# Rapport de la thèse : Factorisation Matricielle, Application à la Recommandation Personnalisée de Préférences

## Table des matières



# Introduction aux Systèmes de Recommandation et Enjeux Associés

Les systèmes de recommandation occupent aujourd'hui une place centrale dans de nombreux domaines, notamment le commerce en ligne, les plateformes de streaming et les réseaux sociaux. Leur objectif principal est d'offrir des suggestions personnalisées aux utilisateurs en fonction de leurs préférences et des caractéristiques des articles ou services proposés. Cette capacité à personnaliser l'expérience utilisateur est devenue un levier stratégique pour de nombreuses entreprises cherchant à améliorer la satisfaction client, la fidélisation et la conversion.

Dans cette thèse, l'étude approfondie des systèmes de recommandation commence par un tour d'horizon de leur évolution historique et des principales problématiques qu'ils rencontrent. Ensuite, les approches classiques et modernes seront explorées, en mettant l'accent sur des méthodes avancées telles que la factorisation matricielle. Enfin, nous examinerons les principaux défis techniques, éthiques et pratiques qui influencent leur efficacité et leur adoption à grande échelle.

# 1. Évolution et Fondements des Systèmes de Recommandation

### 1.1 Contexte Historique

L'émergence des systèmes de recommandation remonte aux années 1990, avec le développement d'outils pionniers tels que **Tapestry**, conçu pour recommander des documents à partir des annotations et préférences des utilisateurs. Par la suite, des systèmes comme **Ringo**

(pour la musique) et le moteur de recommandation d'**Amazon** ont contribué à structurer ce domaine en l'intégrant dans les expériences d'achat en ligne et de consommation de contenu.

Aujourd'hui, ces technologies sont omniprésentes et intégrées dans de multiples services, allant de la recommandation de produits sur les plateformes e-commerce à la suggestion de contacts sur les réseaux sociaux, en passant par la personnalisation des playlists musicales ou des séries sur les plateformes de streaming.

Un tournant majeur a été marqué en 2007 avec le **Netflix Prize**, une compétition lancée par Netflix pour améliorer son moteur de recommandation "Cinematch". Ce concours a mis en lumière de nouvelles techniques avancées, notamment la **factorisation matricielle**, qui a montré des performances supérieures aux approches traditionnelles. Toutefois, l'intégration de ces modèles en production a posé de nombreux défis, notamment en termes de complexité computationnelle et d'interprétabilité.

### 1.2 Problématiques Clés des Systèmes de Recommandation

Les systèmes de recommandation doivent répondre à plusieurs défis majeurs :

- **Nature des données** : Les matrices utilisateur/article sont **sparse** (peu d'interactions réellement renseignées) et peuvent atteindre des tailles massives, rendant leur exploitation complexe.
- **Multiplicité des objectifs** : Selon le contexte d'utilisation, les recommandations peuvent viser à **augmenter les ventes**, diversifier les suggestions, améliorer l'engagement des utilisateurs ou encore mieux comprendre leur comportement.
- **Différents types de recommandations** : Certains systèmes recommandent des **produits**, d'autres des **amis** (réseaux sociaux), des **lieux** (services de navigation) ou des **contenus** (à court/long terme, en fonction des habitudes d’utilisation).

### 1.3 Principales Approches Utilisées

On distingue trois grandes catégories d'approches en recommandation :

1. **Approches Classiques** :
    - **Basées sur le contenu** : Ces méthodes exploitent les caractéristiques des articles pour proposer des recommandations similaires.
    - **Filtrage collaboratif** : Elles s'appuient sur les interactions passées des utilisateurs pour identifier des profils similaires et déduire de nouvelles préférences.
    - **Modèles hybrides** : Ils combinent les deux approches précédentes afin de pallier leurs limites respectives.
2. **Approches Contextuelles** :
    - Ces modèles intègrent des informations additionnelles telles que l'âge, la localisation ou le moment de la journée pour affiner les recommandations.
    - Ils peuvent être appliqués en **prétraitement**, en **post-traitement**, ou directement incorporés dans les modèles de recommandation.

