

Systemes de recommandation

1e partie

Raphaël Fournier-S'niehotta

CNAM Paris, fournier@cnam.fr

HTT-FOD
RCP217
2020-2021

le **cnam**

Plan

1 | Introduction

2 | Filtrage collaboratif par le voisinage

- 1 – Filtrage collaboratif *user-based*
- 2 – Filtrage collaboratif *item-based*
- 3 – Complexité, efficacité algorithmique
- 4 – Comparaison entre *item-* et *user-based CF*
- 5 – Vision "régression" du filtrage par voisinage
- 6 – Approches orientées graphes pour le filtrage par voisinage

3 | Filtrage collaboratif par modèle

- 1 – Introduction
- 2 – Modèles à facteurs latents

Introduction

Définition

Un *système de recommandation* propose des éléments (*items*) d'un catalogue à des utilisateurs (*users*).

- Les utilisateurs·ices sont presque toujours des personnes.
- Les éléments peuvent être d'une nature commerciale (produits à vendre), culturelle (films, chansons, presse), ou professionnelle (articles scientifiques).

Quelques exemples

```

nnn nnnnnnn oo0000 nnnnnnnn nnnn nnnnnnnn nnnnnn nnnnnnnn nnnnnnnn nnnnnn
File Edit Apps Options Buffers Tools Article Threads Misc Post Score Mascrypt Help
[Icons]
| ***** | [ 30: The Ripper ] Popeye's Famous Fried Chicken
| *** | [ 55: Art Poe ] Quiche Lorraine
| ***** | [ 123: Art Poe ] COLLECTION (4) Persimmon Desserts
| *** | [ 97: Art Poe ] COLLECTION (2) Brioche
| NA | [ 58: Art Poe ] Corn Tortillas
| NA | [ 46: Art Poe ] Flour Tortillas
| ** | [ 40: Robyn Walton ] *Czechoslovakian Cabbage Soup
| ** | [ 57: Robyn Walton ] Collection (2) Rice Pudding
<-- Gnus rec.food.recipes/17026 (143 more) 2:48pm (Summary)
From: The Ripper <ripper@Onramp.NET>
Subject: Popeye's Famous Fried Chicken
Newsgroups: rec.food.recipes
Date: 1 Jan 1996 07:28:51 -0700
Organization: Onramp
Reply-To: The Ripper <ripper@Onramp.NET>
Followup-To: rec.food.cooking

>From the book written by Todd Wilbur.

6 cups vegetable oil
2/3 cup all-purpose flour
1 tbls salt
2 tbls white pepper
1 tsp cayenne pepper
2 tsp paprika
3 eggs
1 frying chicken w/skin, cut up

Heat the oil over medium heat in a deep fryer or in a wide, deep pan
on the stove. In a large, shallow bowl, combine the flour, salt, pepper,
and paprika. Break the eggs into a separate shallow bowl and beat
until blended. Check the oil by dropping in a pinch of the flour mixture.
If the oil bubbles rapidly around the flour, it is ready. Dip each piece
of chicken into the eggs, then coat generously with the flour mixture.
Drop each piece into the hot oil and fry for 15 to 25 minutes, or until
it is a dark golden brown. Remove the chicken to paper towels or a rack
to drain.
-- Gnus rev.food.recipes/17026 Popeye's Famous Fried Chicken:1:48pm (Article

```


Usenet news.
 Groupe de recherche
 GroupLens, à l'origine
 de datasets
 importants dans la
 communauté et de
 nombreux
 algorithmes.

Quelques exemples : Amazon

← ↻ 🏠 🔒 https://www.amazon.fr/Recommender-Systems-Textbook-Charu-Aggarwal/dp/3319296574/ref=sr_1_1?_mk_fr_FR=AMAZ


Toutes Ventes Flash Chez Raphaël Coupons Beauté Amazon Basics Sports & Activités en plein air Santé et produits d'hygiène Vendez sur Amazon Amazon Assistant - Offre

Livres anglais et étrangers Recherche détaillée Nos rubriques Nouveautés Meilleures ventes Bonnes affaires Livres audio Tous les livres Vendez !

 **Obtenez 10% de réduction sur votre premier achat effectué dans l'application Amazon**

Recommender Systems: The Textbook (English Edition) et plus de huit millions d'autres livres sont disponibles pour le **Kindle d'Amazon**. En savoir plus

Retour aux résultats



Recommender Systems: The Textbook Relié – 4 avril 2016
Édition en Anglais de Charu C. Aggarwal (Auteur)

★★★★☆ 40 évaluations

Voir les formats et éditions

| | | |
|--------------------------|-------------------------|-------------------|
| Format Kindle 42,64 € | Relié 60,92 € | Broché 62,33 € |
|--------------------------|-------------------------|-------------------|

Lisez avec notre **Appli gratuite**

1 D'occasion à partir de 56,06 €
6 Neuf à partir de 60,92 €

4 Neuf à partir de 62,33 €

This book comprehensively covers the topic of recommender systems, which provide personalized recommendations of products or services to users based on their previous searches or purchases. Recommender system methods have been adapted to diverse applications including query log mining, social networking, news recommendations, and computational advertising. This book synthesizes both fundamental and advanced topics of a research area that has now reached maturity. The chapters of this book are organized into three categories:

Algorithms and evaluation: These chapters discuss the fundamental algorithms in

En lire plus

Signaler des informations incorrectes sur les produits

Neuf: **60,92 €**
Tous les prix incluent la TVA.

Livraison à partir de 0,01 € - **jeudi 22 av.** en France métropolitaine.
Détails

Livraison accélérée : **mercredi 21 av.**
Commandez dans les 10 h et 3 mins
Détails

Livrer à Fournier... - Villejuf 94800

Il ne reste plus que 15 exemplaire(s) en stock (d'autres exemplaires sont en cours d'acheminement).

Disponible au format Kindle. Les eBooks Kindle peuvent être lus sur n'importe quel appareil avec l'appli gratuite Kindle.

Quantité: 1

Ajouter au panier

Acheter cet article


Transaction sécurisée

Expédié et vendu par Amazon.


Livré en 1 jour ouvré

Profitez de tous les avantages de livraison en vous inscrivant à **Amazon Prime**

Les clients ayant acheté cet article ont également acheté



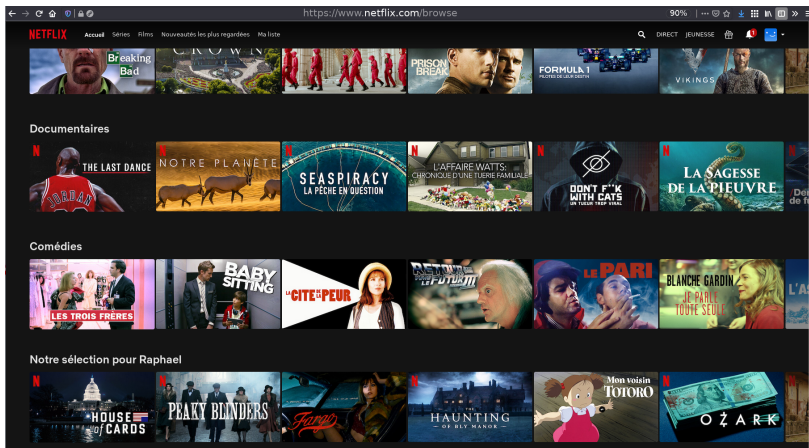
Practical Recommender Systems
Kim Falk



Hands-On Machine Learning with Scikit-Learn, Keras, and ...

Pionniers dans le domaine commercial, échelle 1-5, explications, implicit/explicit

Quelques exemples : Netflix



Explications détaillées, Netflix Prize.

