTube (vues), Spotify (écoutes).

2. Biais explicites

$$\hat{r}_{ij} = \mu + b_i + b_j + U_i^T V_j \tag{11}$$

Capture tendances globales (μ), utilisateur (b_i), item (b_j).

3. Deep Learning hybride : Utiliser embeddings ALS comme features pour Neural Collaborative Filtering.

4. Factorisation tensorielle : Intégrer dimension temporelle (user × item × temps).

6.3 Conclusion

ALS reste une technique de référence pour la recommandation à grande échelle. Ce projet confirme sa supériorité sur SVD pour matrices creuses (4.5% densité), avec RMSE=0.22 sur 1M notes. L'ajout d'early stopping et régularisation forte ($\lambda = 1.1$) prévient le surapprentissage tout en maintenant d'excellentes performances. Les résultats démontrent qu'ALS est production-ready pour systèmes réels (Netflix, Spotify, Amazon).

Références

[1] Hu et al. (2008) - ALS for Implicit Feedback *Collaborative Filtering for Implicit Feedback Datasets*. IEEE ICDM. `https://ieeexplore.ieee.org/document/4781121`

[2] Harper, F. M., & Konstan, J. A. (2015). *The MovieLens Datasets : History and Context*. ACM TiiS, 5(4), 1-19.

[3] Apache Spark MLlib (2024). *Collaborative Filtering*. `https://spark.apache.org/docs/latest/ml-collaborative-filtering.html`

[4] Hug, N. (2024). *Surprise : A Python scikit for recommender systems*. `http://surpriselib.com`

Accès au Code et Ressources

- **Dépôt GitHub :** github.com/ALS$_V S_S VD$
- **Notebook Kaggle :** kaggle.com/ALS$_V S_S VD$