Atelier SD2: Recommandation par Filtrage Collaboratif (MovieLens)

Résumé

Introduction au principe des systèmes de recommandation par filtrage collaboratif pour le commerce en ligne. Recherche de facteurs latents par factorisation non négative de matrice (NMF) ; complétion de grande matrice creuse. Explicitation d'une solution avec la librairie MLlib de Spark sur la base de données movieLens.

1 Introduction

1.1 Marketing et systèmes de recommandation

Gestion de la relation client

La rapide expansion des sites de commerce en ligne a pour conséquence une explosion des besoins en marketing et *gestion de la relation client* (GRC ou CRM : *client relationship management*) spécifiques à ce type de média. C'est même le domaine qui est le principal fournisseur de données massives ou tout du moins la partie la plus visible de l'iceberg.

La GRC en marketing quantitatif traditionnel est principalement basée sur la construction de modèles de scores : d'appétence pour un produit, d'attrition (*churn*) ou risque de rompre un contrat. Voir à ce propos les scénarios d'appétence pour la carte Visa Premier ou celui concernant un produit d'assurance vie à partir de l'enquête INSEE sur le patrimoine des français.

Commerce en ligne

Le commerce en ligne introduit de nouveaux enjeux avec la construction d'algorithmes de *filtrage* : sélection et recommandation automatique d'articles, appelés encore *systèmes de recommandation* [1]. Certains concernent des méthodes *adaptatives* qui suivent la navigation de l'internaute, son flux de clics, jusqu'à l'achat ou non. Ces approches sont basées sur des algorithmes de bandits manchots et ne font pas l'objet de cet atelier. D'autres stratégies sont définies à partir d'un historique des comportements des clients, d'informations complémentaires sur leur profil, elles rejoignent les méthodes traditionnelles de marketing quantitatif. D'autres enfin sont basées sur la seule connaissance des interactions clients $\times$ produits à savoir la présence / absence d'achats ou un ensemble d'avis recueillis sous la forme de notes d'appréciation de chaque produit consommé. On parle alors de filtrage collaboratif (*collaborative filtering*).

Filtrage collaboratif

Ce dernier cas a largement été popularisé par le concours Netflix où il s'agit de proposer un film à un client en considérant seulement la matrice très creuse : clients $\times$ films, des notes sur une échelle de 1 à 5. L'objectif est donc de prévoir le goût, la note, ou l'appétence d'un client pour un produit (livre, film...), qu'il n'a pas acheté, afin de lui proposer celui le plus susceptible de répondre à ses attentes. Tous les sites marchands : Amazon, Fnac, Netflix... implémentent de tels algorithmes.

Le filtrage collaboratif basé sur les seules interactions client $\times$ produits : présence / absence d'achat ou note d'appréciation fait généralement appel à deux grandes familles de méthodes :

Méthodes de voisinage fondés sur des indices de similarité (corrélation linéaire ou des rangs de Spearman...) entre clients ou (exclusif) entre produits :

- Basé sur le client avec l'hypothèse que des clients qui ont des préférences similaires vont apprécier des produits de façon similaire. Trouver un sous-ensemble S_i de clients qui notent de manière similaire au client i ; prévoir la note manquante : produit j par client i, par combinaison linéaire des notes de ce sous-ensemble S_i sur le produit j.
- Basé sur le produit avec l'hypothèse que les clients préfèreront des produits similaires à ceux qu'ils ont déjà bien notés. Trouver un sous-ensemble S_j de produits notés de façons similaires par le client i ; prévoir la note manquante : produit j par client i, par combinaison linéaire des notes de ce sous-ensemble S_j du client i.

Modèle à facteurs latents basé sur une décomposition de faible rang avec une éventuelle contrainte de régularisation, de la matrice très creuse des

1. Le site "PodcastScience" donne une introduction assez complète de ce thème.

notes ou avis clients $\times$ produits. La note du client i sur le produit j est approchée par le produit scalaire de deux facteurs $\mathbf{W}_i$ et $\mathbf{H}_j$.

La littérature est très abondante sur le sujet qui soulève plusieurs problèmes dont :

- comment évaluer un système de recommandation ?
- Comment l'initier (*cold start problem*) ? C'est-à-dire comment initier une matrice très creuse avec très peu d'avis ou introduire de nouveaux clients ou produits .

Par ailleurs, Des systèmes hybrides intègrent ces donnés d'interaction avec d'autres informations sur le profil des clients (variables âge, sexe, prénom...) ou encore sur sur la typologie des produits (variables genre, année...).

1.2 Modèles à facteurs latents

La recherche de facteurs latents est basée sur la décomposition en valeurs singulière d'une matrice (SVD) ou la factorisation d'une matrice non négative plus adaptée à des notes d'appréciation. Ces matrices sont généralement très creuses, souvent à peine 2% de valeurs connues sont renseignées et les autres qui ne peuvent être mises à zéro, sont donc des valeurs *manquantes*. il s'agit alors d'un problème de complétion de matrice [2].

