Recommandation de contenu

Nicolas Dugué

26 mai 2021

1 Rappels de cours sur la recommandation

Les données réelles sont très souvent représentées sous forme de matrice. Dans le cas du cours de Big Data, nous avons parlé d'un certain nombre d'applications qui font usage de la représentation matricielle. Soit une matrice $\mathbb{X}$:

Exemple 1 (E-commerce) *Dans l'application E-commerce, on représente les clients en ligne et les produits en colonne. Les valeurs $\mathbb{X}_{ij}$ représentent le nombre d'achats du produit j par le client i :*

$$\mathbb{X} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Dans l'exemple ci-dessus, le client 1 a acheté un exemplaire du produit 2 et du produit 4. Le client 2 a acheté un exemplaire du produit 3 et du produit 5.

Recommandation. À partir de ces informations sous forme matricielle, il est possible de bâtir des algorithmes de recommandation. L'objectif est de recommander à chaque client les produits les plus pertinents étant donné nos connaissances. Ces algorithmes peuvent adopter trois philosophies différentes :

- filtrage collaboratif utilisateur - la recommandation est basée sur la similarité entre utilisateurs. Ici, par exemple si un·e client·e A et B ont acheté les mêmes produits, on peut utiliser les achats de A pour les recommander à B ;
- filtrage collaboratif objet - la recommandation est basée sur la similarité entre objets. Ici, par exemple si deux objets A et B ont été achetés par les mêmes client·e·s, alors on peut recommander B aux client·e·s qui achètent A ;
- filtrage basé contenus - si deux objets A et B sont jugés similaires en contenu, on peut recommander B aux gens intéressés par A. Par exemple, ici, on peut se baser sur la similarité entre les textes de description des objets pour calculer une similarité.

Les représentations matricielles sont également utilisées pour les données structurées sous forme de graphe comme c'est le cas pour les réseaux sociaux :

Exemple 2 (Réseaux sociaux) *Dans ce cas, les utilisateurs sont à la fois les ligne et les colonnes. La matrice $\mathbb{X}$ est carrée et de dimension "nombre d'utilisateurs" :*

$$\mathbb{X} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Dans ce cas, nous considérons une matrice symétrique : si x est ami avec y alors y est ami avec x (cas facebook). Dans notre exemple : l'utilisateur 1 est connecté aux utilisateurs 2 et 4, l'utilisateur 2 aux utilisateurs 1 et 3, l'utilisateur 3 aux utilisateurs 2 et 4 et l'utilisateur 4 aux utilisateurs 1 et 3.

Pour reprendre certains exemples du cours, on peut également représenter sous forme de matrices :

- les documents textuels : chaque document est une ligne, les colonnes représentent les mots qui composent ces documents ;
- des individus : chaque ligne est un individu, les colonnes sont des attributs utiles à leur caractérisation (taille, âge, couleur des yeux) ;
- des CVs : chaque ligne est un CV, chaque colonne une compétence ;
- des déplacements : chaque ligne est un individu, les colonnes sont des coordonnées GPS.

Matrice creuse. Dans tous ces exemples, les matrices utilisées pour représenter les données sont dites "creuses" (*sparse* en anglais). Une matrice "creuse" est une matrice qui contient majoritairement des valeurs 0. Dans le cas de l'e-commerce, le catalogue de produits peut-être considérable : et un·e client·e se contentera bien souvent d'acheter peu de produits : la ligne qui représente ce client dans la matrice sera essentiellement constituée de 0. Dans un réseau social, on dénombre (suivant les applications) des centaines de millions d'utilisateurs. Mais chaque utilisateur n'est connecté qu'à une petite fraction de ces centaines de millions d'utilisateurs. Si le réseau social est composé de $100.000.000$ d'utilisateurs et qu'un utilisateur se connecte à 5.000 d'entre eux seulement : la ligne qui représente l'utilisateur dans la matrice sera composée de 99,9$\%$ de valeurs nulles !

So what ? Si l'on considère naïvement ces matrices dans nos algorithmes, nous allons surcharger le disque dur, la mémoire et les processeurs. En effet, une matrice de taille 1 million * 1 million = 10^{12} valeurs de chacune 2 octets (si entier, 4 si réel par ex), donc $2 * 10^{12}$ octets, soit 2 teraoctets... Il sera donc impossible de charger toute la matrice en mémoire, alors que cela accélère considérablement les algorithmes. Les opérations sur nos matrices seront lourdes. Pourtant, si la matrice est creuse à 99,9% comme c'est fréquemment le cas, alors, en évitant de stocker les zéros, il devient possible de stocker les données sur 2Go ! Cela permettrait donc de tout traiter en mémoire.

Ok... Et comment fait-on cela ? Nous allons utiliser des formats de stockage adaptés aux matrices creuses. L'un des plus utilisés est le format *CSR* : Compressed Sparse Row. Ce format nous permettra de ne pas stocker les 0 tout en conservant la structuration matricielle de nos données.