### 1.4 Obstacles et Contraintes

- **Scalabilité et Performance** : Le volume de données traité est souvent massif, nécessitant des algorithmes optimisés et des architectures adaptées pour assurer des temps de réponse rapides.
- **Problèmes éthiques** : La recommandation repose sur des données personnelles, soulevant des questions de **vie privée**, de **biais algorithmiques** et de **transparence**.
- **Diversité et Exploration** : Un système trop performant risque de limiter les utilisateurs à des recommandations redondantes (effet de bulle de filtre), réduisant ainsi la diversification des contenus explorés.

### 1.5 Perspectives et Futur des Systèmes de Recommandation

Avec l'évolution des besoins et des technologies, les systèmes de recommandation continuent de se perfectionner. Plusieurs axes de développement se dessinent :

- **Intégration du deep learning** : L'utilisation des réseaux neuronaux profonds permet de capter des patterns complexes et d'améliorer la personnalisation des recommandations.
- **Prise en compte des dynamiques temporelles** : Certains modèles évoluent pour mieux anticiper les changements de comportement des utilisateurs.
- **Nouvelles sources de données** : Le **crowdsourcing** et l'exploitation des **réseaux sociaux** offrent des perspectives pour enrichir la personnalisation.

En somme, les systèmes de recommandation sont un domaine en constante évolution, combinant enjeux techniques, éthiques et stratégiques. Leur amélioration repose sur l'optimisation des modèles existants, l'exploration de nouvelles méthodologies et l'intégration de nouvelles dimensions contextuelles.

# 2. Factorisation et Recommandation : Un État de l'Art

Cette section examine les techniques de factorisation matricielle et leur application aux systèmes de recommandation, en explorant les différentes approches, les méthodes de résolution et les critères d'évaluation.

### 2.1 Formalisation du Problème de Factorisation

#### 2.1.1 Notation et Formulation Générale

La factorisation matricielle vise à décomposer une matrice YY (représentant par exemple les interactions utilisateurs/articles) en deux matrices UU et VV contenant des facteurs latents. L'objectif est de reconstruire YY en minimisant une fonction d'attache aux données tout en ajoutant un terme de régularisation pour éviter le sur-apprentissage.

#### 2.1.2 Différentes Approches de Factorisation

Les techniques de factorisation reposent sur plusieurs hypothèses fondamentales :

1. La matrice YY est de faible rang.
2. Certaines informations sont redondantes, tandis que d'autres sont parcimonieuses.

3. Les utilisateurs et les articles peuvent être regroupés en clusters partageant des caractéristiques communes.

Les approches principales incluent :

- **Approximation** : Trouver une matrice de faible rang proche de YY.
- **Complétion** : Remplir les valeurs manquantes dans YY.
- **Décomposition** : Identifier des structures parcimonieuses et des composantes significatives.
- **Classification croissée** : Identifier des groupes d'utilisateurs et d'articles.
- **Ordonnancement** : Hiérarchiser les articles recommandés pour chaque utilisateur.

#### 2.1.3 Critères d'Attache aux Données et Régularisation

Le choix du critère d'attache influence directement la précision de la factorisation. Les approches courantes incluent :

- **Erreur quadratique moyenne (RMSE)** pour une minimisation des erreurs.
- **Métriques probabilistes** basées sur la maximisation de la vraisemblance.

La régularisation est cruciale pour contrôler la complexité du modèle et éviter le sur-apprentissage. Elle impose des contraintes sur la norme des matrices UU et VV.

#### 2.1.4 Nature des Problèmes et Matrice de Bruit

La nature des données (complètes, parcimonieuses, bruitées) détermine la stratégie de factorisation. Un modèle $Y=M+R$ inclut une matrice de bruit RR, dont l'influence peut affecter les performances des méthodes de factorisation.

### 2.2 La Factorisation dans la Recommandation

#### 2.2.1 Méthodes Exactes pour Grandes Matrices Creuses

Les grandes matrices utilisateur/article présentent un taux de remplissage faible, ce qui pose des problèmes de stockage et de calcul. Des solutions existent :

- **Décomposition en Valeurs Singulières (SVD)** pour extraire les facteurs principaux.
- **Bi-diagonalisation** pour réduire la complexité du calcul.
- **Diagonalisation des matrices réduites** pour une meilleure efficacité.