Quelques exemples : Google Actualités (news)

☰ Google Actualités

🔍 Rechercher des sujets, des lieux et des sources

🌐 À la une

👤 Pour vous

☆ Favoris

🔍 Recherches enregistrées

🛡️ COVID-19

🇫🇷 France

🌍 International

📍 Vos actualités locales

📊 Economie

🏭 Sciences et technologies

📺 Divertissement

🚴 Sports

🏠 Santé

Langue et zone géographique
Français (France)

Paramètres

🗨️ Comment les articles sont-ils classés ?

Pour vous

Recommandé en fonction de vos centres d'intérêt

La chance historique de Maxime Vachier-Lagrave au Tournoi des Candidats

L'Équipe.fr · Il y a 17 heures

- Kevin Bordi : «Maxime Vachier-Lagrave gagnant du tournoi des candidats ? Ce serait un aboutissement pour les échecs en France»

CNEWS · Il y a 1 heure

[Voir la couverture complète](#)



Serie A : trois clubs demandent l'exclusion de la Juve, l'Inter et l'AC Milan !

Foot Mercato · Il y a 1 heure

- Trois clubs de Serie A veulent l'exclusion de la Juve, du Milan et de l'Inter

SO FOOT · Il y a 3 heures

[Voir la couverture complète](#)

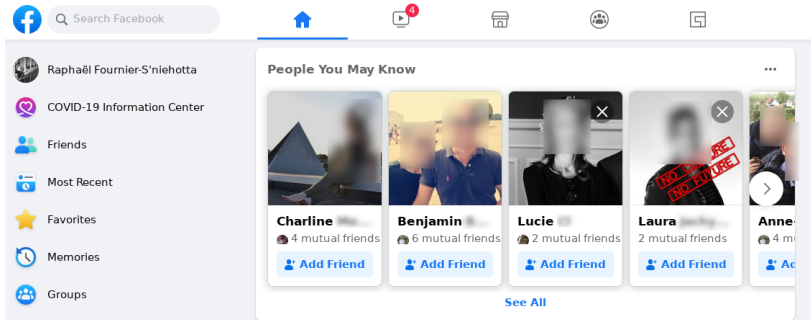


Purificateur d'air Eclipse : Rowenta voudrait-il faire de l'ombre à Dyson ?

Les Numériques · Il y a 2 jours



Quelques exemples : Facebook



The image shows a screenshot of a Facebook profile page. The top navigation bar includes the Facebook logo, a search bar, and navigation icons for Home, Notifications (with a red '4'), Pages, Groups, and Marketplace. The left sidebar lists navigation options: Raphaël Fournier-S'niehotta (profile), COVID-19 Information Center, Friends, Most Recent, Favorites, Memories, and Groups. The main content area is titled "People You May Know" and displays a grid of five suggested profiles. Each profile card shows a blurred profile picture, the name, the number of mutual friends, and an "Add Friend" button. The profiles are: Charline (4 mutual friends), Benjamin (6 mutual friends), Lucie (2 mutual friends), Laura (2 mutual friends), and Anne (4 mutual friends). A red stamp with the text "NO COPY, NO FUTURE" is overlaid on the profile picture of Laura. A "See All" link is located at the bottom of the suggestions grid.

Vise l'accroissement de la taille du réseau (pas directement des "ventes")

Champs d'application

| | |
|-------------|--------------------|
| Amazon | Livres + ... |
| Netflix | VoD |
| Jester | Blagues |
| Google News | Articles de presse |
| Pandora | Musique |
| TripAdvisor | Voyages |
| Youtube | Vidéos en ligne |

Motivation

- "Ces systèmes guident les gens vers des contenus intéressants grâce aux recommandations d'autres personnes." (Paul Resnick)

Quels problèmes cherche-t-on à résoudre ?

- infobésité
- longue traîne
- expérience utilisateur
- reproduction d'un modèle hors ligne

Interdisciplinarité

Le problème de la recommandation est proche de plusieurs disciplines :

- psychologie (expression du besoin, des goûts)
- marketing (accroissement des ventes)
- design (interfaces, graphiques ou non)
- statistiques (probabilités, modélisation)
- informatique (passage à l'échelle)

Modélisation du problème

- De très nombreuses approches existent!
- Les deux plus importantes : "prédiction" et "classement"

Prédiction

- On cherche à prédire l'évaluation qu'un utilisateur donnerait à un élément.
- Avec n items et m users, on a une matrice $m \times n$
- les données d'apprentissage sont les cases remplies,
- on cherche à remplir les cases vides

Classement

- On ne cherche plus l'évaluation d'un élément par une personne
- On détermine les k éléments les plus intéressants (**top- k**)
- Seul l'ordre compte (pas les écarts entre éléments)
- On peut utiliser une modélisation "prédiction" puis trier, pour faire un classement

Modélisation

| | i1 | i2 | i3 | i4 |
|----|----|----|----|----|
| u1 | | 3 | 5 | 2 |
| u2 | 1 | 5 | 2 | 4 |
| u3 | | 3 | | |
| u4 | | | 5 | 2 |
| u5 | | 1 | 4 | |

| | i1 | i2 | i3 | i4 |
|----|----------|----------|----------|----------|
| u1 | 1 | 3 | 5 | 2 |
| u2 | 1 | 5 | 2 | 4 |
| u3 | 2 | 3 | 1 | 4 |
| u4 | 1 | 4 | 5 | 2 |
| u5 | 2 | 1 | 4 | 3 |

u1

i1 1

u3

i4 4
i1 2
i3 1

| | i1 | i2 | i3 | i4 |
|----|----|----|----|----|
| u1 | | 1 | 1 | 1 |
| u2 | 1 | 1 | 1 | 1 |
| u3 | | 1 | | |
| u4 | | | 1 | 1 |
| u5 | | 1 | 1 | |

| | i1 | i2 | i3 | i4 |
|----|----------|----------|----------|----------|
| u1 | 0 | 1 | 1 | 1 |
| u2 | 1 | 1 | 1 | 1 |
| u3 | 0 | 1 | 1 | 1 |
| u4 | 0 | 0 | 1 | 1 |
| u5 | 1 | 1 | 1 | 0 |

u3

i3
i4

u5

i1

Objectifs

- Objectif premier : accroissement des ventes (contexte commercial)
- Manières plus ou moins directes!
 - Facilité d'accès aux contenus du système
 - Améliorer la pertinence de ce que l'utilisateur voit
 - Nouveauté
 - Surprise (légèrement différent)
 - Diversité

Défis

- Taille des matrices
- Matrices creuses
- Diversité des objectifs à satisfaire

Données : échelles de valeurs

- Le système de recommandation que l'on *peut* mettre en place dépend beaucoup des données dont on dispose.
- Les évaluations (*ratings*) sont souvent sur une échelle de valeurs
- Parfois continue, le plus souvent discrète
- Valeurs négatives parfois autorisées ($\{-2, -1, 0, 1, 2\}, [-10; 10]$)
- Systèmes les plus courants : échelle de 1 à 5, ou 1 à 10
- Interprétation variable, échelles équilibrées (o/n), élément neutre (o/n)

- Valeurs catégorielles ordonnées
- Binarité 0-1
- Unarité

Données : réactions explicites et implicites

| | GLADITOR | GODFATHER | BEN-HUR | GOODFELLAS | SCARFACE | SPARTACUS |
|-------|----------|-----------|---------|------------|----------|-----------|
| U_1 | 1 | | | 5 | | 2 |
| U_2 | | 5 | | | 4 | |
| U_3 | 5 | 3 | | 1 | | |
| U_4 | | | 3 | | | 4 |
| U_5 | | | | 3 | 5 | |
| U_6 | 5 | | 4 | | | |