2 Compressed Sparse Row : rappels de cours

La Figure 1 présente à droite la version CSR de la matrice de gauche. Une matrice au format CSR est en effet composée de trois vecteurs. Commençons par le troisième qui est le plus simple, celui nommée *values* sur la Figure. Ce vecteur contient toutes les valeurs non nulles de la matrice, ordonnées de gauche à droite et de haut en bas. On ne stocke ainsi plus les 0 qui nous gênaient tant. Le deuxième vecteur, *column indices*, donne pour chacune des valeurs du vecteur *values*, leur numéro de colonne [1]. Enfin, le premier vecteur, *row offsets*, contient lui les informations sur les lignes. Il est compressé, il existe donc une subtilité quant à son fonctionnement : $0, 2, 4, 7, 9$ signifie que la ligne 0 (de la matrice) commence à l'indice 0 dans les vecteurs *values* et *column indices*

1. les numéros de lignes et de colonnes débutent à 0

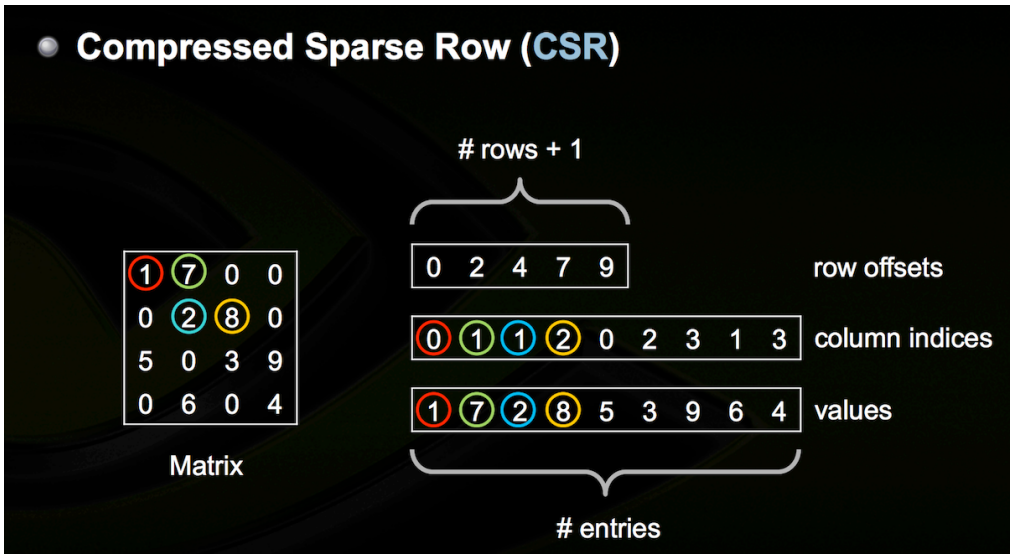

FIGURE 1 – À gauche la représentation matricielle classique. À droite, la représentation au format CSR.

et se termine à l'indice 2 non compris, que la ligne 1 commence à l'indice 2 et se termine à l'indice 4 nom compris, que la ligne 2 commence à l'indice 4 et termine à l'indice 7 non compris, et que la ligne 3 commence à l'indice 7 et termine à l'indice 9 non compris.

Dans le cas d'exemple, la représentation de droite paraît plus gourmande en ressources que celle de gauche. Mais dans le cas d'une matrice remplie à 99% de 0, le gain d'espace devient considérable avec le format CSR.

3 Exercice : Recommandation

Soit le jeu de données du Tableau 1 sur lequel nous souhaitons paramétrer un algorithme de recommandation de type filtrage collaboratif basé utilisateur.

User Movie	M1	M2	M3	M4	M5	M6
U1	L	D			D	D
U2		D			L	D
U3	L		D	D	L	D
U4		D		D	D	D

TABLE 1 – Jeu de données avec des utilisateurs et des films : "D" = Dislike, "L" = Like.

3.1 Représenter les données

1. Représenter les données sous forme de matrice dense : quelles valeurs choisir pour L, D, ou les éléments vides ?
2. Représenter les données sous forme de matrice creuse CSR.

3.2 Recommandation

Pour paramétrer et évaluer notre algorithme de recommandation, on commence à séparer nos données en : données d'apprentissage (80% = 12 notes), données de test (20% = 3 notes).

User Movie	M1	M2	M3	M4	M5	M6
U1	L	D			D	
U2		D			L	D
U3				D	L	D
U4		D		D		D

TABLE 2 – Jeu de données d'apprentissage.

User Movie	M1	M2	M3	M4	M5	M6
U1						D
U2						
U3	L					
U4					D	

TABLE 3 – Jeu de données de test.

1. On cherche donc à faire trois prédictions :
   — Recommande-t-on M6 à U1 ?
   — Recommande-t-on M1 à U3 ?
   — Recommande-t-on M5 à U4 ?
   Pour cela, on cherche le ou les utilisateurs les plus proches de l'utilisateur considéré dans le jeu de données d'apprentissage en utilisation la distance de Manhattan :

$$||u, v||_1 = \sum_{i=0}^{k} |u_i - v_j|$$

   On fait ensuite voter ces utilisateurs : si la majorité vote pour like (resp. dislike), alors le système prédit like (resp. dislike).
2. Calculer accuracy, précision et rappel. Notre algorithme de recommandation est-il efficace ?