Le scénario de présentation des méthodes de factorisation de matrices (SVD, NMF) et complétion introduit ce type d'approche sur un exemple "jouet" à partir d'une matrice de dénombrements de ventes. La prise en compte d'une matrice de notes avec des "0" correspondant à des données manquantes limite les possibilités.

2 Déroulement de l'atelier

2.1 Objectifs

Le scénario précédent est donc un préalable à cet atelier qui vise une "industrialisation" de la démarche avec passage à l'échelle "volume".

L'objectif principal est de produire un *système de recommandation efficace sur des données massives*, à savoir une très grande matrice creuse.

2. Le site d'Igor Carron donne un aperçu assez réaliste de la complexité et du foisonnement de la recherche sur ce thème.

2.2 Ressources

Pédagogiques

Plusieurs tutoriels sont disponibles :

- Initiation à R, Python et les librairies pandas et scikit-learn.
- Présentation de la NMF.
- Introduction à Spark et MLlib qui implémente un algorithme de factorisation (ALS).

Matériel

En fonction du contexte et des accords de partenariat : ordinateur individuel multi-cœurs, cluster...

2.3 Les données

Des données réalistes croisant plusieurs milliers de clients et films, sont accessibles en ligne. Il s'agit d'une extraction du site movielens qui vous "aide" à choisir un film. Entre autres, quatre tailles de matrices creuses sont proposées :

100k 100 000 évaluations de 1000 utilisateurs de 1700 films.

1M Un million d'évaluations par 6000 utilisateurs sur 4000 films.

10M Dix millions d'évaluation par 72 000 utilisateurs sur 10 000 fims.

20M Vingt millions d'évaluations par 138 000 utilisateurs sur 27 000 films.

3 Éléments de solutions

3.1 Avec R

Adapter le scénario précédent en prenant en compte les propriétés de grande parcimonie de la matrice des notes. Utiliser la librairie `Matrix` qui gère des classes de matrices creuses et la librairie `softImpute` d'imputation des valeurs d'une matrice.

Jusqu'à quelle taille de matrice MovieLens cette stratégie est-elle opérationnelle ?

3.2 Python

Python gère des matrices denses et aussi la classe `scipy.sparse` des matrices creuses. La SVD est bien documentée en Python et présente dans plusieurs librairies.

La version tronquée de SVD de `scikit_learn` reconnaît la classe `scipy.sparse` contrairement à la fonction PCA qui réalise l'analyse en composantes principales.

La factorisation non négative de matrices (NMF), ou plutôt un algorithme spécifique (projection du gradient), est accessible dans la librairie `scikit_learn` mais uniquement pour matrices denses.

Le site de Jonny Edwards donne accès à un calepin IPython dont le source est également disponible. Celui-ci utilise la librairie nimfa de NMF qui propose de très nombreuses options d'algorithmes (12) et d'initialisation. La librairie `nimfa` peut être installée à partir du *package manager* de Canopy. Il n'est pas grave que ce ne soit pas la toute dernière version. Cette librairie reconnaît la classe `scipy.sparse`.

Cependant aucun de ces algorithmes ne prennent en compte la complétion de matrice. Appliqués à une matrices de notes, ils vont reconstruire majoritairement les 0. Ils ne peuvent conduire à de bons résultats.

3.3 MLlib de Spark

Un exemple de solution de complétion de très grande matrice est proposé ici et ici avec des exemples d'application aux données MovieLens.

MLlib implémente l'algorithme NMF par ALS (alternative least square) avec deux options. Dans la première, la fonction objectif est une norme L_2 (Froebenius) entre la matrice et sa reconstruction mais cette norme n'est calculée *que* sur les valeurs (notes) observées. Des contraintes (L_1 et L_2) introduisent une régularisation pour renforcer la parcimonie de la factorisation qui opère donc une complétion. Les paramètres sont à optimiser par validation croisée. La documentation n'est pas explicite mais cite Koren et al. (2009) qui sont plus précis. La deuxième option (implicit) traite en principe des matrices archivant des nombres de note ou de chargements d'un film ou...

Le calepin écrit en pyspark exécute les instructions pour opérer cette version de factorisation sur le plus petit fichier des données MovieLens.

Vérifier son bon fonctionnement. L'exécuter sur les jeux de données plus conséquents en fonction des ressources matérielles accessibles ;

4 Conclusion

Faire le bilan des méthodes, algorithmes et implémentations disponibles pour mettre en œuvre un système de recommandation, par filtrage collaboratif et recherche de facteurs latents, en disposant d'une matrice nécessairement très creuse, soit d'effectifs de ventes, soit de notes d'appréciation.

Plus précisément, que dire de l'usage respectif de Spark/MLlib et R ?