Algorithmes pour les systèmes de recommandation : un comparatif

Introduction

Définition : collaborative filtering
Position du problème : Attentes vis-à-vis des algorithmes (précision/temps de réponse/ajout rapide d'un rating/cold start...)
Efficacité peut varier selon la forme du jeu de données (sparsity...)
Objectif : Implémenter (en Matlab) et comparer différents algorithmes en terme de précision et de temps d'exécution sur différents jeux de données, mettre en relief les compromis à faire.

1 Les algorithmes utilisés

1.1 Algorithmes témoins

Algorithmes naïfs servant de point de comparaison :
– "Witness" : Estimer tous les ratings inconnus par la moyenne de tous les ratings possibles
– "PerUserAverage" : Estimer tous les ratings inconnus pour un utilisateur donné par la moyenne des ratings connus qu'il a attribués
– "BiasFromMean" : 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
– "UnbiasedWitness" : 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...).
– "ShiftSVD" : translater la matrice pour avoir 0 de moyenne, SVD, retranslater
– "UnbiasedSVD" : enlever les biais, SVD, remettre les 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 des algorithmes vus en cours : "itemCosSimilarity", "userCosSimilarity".

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.
Algorithme extrait de [1].
Notons R la matrice $(n \times m)$ des ratings, G le gauge set, $k = |G|$.
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 ?

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 ?

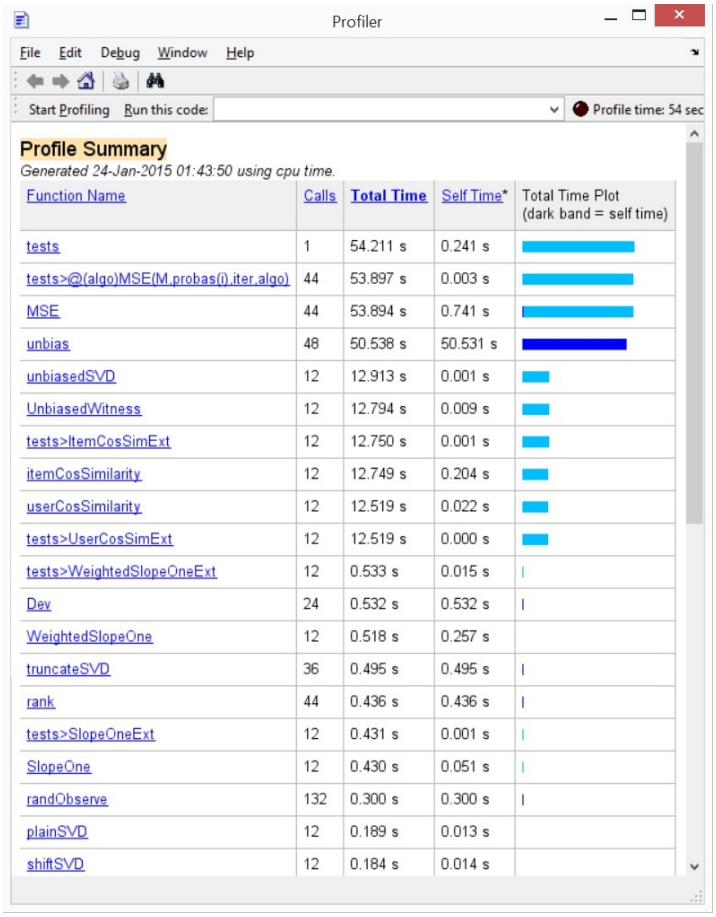
2.2 Mesures d'erreur

p=	0.2	0.4	0.6	0.8
Witness	0.0148	0.0112	0.0074	0.0037
UnbiasedWitness	0.0085	0.0062	0.0041	0.0020
PerUserAverage	0.0149	0.0111	0.0075	0.0037
BiasFromMean	0.0086	0.0062	0.0041	0.0021
PlainSVD	0.0940	0.0490	0.0221	0.0084
ShiftSVD	0.0106	0.0065	0.0037	0.0016
UnbiasedSVD	0.0087	0.0060	0.0036	0.0016
UserCosSim	0.0094	0.0083	0.0076	0.0070
ItemCosSim	0.0091	0.0077	0.0062	0.0049
SlopeOne	0.0086	0.0062	0.0042	0.0020
WeightedSlopeOne	0.0085	0.0062	0.0041	0.0020

Sur la matrice du DM, on obtient ce tableau d'erreurs MSE pour différentes valeurs de p :

Eigentaste et quelques autres ne sont pas encore implémentés. à venir aussi : une autre mesure d'erreur, et des mesures sur les autres jeux de données.
L'algorithme simple "BiasFromMean" obtient d'excellents résultats ! Les variantes de Slope One obtiennent aussi des résultats très comparables aux algorithmes SVD plus complexes à mettre en œuvre.

2.3 Temps d'exécution



Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
tests	1	54.211 s	0.241 s	
tests>@(algo)MSE(M,probas(i),iter,algo)	44	53.897 s	0.003 s	
MSE	44	53.894 s	0.741 s	
unbias	48	50.538 s	50.531 s	
unbiasedSVD	12	12.913 s	0.001 s	
UnbiasedWitness	12	12.794 s	0.009 s	
tests>ItemCosSimExt	12	12.750 s	0.001 s	
itemCosSimilarity	12	12.749 s	0.204 s	
userCosSimilarity	12	12.519 s	0.022 s	
tests>UserCosSimExt	12	12.519 s	0.000 s	
tests>WeightedSlopeOneExt	12	0.533 s	0.015 s	
Dev	24	0.532 s	0.532 s	
WeightedSlopeOne	12	0.518 s	0.257 s	
truncateSVD	36	0.495 s	0.495 s	
rank	44	0.436 s	0.436 s	
tests>SlopeOneExt	12	0.431 s	0.001 s	
SlopeOne	12	0.430 s	0.051 s	
randObserve	132	0.300 s	0.300 s	
plainSVD	12	0.189 s	0.013 s	
shiftSVD	12	0.184 s	0.014 s	

FIGURE 1 – Profiling fourni par Matlab

Les algorithmes qui enlèvent et remettent les biais comme dans le cours s'exécutent beaucoup plus lentement : il faut inverser une matrice creuse, et on n'a pas réussi à vectoriser une partie du calcul de l'argmin pour les biais. Il faudra essayer l'expression plus naïve des biais et voir ce qu'on gagne en temps/perd en précision.
Il faut aussi tenir compte du fait que certains algos (Slope One oui, SVD non) calculent des choses "une fois pour toutes", et c'est rapide d'ajouter un utilisateur et/ou de faire une requête.

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.