Introduction

Définition CF
Attentes (précision/temps de réponse/ajout rapide d'un rating/cold start...)
Efficacité peut varier selon la forme du jeu de données (sparsity...)

1 Les algorithmes utilisés

1.1 Algorithmes témoins

Algorithmes naïfs servant de point de comparaison :
- Estimer tous les ratings inconnus par la moyenne de tous les ratings possibles
- Estimer tous les ratings inconnus pour un utilisateur donné par la moyenne des ratings connus qu'il a attribués
- Estimer la note donnée par un utilisateur à un objet en fonction de sa moyenne, mais aussi des écarts entre les notes attribuées à l'objet et les moyennes des utilisateurs qui l'ont noté
- Retirer les biais comme dans le cours, estimer tous les ratings inconnus par 0, remettre les biais

1.2 SVD

Variantes sur la question bonus du DM (comment traiter les trous dans la matrice, avec ou sans traitement des biais...).

1.3 Algorithmes Slope One

Prédicteur affine, en imposant une pente de 1. Pour tout couple d'objet, on définit une déviation de l'un par rapport à l'autre, un peu comme la mesure de similarité cosinus, mais en beaucoup plus simple (linéaire). La prédiction de la note attribuée à un objet s'obtient alors en ajoutant la moyenne de l'utilisateur et la moyenne des déviations de l'objet aux autres objets notés par l'utilisateur. D'où un prédicteur de forme $f(x) = x + b$.
Calculs simples. Possibilité de garder les déviations en mémoire : traitement d'une requête très rapide. De plus, ces déviations peuvent même être mises à jour sans tout recalculer quand on ajoute un rating.
À quel point est-ce moins précis que d'autres algos plus sophistiqués ?
Variante : donner plus de poids aux déviations des paires d'objets qui ont été notés tous deux à la fois par un grand nombre d'utilisateurs, car elles sont plus fiables.
Algorithmes tirés de [2].

1.4 Algorithmes par similarité cosinus

Implémentation de l'algorithme vu en cours.

1.5 Analyse en composantes principales : algorithme Eigentaste

Idée : s'appuyer sur un petit sous-ensemble d'objets notés par tous les utilisateurs (*gauge set*) pour projeter un utilisateur sur un espace de petite dimension puis estimer ses notes à partir de celles de ses voisins au sens d'un algorithme de clustering.
Défauts : on demande à un nouvel utilisateur de noter l'intégralité du gauge set pour l'ajouter, et le cold start pose problème.
La précision est-elle vraiment meilleure que celle d'autres algorithmes plus simples ?

Algorithme extrait de [1].

2 Observations expérimentales

2.1 Jeux de données

– Matrice de la question bonus du DM (pleine, on observe seulement une certaine fraction des ratings)
– Jeu de données Jester (ratings d'une centaine de blagues, dix blagues sont notées par toutes les utilisateurs) utilisé pour Eigentaste.
– Jeu de données plus grand (MovieLens) pour mesurer les difficultés liées aux temps d'exécutions dans des conditions plus réalistes ?

2.2 Mesures d'erreur

2.3 Temps d'exécution

Conclusion

Références

[1] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste : A constant time collaborative filtering algorithm. *Inf. Retr.*, 4(2) :133–151, July 2001.

[2] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. *CoRR*, abs/cs/0702144, 2007.