(a) Ordered ratings

| | GLADIATOR | GODFATHER | BEN-HUR | GOODFELLAS | SCARFACE | SPARTACUS |
|-------|-----------|-----------|---------|------------|----------|-----------|
| U_1 | 1 | | | 1 | | 1 |
| U_2 | | 1 | | | 1 | |
| U_3 | 1 | 1 | | 1 | | |
| U_4 | | | 1 | | | 1 |
| U_5 | | | | 1 | 1 | |
| U_6 | 1 | | 1 | | | |

(b) Unary ratings

L'expressivité de la 2e est plus faible, on ne sait pas que U_1 n'aime pas Gladiator. La 2e permet seulement d'exprimer des préférences "positives"

Données et sur-apprentissage

- Données unaires, implicit feedback
 - Analyse simplifiée en identifiant 0 et données manquantes
 - Sans ça, sur-apprentissage (pas assez de valeurs distinctes)
 - Biais introduit, mais assez faible (contexte : la plupart des choses ne seront pas vues)
 - Ces données sont en général très faciles à collecter!
 - Nombreuses sources (achat, visite de page, *actions*)
- Données explicites, échelles plus grandes
 - analyse non simplifiée en identifiant 0, biais plus fort
 - Données plus délicates à collecter : contribution volontaire de l'utilisateur
 - Incitations explicites (questions, récompenses, etc.)
 - Incitations implicites (design d'interface)

Données : critiques

- Généralisation des systèmes de recommandation ces dernières années
- La dimension humaine est parfois oubliée
- "Weapons of Math Destruction" (Cathy O'Neil, 2016)

- Attention à ce qu'on note
- BlackMirror, Uberisation
- Crédit social, assurances, droit à l'oubli

Les modèles de RS

Il y a 4 grandes familles de systèmes de recommandation :

1. les modèles de **filtrage collaboratif**, contenant :
 - les méthodes à mémoire, aussi appelés "Par le voisinage" (*neighbour-based collaborative filtering*)
 - les méthodes à modèle (*model-based collaborative filtering*)
2. le filtrage par le contenu (*content-based methods*)
3. les méthodes reposant sur des connaissances externes (*knowledge-based methods*)
4. les systèmes hybrides, combinant des approches précédentes (*ensemble/hybrid methods*)

Filtrage collaboratif par le voisinage

Filtrage collaboratif

- Par le voisinage (*neighbour-based collaborative filtering*)
 - User-based
 - Item-based
- Par modèle (*model-based collaborative filtering*)

User-based

- On cherche des utilisateurs *similaires* à A
- on recommande des éléments à A en faisant une moyenne pondérée des notes de ces utilisateurs.
- On se limite en général à k utilisateurs.
- Les fonctions de similarité sont à calculer sur les lignes de la matrice des notes.

Item-based

- Pour connaître la note de A sur l'item I, on cherche un ensemble d'items similaires à I, que A a noté, et on calcule une moyenne pondérée.
- Les similarités sont calculées sur les colonnes.

Filtrage collaboratif par le voisinage

- Le principe élémentaire : on peut trouver des relations (dépendances) dans ce qui se passe au niveau utilisateur et item
- Un utilisateur intéressé par un documentaire historique a plus de chance d'être intéressé par un autre documentaire historique ou des programmes éducatifs qu'un film d'action
- Des catégories d'éléments présentent des corrélations significatives, que l'on peut utiliser pour faire des recommandations plus précises
- Cela peut aussi se jouer à une granularité plus fine que les catégories
- On présuppose que l'on peut apprendre ces relations, grâce aux données
- Plus on a de données, plus les recommandations seront robustes
- Les modèles peuvent identifier des catégories d'utilisateurs aux comportements de consommation similaires

Avantages, inconvénients

Avantages

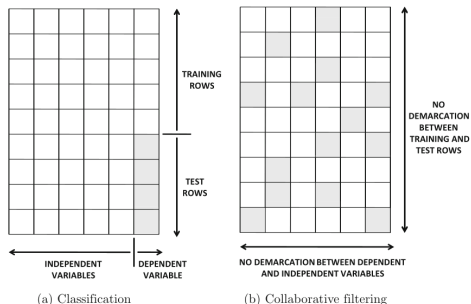
- très simple à implémenter
- recommandations faciles à expliquer

Inconvénients

- peu efficace avec matrice très creuses
- trop peu d'utilisateurs ou d'items similaires : *couverture* faible
- souvent, seuls les top- k éléments suffisent, ce n'est pas toujours un gros problème

Disciplines proches

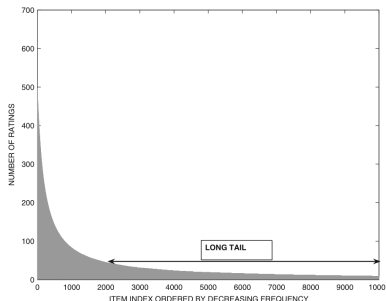
- Analyse de données manquantes
 - cas difficile, large, sparse
 - Classification
-
- toute colonne peut avoir des données manquantes
 - pas de distinction entre class variable et feature variables, chaque caractéristique est une variable dépendante et indépendante.



Inductif, transductif

- Dans un contexte *transductif*, les données de test sont incluses dans les données d'entraînement
- Il est souvent dur de faire des prédictions pour des données de test non présentes lors de l'entraînement
- Les modèles capables de faire des prédictions pour de nouvelles instances sont dits *inductifs* (naïve Bayes)
- L'enchevêtrement des données dans la matrice en recommandation fait qu'on est en transductif

Longue traîne et matrices



- Peu d'intérêt marchand des high freq items
- Recommander la long tail
- Difficulté d'avoir des notes, suggestion d'items populaires
- Baisse de diversité
- Difficulté de calculer des voisinages fiables

Plan

2 | Filtrage collaboratif par le voisinage

1 – Filtrage collaboratif *user-based*

2 – Filtrage collaboratif *item-based*

3 – Complexité, efficacité algorithmique

4 – Comparaison entre *item-* et *user-based* CF

5 – Vision "régression" du filtrage par voisinage

6 – Approches orientées graphes pour le filtrage par voisinage

User-based CF

Principe : les utilisateurs similaires ont des évaluations similaires pour des éléments donnés.

- On cherche un *voisinage* d'utilisateurs similaires à l'utilisateur cible
- On calcule la similarité entre cet utilisateur et tous les autres
- on en garde quelques uns

Difficile car :

- différentes manières de noter : certains notent ce qu'ils aiment (surtout), d'autres ce qu'ils n'aiment pas
- il faut que les ensembles notés coïncident

Notation pour la matrice

Soit $R = r_{uj}$ la matrice $m \times n$.

I_u est l'ensemble des indices des items notés par u .

L'ensemble noté par u et v est : $I_u \cap I_v$.

Exemple :

$I_u = \{1, 3, 5\}$, $I_v = \{1, 2, 3, 4\}$, $I_u \cap I_v = \{1, 3\}$.

