Premier exemple d'apprentissage machine : apprendre les règles d'associations

L'une des thématiques en vogue (pour le grand public) de l'intelligence artificielle (en anglais *Artificial Intelligence*) est l'apprentissage automatique (en anglais *Machine Learning*), l'apprentissage profond (en anglais *Deep Learning*) a depuis une quinzaine d'années le vent en poupe. Mais c'est loin d'être le seul candidat pour faire de l'apprentissage.

Nous allons étudier (en cours et TD) un (plus exactement deux) des algorithmes permettant d'extraire d'une base de données contenant des transactions commerciales des règles d'associations entre articles achetés. L'un des intérêts majeurs (pour les enseignes de distribution) est de pouvoir organiser des campagnes de marketing mieux ciblées (et donc plus rémunératrices) ou de promouvoir certains articles pour susciter l'achat (compulsif ?) chez leurs clients.

Dans la suite j'utiliserai la terminologie anglo-saxonne tout en essayant de fournir l'équivalent français, même s'il est peu utilisé dans le domaine. Les algorithmes *Apriori* et *tid-Apriori* (pour *transactions ID*) ont été proposés en 1994 par R Agrawal & R. Srikant dans
"**Fast Algorithms for Mining Association Rules**"
La fouille de données (en anglais *Data Mining*) a pour objectif, d'extraire de nouvelles connaissances à partir de (large) base de données (en 2023, on parle plus volontiers de *Big Data* du fait de l'évolution du stockage qui est passé de bases structurées manipulées via des langages type **SQL** à des collections de données non nécessairement organisées). Avec la systématisation des codes barre, les caisses enregistreuses n'ont aucun mal à stocker les achats des clients, la date, le numéro de la carte de fidélité, le numéro de la caisse, l'identifiant de la personne qui saisit (petit effet de bord, on peut capter la vitesse de saisie de la personne en caisse et appliquer des pratiques managériales de suivi d'efficacité ...).

On peut formellement décrire le problème à traiter à partir de deux ensembles :

- Un ensemble d'items (en français articles) $\mathcal{I} = \{i_{1,}i_{2,}\dots,i_n\}$. Un *itemset* désigne un sous-ensemble quelconque de $\mathcal{I}$
- Un ensemble de transactions $\mathcal{D} = \{t_1, t_2, \dots t_k\}$, telles que $t_i \subset \mathcal{I}$, à chaque transaction est associée un identifiant unique "tid"

On dira qu'une transaction T contient un ensemble d'articles X si $X \subset T$. L'objectif va être de trouver des co-occurrences fréquentes d'achat, pour cela on définit une mesure de fréquence d'achat (le support) et une règle d'association sera simplement le reflet de cette co-occurrence.

- Le **support d'un itemset** X c'est une *estimation* de la probabilité d'apparition de X dans les transactions, calculée à partir de la fréquence d'apparition de X dans les transactions de $\mathcal{D}$. C'est le rapport entre le cardinal de $\{t \in \mathcal{D} \mid X \subset t\}$ et le cardinal de $\mathcal{D}$, on le note $\mathbf{supp}(X)$
- une *règle d'association* notée $X \longrightarrow Y$ **vérifie** $X \subset \mathcal{I}, Y \subset \mathcal{I}, X \cap Y = \emptyset$

> **Note**
>
> La génération de règles d'association repose sur le produit cartésien des items ayant les plus grosses fréquences d'apparition. Pour 100 articles, il y a $2^{100} \simeq 10^6$ itemsets possibles et pour un itemset de taille $k > 1$ il y a $2^k - 2$ règles possibles

Quelques définitions

- Le **compteur de support** d'un *itemset* X c'est simplement le nombre de transactions où X apparaît
- Un *itemset* est dit **fréquent** si son compteur est supérieur au compteur support minimum
- Un *itemset* est dit **fréquent maximal** si aucun sur-ensemble n'est fréquent
- Un *itemset* X est dit **fermé** si aucun *itemset* de la forme $X \cup \{i_p\}$ n'a un compteur de support aussi grand que lui

> **Exemple**
>
tid	A	B	C	D
1	1	0	1	1
2	0	0	1	1
3	1	1	1	1
4	0	1	0	0
5	1	1	1	1
count	3	3	4	5
>
> Considérons que le support minimal soit fixé à 3
> Tous les items sont fréquents
> A n'est pas maximal puisque ACD est fréquent
> B est fermé puisque AB, BC et BD ont un support de 2, il est maximal puisque aucun sur-ensemble n'a un support ≥ 3
> D est fermé mais nest pas maximal puisque AD et CD sont fréquents