#### 2.2.2 Algorithmes d'Optimisation

- **Descente de Gradient Stochastique (SGD)** : Une approche populaire qui met à jour les paramètres en fonction des observations disponibles.
- **Optimisation Alternée** : Divise le problème en sous-problèmes convexes successivement résolus.
- **Algorithme Multiplicatif** : Utilisé pour la factorisation non-négative (NMF) et adapté aux données à contraintes spécifiques.

- **Factorisation Tensorielle** : Étend la factorisation matricielle en intégrant plusieurs dimensions (temps, contexte).

### 2.3 Méthodes d'Évaluation

#### 2.3.1 Choix et Validation des Modèles

La sélection du modèle dépend du type de données et de l'application visée. L'évaluation repose sur :

- **Validation croisée** : Mesure la généralisation du modèle sur des ensembles de test.
- **Expérimentations sur des jeux de données synthétiques ou réels**.
- **Métriques de performance** : RMSE, précision, rappel et NDCG pour évaluer la pertinence des recommandations.

### 2.4 Conclusion

Les méthodes de factorisation sont des outils puissants pour la recommandation, permettant de capturer des relations complexes dans les données utilisateur/article. Cependant, elles présentent des limites, notamment en termes de complexité computationnelle et d'adaptabilité à des données dynamiques. Leur amélioration repose sur l'optimisation des algorithmes, l'intégration de nouvelles sources d'information et la prise en compte de contraintes réglementaires et éthiques.

# 3. Le rôle fondamental du contexte dans les systèmes de recommandation

La troisième partie de cette thèse, intitulée **"Le rôle fondamental du contexte dans les systèmes de recommandation"**, met en évidence l'importance des informations contextuelles dans l'optimisation des systèmes de recommandation. L'objectif est de démontrer comment l'intégration du contexte peut enrichir les prédictions et offrir des recommandations plus précises et adaptées aux utilisateurs.

### 3.1 Types de données contextuelles

Les données contextuelles constituent un levier essentiel pour affiner les recommandations en tenant compte de la situation unique de chaque utilisateur. Elles se déclinent en plusieurs catégories :

#### 3.1.1 Variables utilisateurs et produits

- Profil des utilisateurs : inclut des caractéristiques démographiques (âge, sexe, localisation, préférences personnelles, historique d’achats, etc.).
- Attributs des produits : classification par catégorie, marque, gamme de prix, disponibilité, etc.
- Objectif : personnaliser les recommandations en alignant les offres aux besoins et habitudes des utilisateurs.

#### 3.1.2 Données temporelles

- Influence des moments de la journée, des saisons et des événements sur les préférences et comportements d'achat.
- Capture des tendances et évolutions dynamiques de la demande.
- Exemples : promotions à certaines périodes, pic de consultation des plateformes en soirée.

#### 3.1.3 Données sociales

- Exploitation des interactions sociales et des réseaux sociaux (avis, commentaires, partages, recommandations d'amis).
- Modélisation de l'influence sociale sur les choix et décisions d'achat.

#### 3.1.4 Informations contextuelles implicites

- Habitudes et comportements récurrents analysés à partir des données d'interaction (ex. : utilisateur achetant régulièrement les mêmes articles à certaines périodes).
- Approches fondées sur l'apprentissage automatique pour détecter des schémas comportementaux.

### 3.2 Différentes approches de contextualisation

L'intégration du contexte peut être effectuée à divers niveaux du processus de recommandation :

#### 3.2.1 Prétraitement

- Filtrage des données en amont pour sélectionner uniquement celles qui sont pertinentes selon le contexte.
- Exemple : si un utilisateur recherche un produit en hiver, le système peut filtrer les articles d'été.

#### 3.2.2 Post-traitement

- Ajustement des recommandations après leur génération en fonction du contexte utilisateur.
- Exemples : réorganisation des articles selon leur pertinence contextuelle, mise en avant des produits en promotion.