Similarité : coefficient de corrélation de Pearson (PCC)

- On calcule d'abord la moyenne des notes de u , μ_u :

$$\mu_u = \frac{\sum_{k \in I_u} r_{uk}}{|I_u|} \quad \forall u \in \{1 \dots m\}$$

- Puis la similarité elle-même :

$$\text{Sim}(u,v) = \frac{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} (r_{uk} - \mu_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} (r_{vk} - \mu_v)^2}} \quad \forall (u,v) \in \{1 \dots m\}^2$$

- Normalement, PCC calculé avec des μ seulement sur les items de $I_u \cap I_v$.
- Simplification calculatoire et programmatique possible
- On pourrait garder le top- k des utilisateurs avec le meilleur PCC
- Mais nombre très variable de ratings pour un élément donné
- On cherche les top- k pour chaque élément

Étape 2 : calcul de la note

- Manière simple : on fait une moyenne pondérée des notes des utilisateurs
- Éventuellement les k plus proches
- Poids : les PCC entre utilisateurs
- Problème : les échelles de notes individuelles
- Solution : évaluations centrées

$$s_{uj} = r_{uj} - \mu_u \quad \forall u \in \{1 \dots m\}$$

- On calcule une évaluation pondérée sur ces notes (positive ou négative)
- La note finale prédite \hat{r}_{uj} est obtenue en rajoutant la moyenne :

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P(j)} \text{Sim}(u,v) \cdot s_{uj}}{\sum_{v \in P(j)} |\text{Sim}(u,v)|}$$

Exemple UBCF

On cherche \hat{r}_{31} et \hat{r}_{36} dans l'exemple suivant :

| Item-Id ⇒ User-Id ↓ | 1 | 2 | 3 | 4 | 5 | 6 | Mean Rating | Cosine(<i>i</i> , 3) (user-user) | Pearson(<i>i</i> , 3) (user-user) |
|------------------------|---|---|---|---|---|---|----------------|--------------------------------------|---------------------------------------|
| 1 | 7 | 6 | 7 | 4 | 5 | 4 | 5.5 | 0.956 | 0.894 |
| 2 | 6 | 7 | ? | 4 | 3 | 4 | 4.8 | 0.981 | 0.939 |
| 3 | ? | 3 | 3 | 1 | 1 | ? | 2 | 1.0 | 1.0 |
| 4 | 1 | 2 | 2 | 3 | 3 | 4 | 2.5 | 0.789 | -1.0 |
| 5 | 1 | ? | 1 | 2 | 3 | 3 | 2 | 0.645 | -0.817 |

$$\text{Cosine}(1, 3) = \frac{6 * 3 + 7 * 3 + 4 * 1 + 5 * 1}{\sqrt{6^2 + 7^2 + 4^2 + 5^2} \cdot \sqrt{3^2 + 3^2 + 1^2 + 1^2}} = 0.956$$

$$\begin{aligned} \text{Pearson}(1, 3) &= \\ &= \frac{(6 - 5.5) * (3 - 2) + (7 - 5.5) * (3 - 2) + (4 - 5.5) * (1 - 2) + (5 - 5.5) * (1 - 2)}{\sqrt{1.5^2 + 1.5^2 + (-1.5)^2 + (-0.5)^2} \cdot \sqrt{1^2 + 1^2 + (-1)^2 + (-1)^2}} \\ &= 0.894 \end{aligned}$$

- PCC plus discriminant que Cosinus
- Le signe peut être informatif : (dis)similarité
- top-2 utilisateurs proches de u_3 : u_1 et u_2

Exemple UBCF

$$\hat{r}_{31} = \frac{7 * 0.894 + 6 * 0.939}{0.894 + 0.939} \approx 6.49$$

$$\hat{r}_{36} = \frac{4 * 0.894 + 4 * 0.939}{0.894 + 0.939} = 4$$

- On privilégierait donc l'item 1, plutôt que le 6
- les prédictions suggèrent que u_3 aimera 1 et 6 plus que *tous* les autres.
- attention : biais, $\{u_1, u_2\}$ est un groupe optimiste.

- regardons en centrant :

$$\hat{r}_{31} = 2 + \frac{1.5 * 0.894 + 1.2 * 0.939}{0.894 + 0.939} \approx 3.35$$

$$\hat{r}_{36} = 2 + \frac{-1.5 * 0.894 - 0.8 * 0.939}{0.894 + 0.939} \approx 0.86$$

- même résultat, 1 mieux que 6
- mais! $r_{36} = 0.86$, moins que les autres ratings
- u_1 et u_2 avaient aussi noté 6 moins que les autres

Attention : prédictions pour 6 éventuellement hors de l'intervalle des valeurs

UBCF : variations

Au lieu de centrer, on peut utiliser le z-score, reposant sur l'écart-type :

$$\sigma_u = \sqrt{\frac{\sum_{j \in I_u} (r_{uj} - \mu_u)^2}{|I_u| - 1}} \quad \forall u \in \{1 \dots m\}$$

Z-score :

$$z_{uj} = \frac{r_{uj} - \mu_u}{\sigma_u}$$

Et la note prédite se calcule ainsi :

$$\hat{r}_{uj} = \mu_u + \sigma_u \frac{\sum_{v \in P(j)} \text{Sim}(u,v) \cdot s_{uj}}{\sum_{v \in P(j)} |\text{Sim}(u,v)|}$$

- En général, si on utilise une fonction de normalisation pour les notes, il faut utiliser son inverse pour obtenir la prédiction finale. Ici, on re-multiplie par σ_u .
- La normalisation améliore la prédiction, mais il n'y a pas de consensus pour savoir si le z-score est meilleur que le centrage.
- Inconvénient du z-score : souvent hors de l'intervalle. On s'en sert pour classer.

Amplification des poids

- on amplifie parfois le poids des utilisateurs dans la moyenne pondérée

$$\text{Sim}(u,v) = \text{Pearson}(u,v)^\alpha \quad \text{avec } \alpha > 1$$

- On peut aussi filtrer le "voisinage" (*peer group*) d'un utilisateur de différentes façons
- on peut éliminer des notes faiblement ou négativement corrélées

Impact de la longue traîne

- Les éléments populaires apparaissent souvent dans les ensembles d'éléments en commun entre deux utilisateurs
- Pourtant, ils sont peu discriminants
- Recommandation de moindre qualité (sélection du voisinage, calcul de note)

Solution (cf *Information retrieval*) :

- Termes peu informatifs et fréquents (*stop-words*, "le", "un·e", "de", etc.) évacués avec Inverse Document Frequency (tf.idf)
- Ici : Inverse User Frequency. Pour chaque élément j , noté m_j fois, on définit :

$$w_j = \log\left(\frac{m}{m_j}\right) \quad \forall j \in \{1 \dots n\}$$

- Ce poids est introduit dans le PCC :

$$\text{Pearson}(u,v) = \frac{\sum_{k \in I_u \cap I_v} w_k (r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in I_u \cap I_v} w_k \cdot (r_{uk} - \mu_u)^2} \cdot \sqrt{\sum_{k \in I_u \cap I_v} w_k \cdot (r_{vk} - \mu_v)^2}}$$

Plan

2 | Filtrage collaboratif par le voisinage

1 – Filtrage collaboratif *user-based*

2 – Filtrage collaboratif *item-based*

3 – Complexité, efficacité algorithmique

4 – Comparaison entre *item-* et *user-based* CF

5 – Vision "régression" du filtrage par voisinage

6 – Approches orientées graphes pour le filtrage par voisinage

Item-based CF

- Les voisinages sont construits en termes d'items, et non d'utilisateurs
- Comme précédemment, on calcule des évaluations centrées s_{uj}
- On définit l'ensemble U_i des utilisateurs qui ont évalué i
- On calcule une similarité cosinus :

$$\text{cosA}(i, j) = \frac{\sum_{u \in U_i \cap U_j} s_{ui} \cdot s_{uj}}{\sqrt{\sum_{u \in U_i \cap U_j} s_{ui}^2} \cdot \sqrt{\sum_{u \in U_i \cap U_j} s_{uj}^2}}$$

- Un PCC entre colonnes pourrait aussi être utilisé, mais les résultats sont généralement moins bons

Item-based CF

- Pour prédire la note \hat{r}_{ut} de l'utilisateur u sur l'objet t , on calcule :
 - un ensemble $Q_t(u)$ contenant les top- k éléments
 - puis :

$$\hat{r}_{ut} = \frac{\sum_{j \in Q_t(u)} \text{cosA}(j, t) \cdot r_{uj}}{\sum_{j \in Q_t(u)} |\text{cosA}(j, t)|}$$