Quelques mesures pour les règles d'association

- L'*indice* de **support d'une règle** $X \longrightarrow Y$ c'est une *estimation* de la probabilité d'apparition de X et Y parmi les transactions. C'est le rapport entre le cardinal $\{t \in \mathcal{D} \mid (X \cup Y) \subset t\}$ et le cardinal de $\mathcal{D}$, on le note $\mathbf{supp}(X \longrightarrow Y) = \mathbf{supp}(\boldsymbol{X \cup Y})$
- L'*indice* de **confiance d'une règle** (en anglais *confidence*) $X \longrightarrow Y$ c'est une *estimation* de la probabilité d'apparition de Y sachant X. C'est le rapport entre le cardinal $\{t \in \mathcal{D} \mid (X \cup Y) \subset t\}$ et le cardinal de $\{t \in \mathcal{D} \mid X \subset t\}$, on le note $\mathbf{conf}(X \longrightarrow Y) = \frac{\mathbf{supp}(X \cup Y)}{\mathbf{supp}(X)}$
- L'indice **d'amélioration d'une règle** (en anglais *lift*) $X \longrightarrow Y$ permet de quantifier l'amélioration apportée par la règle par rapport à une situation où X et Y seraient indépendantes. C'est le rapport entre le cardinal $\{t \in \mathcal{D} \mid (X \cup Y) \subset t\}$ et le produit du cardinal de $\{t \in \mathcal{D} \mid X \subset t\}$ et du cardinal de $\{t \in \mathcal{D} \mid Y \subset t\}$, on le note $\mathbf{lift}(X \longrightarrow Y) = \frac{\mathbf{supp}(\boldsymbol{X \cup Y})}{\mathbf{supp}(X) \times \mathbf{supp}(Y)}$. Si le score est > 1, il y a une corrélation positive entre X et Y, la règle est donc pertinente. Si le score est < 1, cela signifie que la présence de X a un effet négatif sur la présence de Y (et réciproquement) dans une même transaction.
- L'indice de **levier** (en anglais *leverage*) $X \longrightarrow Y$ est très similaire à l'indice d'amélioration. Au lieu de faire le ration, on effectue la différence entre le cardinal $\{t \in \mathcal{D} \mid (X \cup Y) \subset t\}$ et le produit du cardinal de $\{t \in \mathcal{D} \mid X \subset t\}$ et du cardinal de $\{t \in \mathcal{D} \mid Y \subset t\}$ selon support.bigml.com

  > Quote
  >
  > The implications are that lift may find very strong associations for less frequent items, while leverage tends to prioritize items with higher frequencies/support in the dataset.

- L'indice de **conviction d'une règle** $X \longrightarrow Y$ est une estimation de la probabilité que X apparaisse sans que Y n'apparaisse. C'est en quelque sorte une estimation que la règle soit fausse. $\mathbf{conv}(X \longrightarrow Y) = \frac{1 - \mathbf{supp}(Y)}{1 - \mathbf{conf}(X \longrightarrow Y)}$
- D'autres mesures sont possibles, les curieux peuvent se reporter à "**A Probabilistic Comparison of Commonly Used Interest Measures for Association Rules**" de Michael Hahsler (2015)

Exemple

source wikipédia

tid	lait	pain	beurre	couches	bière	oeufs	fruits
1	1	1	0	0	0	0	1

tid	lait	pain	beurre	couches	bière	oeufs	fruits
2	0	0	1	0	0	1	1
3	0	0	0	1	1	0	0
4	1	1	1	0	0	1	1
5	0	1	0	0	0	0	0
count	2	3	2	1	1	2	3

- $\textbf{supp}(\text{lait}) = \frac{2}{5}$; $\textbf{supp}(\{\text{lait}, \text{pain}\}) = \frac{2}{5}$; $\textbf{supp}(\{\text{beurre}, \text{pain}\}) = \frac{1}{5}$; $\textbf{supp}(\{\text{beurre}, \text{pain}, \text{lait}\}) = \frac{1}{5}$
- $\textbf{supp}(\{\text{lait}, \text{pain}\} \longrightarrow \{\text{beurre}\}) = \frac{1}{5}$, de même $\textbf{supp}(\{\text{beurre}, \text{pain}\} \longrightarrow \{\text{lait}\}) = \frac{1}{5}$
- $\textbf{conf}(\{\text{lait}, \text{pain}\} \longrightarrow \{\text{beurre}\}) = \frac{1}{2}$ **mais** $\textbf{conf}(\{\text{beurre}, \text{pain}\} \longrightarrow \{\text{lait}\}) = 1$
- $\textbf{lift}(\{\text{lait}, \text{pain}\} \longrightarrow \{\text{beurre}\}) = \frac{1\times5}{2\times2}$ **et** $\textbf{lift}(\{\text{beurre}, \text{pain}\} \longrightarrow \{\text{lait}\}) = \frac{1\times5}{1\times2}$
- $\textbf{conv}(\{\text{lait}, \text{pain}\} \longrightarrow \{\text{beurre}\}) = \frac{1-\frac{2}{5}}{1-\frac{1}{2}} = \frac{6}{5}$ **et** $\textbf{conv}(\{\text{beurre}, \text{pain}\} \longrightarrow \{\text{lait}\}) = +\infty$

Principe général de l'algorithme "Apriori"

Cet algorithme repose sur la propriété que **tout sous-ensemble d'un *itemset* fréquent est fréquent** et, en conséquence **tout sur-ensemble d'un *itemset* non fréquent n'est pas fréquent**