#### 3.2.3 Modélisation contextuelle

- Intégration directe du contexte dans le modèle de recommandation (ex. : intégration des données contextuelles dans un modèle de factorisation matricielle).
- Approche plus complexe mais plus précise, adaptée aux systèmes basés sur l'apprentissage profond.

### 3.3 Application concrète : recommandation d'outils pour professionnels

#### 3.3.1 Présentation du problème

- Cas d'étude centré sur la recommandation d'outils pour des professionnels du bâtiment.

- Besoin d'adapter les suggestions au contexte d'utilisation (ex. : travaux de grande ampleur vs petits chantiers).

**3.3.2 Approches contextuelles appliquées**

- **Contexte temporel (3.3.2.1)** : analyse des variations saisonnières de la demande.
- **Contexte de chantier (3.3.2.2)** : adaptation des recommandations selon le type de projet.

**3.3.3 Protocole expérimental**

- **Méthodes de factorisation (3.3.3.1)** : choix des techniques de modélisation contextuelle.
- **Evaluation (3.3.3.2)** : comparaison de la précision des recommandations.
- **Tests de contextualisation** : simulations sur différents scénarios temporels et projets.

**3.4 Discussion**

- Amélioration notable des performances grâce à la contextualisation.
- Défis de collecte et de traitement des données.
- Importance d'un compromis entre complexité et précision.

# 4. Intégration des réseaux sociaux dans les systèmes de recommandation

**4.1 Concepts fondamentaux**

- Représentation des utilisateurs sous forme de nœuds dans un graphe social.
- Exploitation des liens sociaux pour améliorer la personnalisation des recommandations.

**4.2 Etat de l'art**

- **Formalisation (4.2.1)** : modélisation des graphes et matrices utilisateur-article.
- **Ajout de contraintes (4.2.2)** : influence des relations sociales sur les recommandations.
- **Modification des fonctions de décision (4.2.3)** : prise en compte des interactions sociales dans les modèles prédictifs.

**4.3 Recommandation et apprentissage d'influence**

- **Influence des amis (4.3.1)** : intégration de la confiance inter-utilisateurs.
- **Optimisation (4.3.2)** : mise à jour des facteurs latents (utilisateurs et articles) et inclusion des matrices de relations sociales.
- **Expérimentations (4.3.3)** :
    - Problèmes de collecte et traitement des données sociales.

- Protocole d'évaluation des performances.
- Comparaison avec des méthodes classiques.

**4.4 Discussion**

- Amélioration des recommandations grâce aux interactions sociales.
- Défis liés à la complexité computationnelle et à l'exploitation des données personnelles.

Cette partie met en évidence le potentiel des réseaux sociaux dans la personnalisation des systèmes de recommandation et explore les implications pratiques et éthiques de leur intégration.

# 5.Sélection de modèle pour la factorisation

**5.1 La sélection de modèle**

**5.1.1 Cadre général de la sélection de modèle**

La sélection de modèle consiste à choisir la méthode de factorisation la plus adaptée parmi plusieurs alternatives en fonction de critères quantitatifs et de validations empiriques. Ce processus repose sur des approches systématiques permettant d'optimiser la précision des prédictions tout en assurant une bonne généralisation aux nouvelles données.

Les bases méthodologiques incluent :

- **La définition de métriques d'évaluation** telles que l'erreur quadratique moyenne (MSE), l'erreur absolue moyenne (MAE) ou encore la divergence de Kullback-Leibler.
- **L'utilisation de validations croisées** (cross-validation) afin d'éviter le sur-apprentissage et de tester la robustesse des modèles.
- **L'analyse de la complexité computationnelle** pour s'assurer que le modèle est adapté aux ressources disponibles et aux contraintes de scalabilité.

**5.1.2 Estimation du risque**

L'évaluation du risque est essentielle dans le choix d'un modèle, car elle permet de quantifier l'incertitude et les erreurs potentielles dans les prédictions.