- On utilise les propres notes de l'utilisateur pour faire la prédiction
- Dans le cas de films, les top- k éléments seront souvent des films du même genre

Item-based CF

On cherche \hat{r}_{31} et \hat{r}_{36} :

| Item-Id ⇒ | 1 | 2 | 3 | 4 | 5 | 6 |
|-----------------------------|--------|--------|--------|--------|--------|--------|
| User-Id ↓ | | | | | | |
| 1 | 1.5 | 0.5 | 1.5 | -1.5 | -0.5 | -1.5 |
| 2 | 1.2 | 2.2 | ? | -0.8 | -1.8 | -0.8 |
| 3 | ? | 1 | 1 | -1 | -1 | ? |
| 4 | -1.5 | -0.5 | -0.5 | 0.5 | 0.5 | 1.5 |
| 5 | -1 | ? | -1 | 0 | 1 | 1 |
| Cosine(1, j) (item-item) | 1 | 0.735 | 0.912 | -0.848 | -0.813 | -0.990 |
| Cosine(6, j) (item-item) | -0.990 | -0.622 | -0.912 | 0.829 | 0.730 | 1 |

$$\text{AdjustedCosine}(1, 3) = \frac{1.5 * 1.5 + (-1.5) * (-0.5) + (-1) * (-1)}{\sqrt{1.5^2 + (-1.5)^2 + (-1)^2} \cdot \sqrt{1.5^2 + (-0.5)^2 + (-1)^2}} = 0.912$$

- les items 2 et 3 sont les plus proches de 1
- les items 4 et 5 sont les plus proches de 6

Item-based CF

On obtient :

$$\hat{r}_{31} = \frac{3 * 0.735 + 3 * 0.912}{0.735 + 0.912} = 3$$
$$\hat{r}_{36} = \frac{1 * 0.829 + 1 * 0.730}{0.829 + 0.730} = 1$$

- Évaluations dans l'intervalle et plus en phase avec les autres notes de l'utilisateur
- En général les listes de recommandations UBCF et IBCF sont proches

Plan

2 | Filtrage collaboratif par le voisinage

1 – Filtrage collaboratif *user-based*

2 – Filtrage collaboratif *item-based*

3 – Complexité, efficacité algorithmique

4 – Comparaison entre *item-* et *user-based* CF

5 – Vision "régression" du filtrage par voisinage

6 – Approches orientées graphes pour le filtrage par voisinage

Complexité

- Approche gloutonne : on calcule les notes de tous les éléments pour un utilisateur donné, puis on classe.
- Possible, mais coûteux
- Réutilisation de calculs intermédiaires

En pratique : une phase *offline* et une phase *online* :

- En *offline*, les similarités entre utilisateurs et voisinage sont pré-calculés
- En *online*, on calcule les notes et on classe

Complexité

- Soit $n' \ll n$ le nombre max d'évaluations d'items par les utilisateurs
- Soit $m' \ll m$ le nombre max d'évaluations reçues par un élément
- Calculer d'un voisinage en $\mathcal{O}(m \cdot n')$, de tous en $\mathcal{O}(m^2 \cdot n')$
- En item-based : $\mathcal{O}(n^2 \cdot m')$
- En online, chaque note est calculée en $\mathcal{O}(k)$, k taille du voisinage considéré
- Donc $\mathcal{O}(k \cdot n)$ pour toutes, avant classement (tri).

Plan

2 | Filtrage collaboratif par le voisinage

1 – Filtrage collaboratif *user-based*

2 – Filtrage collaboratif *item-based*

3 – Complexité, efficacité algorithmique

4 – Comparaison entre *item-* et *user-based* CF

5 – Vision "régression" du filtrage par voisinage

6 – Approches orientées graphes pour le filtrage par voisinage

Quelques éléments de comparaison

- Utiliser *les propres notes* de l'utilisateur est généralement meilleur que *les notes des autres utilisateurs similaires*
- Mais cela peut conduire à privilégier des éléments trop (ou toujours) similaires à ceux déjà vus : moins de diversité et de surprise!
- Stabilité meilleure en IB. Plus d'users que d'items, donc les items auront des ensembles d'users en commun plus larges. Et plus d'utilisateurs arrivent dans le système, donc moins de recalcul de voisinage.
- Explicabilité meilleure en item-based :

Because you watched "Secrets of the Wings," [the recommendations are] (List) .

Les clients ayant acheté cet article ont également acheté



- En user-based, on ne connaît pas les *utilisateurs similaires*

Avantages et inconvénients de ces méthodes

Avantages

- Facilité d'implémentation, de débogage
- Interprétabilité raisonnable (vs méthodes à modèle)
- Stabilité

Inconvénients

- Phase offline couteuse en temps et espace
- Couverture limitée (matrice creuse)

Réduction de dimension

- On cherche une représentation à base de facteurs latents dans un espace de basse dimension *latent factor models*
- une distance entre facteurs latents d'utilisateurs peut-être calculée, même avec peu d'items communs
- efficacité accrue pour les calculs de voisinages

Deux possibilités :

- réduction de dimension sur les lignes **ou** les colonnes de la matrice
- réduction sur les deux simultanément (*cf.* plus tard)
- Méthodes possibles : SVD ou ACP

Réduction de dimension : SVD

- Augmentation de valeurs manquantes de la matrice (avec les moyennes de ligne ou colonne) : on obtient une matrice R_f
- On calcule une matrice de similarité $S = R_f^T R_f$
- S est semi-définie positive, peut s'écrire :

$$S = P\Delta P^T$$

- Avec :
 - P est une matrice $n \times n$, avec les vecteurs propres de S (orthonormés)
 - Δ est une matrice diagonale avec les valeurs propres non nulles de S
- Soit P_d une matrice $n \times d$ contenant les vecteurs correspondant aux valeurs propres les plus grandes
- La représentation réduite de R_f est $R_f P_d$ ($m \times d$)
- Chaque utilisateur a une représentation en dimension d
- On l'utilise pour calculer les voisinages, avant calcul de prédictions.

Plan

2 | Filtrage collaboratif par le voisinage

1 – Filtrage collaboratif *user-based*

2 – Filtrage collaboratif *item-based*

3 – Complexité, efficacité algorithmique

4 – Comparaison entre *item-* et *user-based* CF

5 – Vision "régression" du filtrage par voisinage

6 – Approches orientées graphes pour le filtrage par voisinage

Régression

- Ces méthodes prédisent des notes comme étant des fonctions linéaires (des notes d'un éléments par les utilisateurs voisins, ou du même utilisateur pour des utilisateurs similaires)
- Vision régression : les coefficients de la régression sont
 - 0 pour les utilisateurs/items non liés
 - $\text{Sim}(u,v)$ pour les utilisateurs similaires
- peut-on apprendre ces poids?