L'algorithme va alterner entre la création d'itemsets candidats de longueur k et de listes d'itemsets de longueur k fréquents. Nous allons illustrer le fonctionnement à partir d'un exemple tiré de An improved Apriori algorithm for association rules de M. Al-Maolegi, B. Arkok paru dans la revue IJNLC vol 3, num 1, 2014

Exemple

TID	Items
1	1,2,5
2	2,4
3	2,4
4	1,2,4
5	1,3
6	2,3
7	1,3
8	1,2,3,5
9	1,2,3

Pour faciliter la lecture, la table va être présentée différemment

TID	1	2	3	4	5
1	1	1	0	0	1
2	0	1	0	1	0
3	0	1	0	1	0
4	1	1	0	1	0
5	1	0	1	0	0
6	0	1	1	0	0
7	1	0	1	0	0
8	1	1	1	0	1
9	1	1	1	0	0
c	6	7	5	3	2

On fixe la valeur seuil de support à $\mathbf{3}$

- On calcule C_1 les candidats de taille 1, on calcule le support de chaque candidat
- On calcule L_1 les candidats qui ont un support $\geq$ au seuil, on trouve { {1}, {2}, {3}, {4}}
- On calcule C_2, à partir de L_1. Pour éviter de faire des calculs inutiles, on va considérer que les items sont ordonnés dans un *itemset*, on ne combinera deux éléments de L_k que si les $k-1$ premiers éléments sont identiques. Ici comme L_1 est constitué de singleton, il n'y pas de vérification à faire, on trouve $C_2 = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}$, pour chaque candidat on calcule le support $[4,4,1,3,3,0]$
- D'où L_2 { {1,2}, {1,3}, {2,3}, {2,4} }
- On calcule C_3, on obtient { {1,2,3}, {2,3,4} } en combinant {1,2} et {1,3} d'une part et en combinant {2,3} et {2,4} d'autre part. Il faut maintenant vérifier que chaque sous-ensemble des 3-itemsets sont fréquents c'est l'opération d'**élagage** (en anglais *pruning*). Pour {1,2,3} il faut vérifier si {1,2}, {1,3}, {2,3} appartiennent à L_1, pour {2,3,4} il faut vérifier si {2,3}, {2,4} et {3,4} appartiennent à L_1. Après élagage, $C_3 = \{\{1,2,3\}\}$, on calcule son support $[2]$. D'une manière générale pour un candidat de taille k, on regarde si tous les sous-ensembles de taille $k-1$ sont dans L_{k-1}
- D'où $L_3 = \emptyset$

L'algorithme s'arrête lorsque le cardinal de L_k est ≤ 1 et renvoie $L = \bigcup_{i=1}^{k} L_k$ qui sera ici { {1}, {2}, {3}, {4}, {1,2}, {1,3}, {2,3}, {2,4} }

TID-Apriori

L'algorithme TID-Apriori vise à améliorer le calcul du support en évitant de consulter la base de données (autrement que pour le calcul de C_1). L'idée est de maintenir une structure qui, à chaque transaction (identifiée par son TID), va associer la liste des itemsets qu'elle englobe, ayant un support plus grand ou égal au support minimum. On notera T_k cette structure, qui sera en python un dictionnaire, dont les clefs seront les identifiants de transactions, la valeur une liste de tuples de longueur k et dont les éléments seront des items.

Principe général de la construction des règles

Etant donné un itemset I ={$i_1, i_2, \ldots i_p$}, on va construire, pour tout sous-ensemble non vide $a \subset I$ une règle $a \longrightarrow (I \setminus a)$ à condition que la confiance de la règle soit supérieure à un minimum de confiance. Comme la confiance est définie par $\frac{\mathbf{supp}(l)}{\mathbf{supp}(a)}$, que le $\mathbf{supp}(a)$ diminue avec la longueur de a il découle que si a ne génère pas de règle avec une confiance suffisante, tout sous-ensemble de a ne pourra pas générer de règle avec une confiance suffisante. En effet si, dans une fraction on augmente le *dénominateur* cela diminue la valeur de la fraction.
Ainsi on peut mettre en oeuvre un algorithme récursif, qui à chaque itération diminue de 1 la longueur de la partie gauche de la règle. L'algorithme s'arrêtera, soit parce qu'on a atteint une partie gauche de taille 1, soit parce qu'on n'aura pas obtenu une confiance suffisante.

De fait on utilisera une version plus rapide qui va faire croître la partie droite de la règle,et qui va s'appuyer sur la notion de produit cartésien. Supposons qu'on ait un itemset ABCDE, et qu'on n'ai pu générer que 2 règles ayant une partie droite de taille 1 :(ABCD $\longrightarrow$ E) et la seconde (ACDE $\longrightarrow$ B), la seule règle que l'on puisse construire avec une partie droite de taille 2 sera (ACD $\longrightarrow$ BE), il est inutile de considérer les parties droites contenant A ou C ou D. C'est une conséquence **directe** de *si une partie gauche ne génère pas une règle, tout sous-ensemble de cette partie gauche ne générera pas de règle* puisque (BCDE $\longrightarrow$ A) n'est pas une règle toute extension de A (à droite) correspond à un sous-ensemble de BCDE à gauche et ne générera pas de règle

A suivre >>>