Université de Paris

UFR Maths-Info

Rapport de Projet

Responsable de l'UE *Apprentissage et factorisation matricielle* :

- Nejmeddine ALLAB

Étudiants :

- Hacene ISSELNANE
- Ayale HADDAD

Master 2 MLDS

Année universitaire : 2020/2021

1. Définition du projet

Le projet s'inscrit dans le cadre de l'UE Apprentissage et factorisation matricielle, son objectif est de permettre de mettre en pratique les connaissances théoriques vues en cours.

1.1. Objectif du projet

L'objectif est d'implémenter deux modèles de prédiction pour une problématique de recommandations de contenus vidéo. Les prédictions portent sur l'intérêt exprimé par un utilisateur sur un contenu (film, série, etc.). La recommandation consiste à proposer au client une liste de contenus ordonnés selon l'intérêt estimé.

2. Description succincte de l'implémentation

Nous avons conçu le projet avec les objectifs suivants :

- Soulager la manipulation des jeux de données : Les utilisateurs peuvent charger leurs propres ensembles de données rien qu'en précisant les noms des colonnes contenant les données à utiliser.
- Optimiser les algorithmes de prédiction afin de minimiser le temp de calcul et exploiter au mieux les ressources lors de l'exécution.
- Implémenter des approches en formalisme objet.

Nous avons choisi de structurer le projet en trois fichiers :

- Un fichier « *Dataset* » : Celui-ci contient un ensemble de méthodes permettant de générer une structure sous forme d'une base de données contenant (user id, item id, rating associés etc…), qui par la suite sera utilisée par les algorithmes de prédiction. Permettant ainsi de construire une base commune aux deux méthodes implémentées.
- Un fichier « Baseline » : Il contient l'implémentation de la méthode décrite dans l'article [Koren].
- Un fichier « SVD++ » : Ce dernier, à l'instar du fichier « baseline » contient notre implémentation de la méthode svd++.

Nous nous sommes inspirés du célèbre package « *sci-kit learn* ».

3. Approche Baseline

L'algorithme Baseline cherche à minimiser la fonction d'erreur indiquée dans l'article [Koren] :

$$\sum_{(u,i) \in k} (r_{ui} - b_{ui})^2 + \lambda_1 \left(\sum_u b_u^2 + \sum_i b_i^2 \right)$$

Cette minimisation est réalisée en effectuant une descente de gradient stochastique, dont les paramètres par défaut sont :

- *« reg »* : Le paramètre de régularisation de la fonction de coût qui est optimisée, correspondant à λ_1 dans [*Koren*]. Sa valeur par défaut est de *0,02*.
- *« learning_rate »* : Le taux d'apprentissage correspondant à ɣ dans [*Koren*]. Sa valeur par défaut est de *0,005*.
- *« n_epochs »* : Le nombre d'itération de la procédure de décente de gradient. Sa valeur par défaut a été fixée à 20.

4. Approche SVD++

Notre implémentation prend en compte ces paramètres :

- *« n_factors »* : Le nombre de facteurs, sa valeur par défaut est de 20.
- *« n_epochs »* : Le nombre d'itérations de la procédure de descente de gradient, sa valeur par défaut est de 20 (convergence atteinte au bout de 10 pour baseline et 15 pour SVD++).
- *« init_mean »* : La moyenne de la distribution normale pour l'initialisation des vecteurs de facteurs dont la valeur par défaut est 0.
- *« init_std_dev »* : L'écart-type de la distribution normale pour l'initialisation des vecteurs de facteurs, sa valeur par défaut est 0,1.
- *« lr_all »* : Le taux d'apprentissage pour tous les paramètres, dont la valeur par défaut est 0,007.
- *« reg_all »* : Le terme de régularisation pour tous les paramètres dont la valeur par défaut est de 0,02.
- *« random_state »* : Détermine la valeur de départ qui sera utilisé pour l'initialisation.

Le taux d'apprentissage ainsi que la régularisation peuvent également être appliqués indépendamment pour chaque paramètre (b_i, b_u, p_u, q_i, y_j). Par défaut ils prennent respectivement les valeurs *lr_all* et *reg_all*.

5. Résultats

Après avoir ajusté les modèles sur le jeu de données d'entrainement. Nous les avons par la suite testés sur le reste des données non vues pendant la phase d'apprentissage.

Voici les résultats obtenus :

	Baseline	SVD++
RMSE	0.5393	0.3912
Temps/Epoch (minutes)	0.5	25

On peut constater que l'approche SVD++ obtient la plus basse RMSE, mais est cependant beaucoup plus couteuse (en temps et en mémoire).

L'approche Baseline est, quant à elle plus efficiente tout en obtenant une *RMSE* légèrement plus élevée.