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u(j)} \text{Sim}(u, v) \cdot (r_{vj} - \mu_v)}{\sum_{v \in P_u(j)} |\text{Sim}(u, v)|}$$

$$\hat{r}_{uj} = \mu_u + \sum_{v \in P_u(j)} w_{vu}^{user} \cdot (r_{vj} - \mu_v)$$

Régression user-based

- Attention : on définit un $P_u(j)$ un peu différemment, un ensemble de k utilisateurs proches de u , dont on ne retient que ceux qui ont des évaluations de j : $P_u(j) < k$. k doit être plus grand dans ce contexte
- L'idée : on agrège les différences entre les \hat{r}_{uj} et r_{uj} , pour créer une fonction objectif évaluant la qualité d'un ensemble de coefficients
- On cherche les w_{vu}^{user} minimisant l'erreur
- On somme les carrés d'erreurs sur tous les items d'un utilisateur

$$\text{Minimiser } J_u = \sum_{j \in I_u} (r_{uj} - \hat{r}_{uj})^2$$

- Comme les J_u sont sur des ensembles de variables w_{vu}^{user} disjoints, on peut sommer les J_u de différents utilisateurs sans changer la solution optimale :

$$\text{Minimiser } \sum_{u=1}^m J_u = \sum_{u=1}^m \sum_{j \in I_u} (r_{uj} - \hat{r}_{uj})^2$$

Régression user-based

- Peut être résolu avec des solveurs classiques
- Gestion de la régularisation, pour éviter le sur-apprentissage
- Ajout d'un terme

$$\lambda \sum_{j \in I_u} \sum_{v \in P_u(j)} (w_{vu}^{user})^2$$

- Pénalisation de grands coefficients
- Pas toujours suffisant
- Heuristiques non linéaire avec ajout de biais utilisateur ou item

$$\hat{r}_{uj} = b_u^{user} + b_j^{item} + \frac{\sum_{v \in P_u(j)} w_{vu}^{user} \cdot (r_{vj} - b_v^{user} - b_j^{item})}{\sqrt{|P_u(j)|}}$$

- Mêmes modèles en item-based, un peu moins performant mais plus efficace
- On peut combiner les deux

Plan

2 | Filtrage collaboratif par le voisinage

1 – Filtrage collaboratif *user-based*

2 – Filtrage collaboratif *item-based*

3 – Complexité, efficacité algorithmique

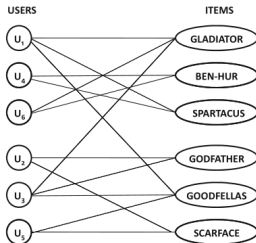
4 – Comparaison entre *item-* et *user-based* CF

5 – Vision "régression" du filtrage par voisinage

6 – Approches orientées graphes pour le filtrage par voisinage

Graphes

| | GLADIATOR | GODFATHER | BEN-HUR | GOODFELLAS | SCARFACE | SPARTACUS |
|-------|-----------|-----------|---------|------------|----------|-----------|
| U_1 | 1 | | | 5 | | 2 |
| U_2 | | 5 | | | 4 | |
| U_3 | 5 | 3 | | 1 | | |
| U_4 | | | 3 | | | 4 |
| U_5 | | | | 3 | 5 | |
| U_6 | 5 | | 4 | | | |



- Les graphes disposent d'outils et de concepts dédiés, que l'on peut mobiliser
- Variétés d'algorithmes différents

Graphes user-items

- Voisinages définis sans PCC, efficaces sur matrices creuses
- Graphe biparti $G = (U \cup I, E)$
- U utilisateurs, I items, E arêtes (u a noté j)

- Connectivité *indirecte*, voisinages par chemins (longueurs) ou marches aléatoires
- Problème lié à la notion de *prédiction de liens* en SNA

Marches aléatoires, chemins

Marches aléatoires

- Méthodes probabilistes, telles que le Personal PageRank ou SimRank
- Efficacité car connexité peut-être large dans un graphe

Chemins

- Affinité basée sur les nombres de chemins et leurs longueurs :
- Centralité de Katz

$$\text{Katz}(i, j) = \sum_{t=1}^{\infty} \beta^t n_{ij}^{(t)}$$

- $n_{ij}^{(t)}$ nombre de chemins entre i et j
- $\beta < 1$ paramètre
(inférieur à l'inverse de la plus grande v.p de A)

- La matrice des centralités Katz K se calcule :

$$K = \sum_{i=1}^{\infty} (\beta A)^i = (I - \beta A)^{-1} - I$$

- A peut être remplacé par une version "avec poids"

Filtrage collaboratif par modèle

Plan

3 | Filtrage collaboratif par modèle

1 – Introduction

2 – Modèles à facteurs latents

Filtrage collaboratif par modèle

- Dans le FC par voisinage, pas de modèle, les données orientent l'apprentissage
- Généralisation de *instance-based learning*
- Ici, on a un apprentissage (*training*), une construction du modèle
- Ensuite on a une phase distincte de prédiction
- Apprentissage *classique* : arbre de décision, règles, classifieur Bayes, réseaux de neurones
- La plupart ont été adaptés au contexte "recommandation"

La complétion de matrices généralise la classification "classique", avec deux différences :

- Pas de séparation claire entre variables dépendantes et indépendantes (colonnes)
- Pas de séparation claire entre données d'entraînement et de test (lignes)

Arbres de décision

Colonnes :

- création d'arbres pour prédire la note *de chaque élément*
- n arbres à construire

Lignes : données manquantes

- où placer dans l'arbre les utilisateurs qui n'ont pas noté ?
- des deux côtés ? on perd la séparation dans l'arbre
- astuces avec la réduction de dimension

Règles d'association

- Historiquement, règles d'association proposées dans le contexte du supermarché
- On considère un ensemble de transactions $T = \{T_1 \dots T_m\}$ définies sur n éléments de I .
- Chaque T est un sous-ensemble d'éléments de I , on cherche les corrélations de sous-ensembles
- Exemple : $\{\text{Bread, Butter, Milk}\}$ et $\{\text{Fish, Beef, Ham}\}$ sont fréquents
- Mary a acheté $\{\text{Butter, Milk}\}$, on peut penser qu'elle achètera Bread

| Item \Rightarrow | <i>Bread</i> | <i>Butter</i> | <i>Milk</i> | <i>Fish</i> | <i>Beef</i> | <i>Ham</i> |
|-----------------------|--------------|---------------|-------------|-------------|-------------|------------|
| Customer \Downarrow | | | | | | |
| Jack | 1 | 1 | 1 | 0 | 0 | 0 |
| Mary | 0 | 1 | 1 | 0 | 1 | 0 |
| Jane | 1 | 1 | 0 | 0 | 0 | 0 |
| Sayani | 1 | 1 | 1 | 1 | 1 | 1 |
| John | 0 | 0 | 0 | 1 | 0 | 1 |
| Tom | 0 | 0 | 0 | 1 | 1 | 1 |
| Peter | 0 | 1 | 0 | 1 | 1 | 0 |

Règles d'association

- Une règle se note $X \Rightarrow Y$ ($\{\text{Butter, Milk}\} \Rightarrow \{\text{Milk}\}$)
- On utilise deux notions pour caractériser les fréquences des sous-ensembles (*itemset*)
 - Le support de $X \subseteq I$, défini comme la fraction des transactions dont X est un sous-ensemble
 - La confiance, définie comme la probabilité conditionnelle que T contienne Y sachant qu'elle contient X .
Obtenue en divisant le support de $X \cup Y$ par celui de X
- On dit qu'une règle est une règle d'association de support minimal s et de confiance minimale c si :
 - Le support de $X \cup Y$ est d'au moins s
 - La confiance $X \Rightarrow Y$ est au moins c

Règles d'association

Pour trouver les règles d'associations, on procède en 2 étapes :

- On cherche tous les *itemsets* Z de support $s > s_\theta$
- Pour chacun, on calcule toutes les partitions $(X, Z-X)$, pour créer des règles $X \Rightarrow Z - X$
On garde toutes les règles de confiance $c > c_\theta$
- c_θ et s_θ sont des seuils
- La première phase est coûteuse, c'est un champ de recherche (*frequent itemset mining*)

Adaptation au CF :

- Ces règles sont utiles avec des matrices unaires.
- On garde celles dont le 2nd membre contient 1 élément (exactement)
- On cherche les règles d'un utilisateur $(X \cup X_u)$, on trie par confiance décroissante
- Les k premiers éléments sont le top- k pour cet utilisateur
- (plein de variantes)