- Un modèle est jugé optimal s'il minimise une fonction de risque qui mesure la divergence entre les valeurs prédites et les valeurs réelles.
- Les approches incluent l'estimation empirique du risque par validation croisée, l'utilisation d'indicateurs tels que le biais et la variance, ainsi que la prise en compte du compromis entre biais et variance.

**5.1.3 Critères de sélection, pénalisation et coût computationnel**

Le choix d'un modèle repose sur plusieurs critères fondamentaux :

- **Critères de performance** : MSE, MAE, $R^2$, etc.
- **Régularisation et pénalisation** : Ajout de termes de pénalité comme la norme L1 (lasso) ou L2 (ridge) pour éviter le sur-apprentissage.

- **Coût computationnel** : Analyse du temps de calcul, de la mémoire requise et de l'évolutivité du modèle à grande échelle.

### 5.2 Sélection de modèle dans la factorisation

#### 5.2.1 État de l'art

La sélection de modèle en factorisation a fait l'objet de nombreuses recherches, notamment dans les systèmes de recommandation. Les approches les plus courantes incluent :

- **Les méthodes basées sur des pénalités**, qui ajoutent une contrainte à la fonction d'optimisation pour éviter le sur-ajustement.
- **Les approches bayésiennes et probabilistes**, qui permettent d'intégrer une incertitude sur les paramètres du modèle.
- **Les techniques de validation empirique**, comme la validation croisée, qui testent les performances sur des sous-ensembles des données.

#### 5.2.2 Formalisation du problème

Le problème de sélection de modèle dans la factorisation peut être formulé de manière mathématique comme suit :

- Soit une matrice **R** représentant les préférences d'utilisateurs sur des items, où certains éléments sont manquants.
- Le but est de trouver une décomposition $\mathbf{R \approx U\Sigma V^T}$ qui minimise un critère de coût tout en respectant des contraintes de généralisation et de complexité computationnelle.
- Dans le contexte des systèmes de recommandation, il est crucial d'intégrer des contraintes spécifiques telles que la sparsité des données et la dynamique des préférences utilisateurs.

#### 5.2.3 Pénalité et estimation du risque

- L'introduction de termes de régularisation permet de contrôler la complexité du modèle et d'éviter le sur-ajustement.
- L'estimation du risque peut être réalisée à l'aide de métriques comme le risque de prédiction moyen et la variance du modèle.

#### 5.2.4 Estimateur sans biais du risque

- Afin d'obtenir une estimation plus fidèle des performances du modèle, il est crucial d'éliminer les biais introduits par les données incomplètes ou bruitées.
- Des techniques comme la rééchantillonnage bootstrap ou l'ajustement bayésien permettent d'améliorer la robustesse de ces estimations.

#### 5.2.5 Calcul de la divergence

- L'objectif est d'évaluer la différence entre la distribution réelle des données et les prédictions du modèle.
- Des métriques comme la **divergence de Kullback-Leibler** ou la **divergence de Jensen-Shannon** sont souvent utilisées pour quantifier cette différence.

#### 5.2.6 Limites des modèles actuels

- Les approches traditionnelles peuvent être limitées par la capacité à traiter des **données massives** ou très hétérogènes.
- La nécessité d'**adapter les modèles à des environnements en évolution rapide** reste un défi majeur.

### 5.3 Adaptation et élargissement de la méthode

#### 5.3.1 Amélioration de la fonction de seuillage

- L'adoption de nouvelles fonctions de seuillage permet d'améliorer la prise en compte des structures latentes des données.
- **Propositions de fonctions alternatives** :
    - Seuillage dynamique basé sur l'entropie
    - Approches adaptatives selon le taux de sparsité des données
- **Protocole expérimental** : Comparaison des nouvelles fonctions de seuillage sur différents jeux de données.
- **Résultats expérimentaux** : Analyse des performances obtenues et comparaison avec les approches traditionnelles.

#### 5.3.2 Passage à l'échelle

- L'un des défis majeurs reste la scalabilité des méthodes de sélection de modèle.
- **Approximation de la divergence** :
    - Techniques de réduction de la complexité computationnelle (échantillonnage, factorisation approximative).
- **Protocole expérimental et résultats** : Test de ces méthodes sur des jeux de données volumineux.