Classifieur Bayésien naïf

- On oublie l'ordre des notes, on considère qu'il s'agit de l catégories $v_1 \dots v_l$
- On cherche r_{uj} , avec une valeur parmi $v_1 \dots v_l$
- On doit déterminer la probabilité que r_{uj} prenne chacune de ses valeurs, sachant que l'on dispose d'un ensemble de notes I_u :

$$P(r_{uj} = v_s | \text{notes de } I_u) \quad \forall s \in \{v_1, \dots, v_l\}$$

- On transforme avec :

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(B)}$$

- en :

$$P(r_{uj} = v_s | \text{notes de } I_u) = \frac{P(r_{uj} = v_s) \cdot P(\text{notes de } I_u | r_{uj} = v_s)}{P(\text{notes de } I_u)}$$

- On gardera la plus grande probabilité

Classifieur Bayésien naïf

- Le dénominateur ne dépend pas de s :

$$P(r_{uj} = v_s | \text{notes de } I_u) \propto P(r_{uj} = v_s) \cdot P(\text{notes de } I_u | r_{uj} = v_s)$$

- $P(r_{uj} = v_s)$, *prior*, est la fraction des utilisateurs ayant donné v_s comme note, parmi ceux ayant noté j
- Avec l'hypothèse *naïf* (indépendance des notes), on a :

$$P(\text{notes de } I_u | r_{uj} = v_s) = \prod_{k \in I_u} P(r_{uk} | r_{uj} = v_s)$$

- chaque $P(r_{uk} | r_{uj} = v_s)$ est la fraction des utilisateurs qui ont noté r_{uk} l'élément k , sachant qu'ils ont noté v_s l'élément j

On peut ensuite estimer \hat{r}_{uj} :

- en calculant les probabilités pour tous les s , prendre la plus grande, garder v_s
- raisonnable si l est petit
- utiliser une moyenne pondérée, dont les poids des valeurs possibles sont les probabilités

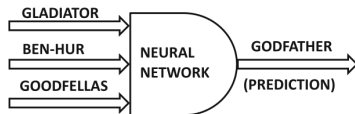
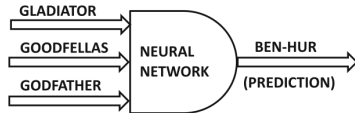
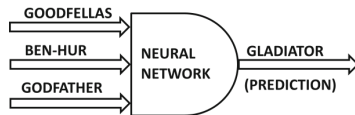
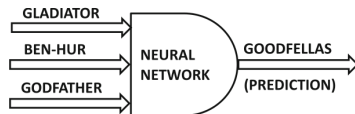
Classifieur arbitraire comme boîte noire

- Dans le cas non unaire, travailler sur une matrice incomplète est délicat
- réduction de dimension possible
- dans l'espace original, on peut adopter une solution itérative :
 - on initialise les valeurs manquantes (moyenne ligne, colonne), éventuellement centrage
 - on fixe une colonne (cible), les autres servent d'entrée (feature variable)
 - on apprend avec les notes existantes de la colonne, on prédit le reste
 - on met à jour la colonne avec les prédictions
 - on répète, jusqu'à convergence

Modèle neuronal comme boîte noire

| | GLADIATOR | BEN-HUR | GODFATHER | GOODFELLAS |
|-------|-----------|---------|-----------|------------|
| U_1 | 2 | | 5 | 5 |
| U_2 | | 1 | 4 | 4 |
| U_3 | 3 | | 1 | |
| U_4 | | 5 | 1 | |
| U_5 | 1 | 1 | 4 | |
| U_6 | 5 | | | 1 |

| | GLADIATOR | BEN-HUR | GODFATHER | GOODFELLAS |
|-------|-----------|---------|-----------|------------|
| U_1 | -2 | 0 | 1 | 1 |
| U_2 | 0 | -2 | 1 | 1 |
| U_3 | 1 | 0 | -1 | 0 |
| U_4 | 0 | 2 | -2 | 0 |
| U_5 | -1 | -1 | 2 | 0 |
| U_6 | 2 | 0 | 0 | -2 |



Plan

3 | Filtrage collaboratif par modèle

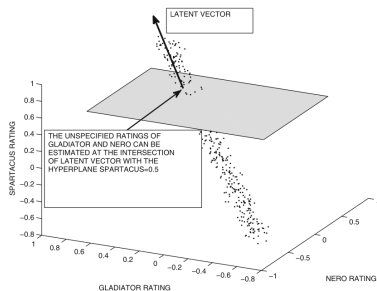
1 – Introduction

2 – Modèles à facteurs latents

Modèles à facteurs latents

- L'idée est de bénéficier du fait que des parties importantes des lignes/colonnes sont grandement corrélées
- Redondances
- On peut approximer la matrice complète raisonnablement avec une matrice de rang faible
- Un sous-ensemble d'entrées de la matrice suffit pour obtenir une matrice complète de rang faible
- Cette matrice de rang faible fournit des estimations robustes des entrées manquantes de la matrice initiale
- Ces méthodes sont l'état de l'art pour les systèmes de recommandation

Intuition géométrique



- Notes corrélées, rang 1 (droite)
- Ici, une seule note (Spartacus 0.5) suffit pour estimer Nero et Gladiateur!
- Intersection entre le vecteur latent (droite) et l'hyperplan (parallèle aux axes) tel que Spartacus ait 0.5

- En pratique : on n'a pas besoin de toute la matrice pour estimer les vecteurs latents principaux
- Orthogonalité des vecteurs latents

Principe de la factorisation de matrices

- Une matrice R de rang $k \ll \min(m, n)$ peut toujours être écrite sous la forme :

$$R = UV^T$$

- U est une matrice $m \times k$, V $n \times k$
- Les colonnes de U sont des vecteurs d'une base de l'espace de dim k des colonnes de R
- Une ligne de V contient les coefficients pour combiner ces vecteurs en une colonne de R .
- Il existe un nombre infini de factorisations, correspondants à divers ensembles de vecteurs
- La SVD est un exemple, dans lequel il y a orthogonalité des colonnes/lignes

Un peu d'algèbre linéaire

- si la matrice R a un rang plus grand que k , on peut l'approximer par

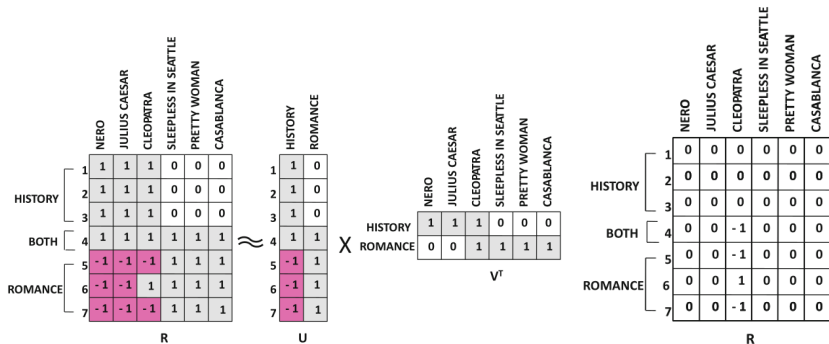
$$R \approx UV^T$$

- U est une matrice $m \times k$, V $n \times k$
- l'erreur d'approximation est

$$\|R - UV^T\|^2$$

- $\|\cdot\|^2$ représente la somme des carrés des entrées de $R - UV^T$, c'est la norme de Frobenius

Un peu d'algèbre linéaire



- les utilisateurs 1-3 aiment les films historiques, neutre sur Romance
- 4 aime les 2 genres
- 4-7 aiment Romance, pas films historiques
- nombreuses corrélations!
- si on multiplie par -1, interprétations plus délicates