#### 5.3.3 Estimation de la variance

- La quantification de la variance est essentielle pour comprendre l'incertitude associée aux modèles sélectionnés.
- **Protocole expérimental et résultats** : Mesure de l'impact de la prise en compte de la variance sur la précision des prédictions.

### 5.4 Discussion et perspectives

- Analyse des contributions des approches proposées.
- Discussion des **limites persistantes**, notamment en termes d'**interprétabilité** des modèles et d'adaptabilité à des contextes variés.
- Perspectives pour de futures recherches :
    - Intégration de **modèles plus complexes**, comme les réseaux de neurones profonds.

- Exploration d'**approches hybrides**, combinant factorisation et apprentissage supervisé pour améliorer la précision des recommandations.

Cette étude approfondie apporte des contributions significatives à la sélection de modèle en factorisation, notamment pour améliorer la précision et l'efficacité des **systèmes de recommandation** tout en tenant compte des contraintes computationnelles et des enjeux liés aux données massives.

**Conclusion générale**

Cette thèse met en lumière les progrès significatifs réalisés dans le domaine des systèmes de recommandation, tout en identifiant les défis persistants qui nécessitent encore des solutions adaptées. Parmi ces défis, la gestion efficace des volumes croissants de données, l'intégration fine du contexte utilisateur et les considérations éthiques liées à la transparence et à la protection des données personnelles demeurent des enjeux majeurs.

Les approches explorées, telles que la factorisation matricielle et l'intégration contextuelle, ont démontré leur efficacité pour améliorer la pertinence et la personnalisation des recommandations. Néanmoins, ces méthodes présentent certaines limitations, notamment en termes de scalabilité et de capacité à capter la dynamique des préférences utilisateur en temps réel.

L'avenir des systèmes de recommandation repose ainsi sur l'adoption de techniques plus avancées et plus robustes, notamment celles issues de l'intelligence artificielle, telles que l'apprentissage profond et les modèles de reinforcement learning. Par ailleurs, l'essor des modèles hybrides, combinant différentes méthodes de filtrage (collaboratif, basé sur le contenu, contextuel), ouvre la voie à des recommandations plus précises et adaptatives.

Ces avancées offrent des perspectives prometteuses pour de nombreux secteurs d'application, allant de l'e-commerce et du divertissement à la santé et à la finance. L'optimisation des algorithmes de recommandation, associée à une meilleure prise en compte des enjeux éthiques et de l'explicabilité des décisions algorithmiques, permettra d'améliorer leur adoption et leur impact. En définitive, les recherches futures devront viser un équilibre entre performance, personnalisation et respect des valeurs éthiques pour assurer une expérience utilisateur optimale et durable.

**Factorisation Matricielle avec Descente de Gradient Stochastique**

Ce document présente une implémentation en Python de la factorisation matricielle utilisant la descente de gradient stochastique (SGD) avec régularisation. Cette technique est couramment utilisée dans les systèmes de recommandation pour approximer des matrices utilisateur-produit avec des valeurs manquantes.

# Code Python : Factorisation

**1. Introduction**

La factorisation matricielle est une approche puissante permettant de réduire la dimensionnalité des données en décomposant une matrice RR en deux matrices plus petites :

- **P** : matrice des utilisateurs (dimension : utilisateurs × facteurs latents)

- **Q** : matrice des produits (dimension : produits × facteurs latents)

L'objectif est de trouver ces deux matrices de telle sorte que leur produit approximatif soit le plus proche possible de la matrice d'origine RR, minimisant ainsi l'erreur de reconstruction.

### 2. Implémentation en Python

#### 2.1 Importation des bibliothèques

```
import numpy as np
```

#### 2.2 Définition de la classe MatrixFactorization

```
class MatrixFactorization:

    def __init__(self, R, num_factors=10, learning_rate=0.01, regularization=0.1,
max_iter=1000, tol=1e-4):

        """

        Modèle de factorisation matricielle utilisant la descente de gradient stochastique
avec régularisation.



        Paramètres :

        - R : Matrice utilisateur-produit avec des valeurs manquantes remplies par 0.

        - num_factors : Nombre de facteurs latents.

        - learning_rate : Taux d'apprentissage.

        - regularization : Coefficient de régularisation pour éviter le sur-apprentissage.

        - max_iter : Nombre maximal d'itérations.

        - tol : Seuil de tolérance pour la convergence.

        """

        self.R = np.array(R)

        self.num_users, self.num_items = R.shape

        self.num_factors = num_factors

        self.learning_rate = learning_rate

        self.regularization = regularization

        self.max_iter = max_iter

        self.tol = tol


    def fit(self):

        """

        Entraîne le modèle en utilisant la descente de gradient stochastique.

        """

        self.P = np.random.rand(self.num_users, self.num_factors)
```

```
        self.Q = np.random.rand(self.num_items, self.num_factors)

        previous_error = float('inf')
        for iteration in range(self.max_iter):
            for i in range(self.num_users):
                for j in range(self.num_items):
                    if self.R[i, j] > 0:  # Ignorer les valeurs manquantes
                        error_ij = self.R[i, j] - np.dot(self.P[i, :], self.Q[j, :])
                        for k in range(self.num_factors):
                            self.P[i, k] += self.learning_rate * (2 * error_ij * self.Q[j,
k] - self.regularization * self.P[i, k])
                            self.Q[j, k] += self.learning_rate * (2 * error_ij * self.P[i,
k] - self.regularization * self.Q[j, k])

            total_error = self._calculate_error()
            if iteration % 100 == 0:
                print(f"Iteration {iteration}, Erreur : {total_error:.4f}")

            if abs(previous_error - total_error) < self.tol:
                print("Convergence atteinte.")
                break
            previous_error = total_error

    def _calculate_error(self):
        """
        Calcule l'erreur totale avec régularisation.
        """
        error = 0
        for i in range(self.num_users):
            for j in range(self.num_items):
                if self.R[i, j] > 0:
                    error += (self.R[i, j] - np.dot(self.P[i, :], self.Q[j, :]))**2
        error += self.regularization * (np.linalg.norm(self.P) + np.linalg.norm(self.Q))
        return error

    def predict(self):
        """
```

```
        Reconstruit la matrice approximative des préférences.
        """
        return np.dot(self.P, self.Q.T)
```

**2.3 Exécution du modèle**

```
if __name__ == "__main__":
    # Exemple de matrice utilisateur-produit
    R = np.array([
        [5, 3, 0, 1],
        [4, 0, 0, 1],
        [1, 1, 0, 5],
        [1, 0, 0, 4],
        [0, 1, 5, 4],
    ])


    model = MatrixFactorization(R, num_factors=3, learning_rate=0.01, regularization=0.1,
max_iter=5000, tol=1e-5)
    model.fit()
    predictions = model.predict()


    print("\nMatrice prédite :")
    print(predictions)
```

### 3. Explication du Code

1. **Initialisation du modèle** :
    - La matrice d'entrée RR est fournie par l'utilisateur.
    - Les matrices PP et QQ sont initialisées aléatoirement.
2. **Phase d'apprentissage** :
    - La descente de gradient stochastique met à jour les poids de PP et QQ en minimisant l'erreur entre les valeurs observées et prédites.
    - Un terme de régularisation est inclus pour éviter le sur-apprentissage.
    - L'entraînement s'arrête lorsque la convergence est atteinte (changement d'erreur inférieur à un seuil défini).
3. **Prédiction** :

- Une fois entraîné, le modèle peut prédire les valeurs manquantes en reconstruisant la matrice complète.

---

**4. Conclusion**

La factorisation matricielle est une méthode efficace pour recommander des éléments dans des bases de données utilisateur-produit. L'utilisation de la descente de gradient permet une mise à jour itérative des paramètres pour améliorer progressivement la précision des recommandations. Ce modèle peut être adapté et amélioré avec des variantes telles que la factorisation non-négative ou l'intégration d'approches basées sur l’apprentissage profond.