Un peu d'algèbre linéaire

- chaque colonne de U est appelée vecteur latent, chaque ligne facteur latent
- la i ème ligne u_i de U est un facteur utilisateur, contenant "l'affinité" de l'utilisateur i pour chacun des k concepts (genres)
- chaque ligne v_j de V est un facteur "item", contenant l'appartenance des films à chaque genre

Apprentissage

- On cherche à résoudre :

$$\text{minimiser } J = \frac{1}{2} \|R - UV^T\|^2$$

- Sans contraintes sur U et V
- "loss" quadratique, diverses SGD peuvent le résoudre
- MAIS : valeurs manquantes!
- Si on pouvait factoriser R en UV^T complètes, on pourrait prédire les entrées manquantes de R :

$$\hat{r}_{ij} = \sum_{s=1}^k u_{is} \cdot v_{js}$$

- l'erreur pour un item est :

$$e_{ij} = r_{ij} - \hat{r}_{ij}$$

- d'où :

$$\text{minimiser } J = \frac{1}{2} \sum_{(i,j) \in S} e_{ij}^2$$

- noter le "seulement sur les valeurs de S"

Calculer : batch update method

Algorithm $GD(\text{Ratings Matrix: } R, \text{ Learning Rate: } \alpha)$

begin

Randomly initialize matrices U and V ;

$S = \{(i, j) : r_{ij} \text{ is observed}\}$;

while not(convergence) **do**

begin

Compute each error $e_{ij} \in S$ as the observed entries of $R - UV^T$;

for each user-component pair (i, q) **do** $u_{iq}^+ \leftarrow u_{iq} + \alpha \cdot \sum_{j:(i,j) \in S} e_{ij} \cdot v_{jq}$;

for each item-component pair (j, q) **do** $v_{jq}^+ \leftarrow v_{jq} + \alpha \cdot \sum_{i:(i,j) \in S} e_{ij} \cdot u_{iq}$;

for each user-component pair (i, q) **do** $u_{iq} \leftarrow u_{iq}^+$;

for each item-component pair (j, q) **do** $v_{jq} \leftarrow v_{jq}^+$;

Check convergence condition;

end

end

Calculer : SGD

Algorithm *SGD*(Ratings Matrix: R , Learning Rate: α)

```

begin
  Randomly initialize matrices  $U$  and  $V$ ;
   $S = \{(i, j) : r_{ij} \text{ is observed}\}$ ;
  while not(convergence) do
    begin
      Randomly shuffle observed entries in  $S$ ;
      for each  $(i, j) \in S$  in shuffled order do
        begin
           $e_{ij} \leftarrow r_{ij} - \sum_{s=1}^k u_{is}v_{js}$ ;
          for each  $q \in \{1 \dots k\}$  do  $u_{iq}^+ \leftarrow u_{iq} + \alpha \cdot e_{ij} \cdot v_{jq}$ ;
          for each  $q \in \{1 \dots k\}$  do  $v_{jq}^+ \leftarrow v_{jq} + \alpha \cdot e_{ij} \cdot u_{iq}$ ;
          for each  $q \in \{1 \dots k\}$  do  $u_{iq} = u_{iq}^+$  and  $v_{jq} = v_{jq}^+$ ;
        end
        Check convergence condition;
      end
    end
  end

```

- U et V mises à jour simultanément, convergence plus rapide que batch
- SGD préférable pour données larges et temps de calcul goulet d'étranglement
- $\alpha = 0.005$ (learning rate), peut être adapté à chaque étape
- seuil de convergence à fixer astucieusement, trop d'itérations peuvent dégrader la qualité
- initialisation dans $(-1, 1)$?

Régularisation

- Pénaliser les gros coefficients dans U et V
- Ajout d'un terme $\frac{\lambda}{2}(\|U\|^2 + \|V\|^2)$, $\lambda > 0$

$$u_{iq} \leftarrow u_{iq} + \alpha \left(\sum_{j:(i,j) \in S} e_{ij} \cdot v_{jq} - \lambda \cdot u_{iq} \right) \quad \forall q \in \{1 \dots k\}$$

$$v_{jq} \leftarrow v_{jq} + \alpha \left(\sum_{i:(i,j) \in S} e_{ij} \cdot u_{iq} - \lambda \cdot v_{jq} \right) \quad \forall q \in \{1 \dots k\}$$

The updates can be executed to convergence. One can also write these updates in terms of the $m \times n$ error matrix $E = [e_{ij}]$ in which unobserved entries of E are set to 0:

$$U \leftarrow U(1 - \alpha \cdot \lambda) + \alpha EV$$

$$V \leftarrow V(1 - \alpha \cdot \lambda) + \alpha E^T U$$

Principe de l'ALS

- Alternate Least Squares (Moindre Carrés Alternés)
- plus stable que SGD
- moins sensible à l'initialisation et la taille des pas
- moins efficace que SGD sur grandes données explicites
- version pondérée avec implicit feedback possible

Principe

- U fixé, on traite le problème comme un problème de moindre carrés sur V. On détermine ainsi les k facteurs latents de V. Il faut résoudre n problèmes, mais ils sont indépendants (parallélisation)
- V fixé, on traite le problème comme un problème de moindre carrés sur U. On détermine ainsi les k facteurs latents de U. Il faut résoudre m problèmes, mais ils sont indépendants (parallélisation)

NMF : non-negative Matrix Factorisation

- Outre la SVD, il existe d'autres factorisations, comme la NMF
- même fonction à optimiser, mais contraintes sur U et V :

$$U > 0, V > 0$$

- plus interprétable que d'autres méthodes
- marche bien avec échelles (1-5) mais surtout "like sans dislike" (implicit)

$$u_{ij} \leftarrow \frac{(RV)_{ij} u_{ij}}{(UV^T V)_{ij} + \epsilon} \quad \forall i \in \{1 \dots m\}, \forall j \in \{1 \dots k\}$$

$$v_{ij} \leftarrow \frac{(R^T U)_{ij} v_{ij}}{(V U^T U)_{ij} + \epsilon} \quad \forall i \in \{1 \dots n\}, \forall j \in \{1 \dots k\}$$

- $\epsilon \sim 10^{-9}$
- les valeurs dans U et V sont fixées à celles de la fin de l'itération précédente : mise à jour simultanées
- initialisation avec valeurs faibles dans $[0, 1]$.

Factorisation de matrices : résumé

| Method | Constraints | Objective | Advantages/Disadvantages |
|---------------|------------------|----------------------------------|--|
| Unconstrained | No constraints | Frobenius + regularizer | Highest quality solution Good for most matrices Regularization prevents overfitting Poor interpretability |
| SVD | Orthogonal Basis | Frobenius + regularizer | Good visual interpretability Out-of-sample recommendations Good for dense matrices Poor semantic interpretability Suboptimal in sparse matrices |
| Max. Margin | No constraints | Hinge loss + margin regularizer | Highest quality solution Resists overfitting Similar to unconstrained Poor interpretability Good for discrete ratings |
| NMF | Non-negativity | Frobenius + regularizer | Good quality solution High semantic interpretability Loses interpretability with both like/dislike ratings Less overfitting in some cases Best for implicit feedback |
| PLSA | Non-negativity | Maximum Likelihood + regularizer | Good quality solution High semantic interpretability Probabilistic interpretation Loses interpretability with both like/dislike ratings Less overfitting in some cases Best for implicit feedback |

Semaine prochaine

Suite du cours :

- Content-based filtering
- Recommandation Sociale
- Recommandation hybride
- Évaluation

Références

Merci

Ce livre repose notamment sur :

- le livre Recommender systems, the textbook, de Charu C. Aggarwal. Springer, 2016.

