Chapitre 5

Règles d'association

5.1 Introduction

De nombreuses entreprises commerciales accumulent d'importantes quantités de données lors de leurs opérations quotidiennes. Par exemple, les grands magasins collectent énormément de données sur les achats des consommateurs via les tickets de caisse. Le tableau 5.1 donne une illustration de ce type de données. Chaque rang correspond à une transaction et reporte le numéro de ticket ainsi qu'une liste de produits achetés. Les commerçants sont intéressés par l'analyse de ce type de données pour mieux connaître les comportements d'achat de leurs clients. Ces informations servent à bien mener les campagnes marketing, mieux gérer les inventaires ou améliorer les relations clients.

TID	Items
1	{Pain, Lait}
2	{Pain, Couches, Bière, Oeufs}
3	{Lait,Couches,Bière,Coca}
4	{Pain, Lait, Couches, Bière}
4	{Pain, Lait, Couches, Coca}

TABLE 5.1 – Panier de la ménagère

Dans ce chapitre, nous allons étudier une méthode connue sous le non d'analyse des associations et qui est utilisée pour découvrir des associations ou des relations cachées dans les grandes bases de données. Les relations découvertes peuvent être représentées sous forme de *règles d'association* ou d'ensemble d'items fréquents. Par exemple, on peut extraire la règle suivant de la table 5.1 :

$$\{Couches\} \rightarrow \{Biere\}$$

Cette règle suggère qu'il existe une relation forte entre la vente de couches et de bières parce de nombreux clients qui achètent des couches achètent aussi de la bière.

Les règles d'association sont bien sûr aussi applicables à d'autres domaines tels que la bioinformatique (génétique), les diagnostiques médicaux, le web mining, etc.

Il y a deux problèmes clés qui doivent être considérés quand on utilise des règles d'association. Tout d'abord l'extraction de motifs (plus ou moins fréquents) peut être numériquement coûteux si les bases de données sont importantes. Deuxièmement, certaines associations sont potentiellement fausses ou sans intérêt, elles apparaissent simplement par hasard.

Dans la première partie de ce chapitre, nous expliquons les concepts de base des l'analyses d'association et les algorithmes associés. Dans la seconde partie, nous nous intéressons à l'évaluation de la qualité des règles.

5.2 Définition du problème

5.2.1 Représentation binaire et définitions

Les données du panier de la ménagère, telles de celles présentées dans la table 5.1 peuvent facilement se mettre sous forme binaire.

TID	Pain	Lait	Couches	Bière	Oeufs	Coca
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

TABLE 5.2 – Panier de la ménagère en représentation binaire

5.2.2 Ensemble d'items et support

Soient $I = \{i_1, \cdots, i_d\}$ l'ensemble de tous les items des paniers et $T = \{t_1, \cdots, t_N\}$ l'ensemble de toutes les transactions. Chaque transaction contient un sous ensemble d'items.

Le *volume*[1] *de la transaction* est le nombre d'items contenus dans la transaction.

Une notion importante pour un ensemble d'items est son *support* qui fait référence au nombre de transactions (observées) qui le contiennent. Mathématiquement, le support $\sigma(X)$ d'un ensemble d'items X est défini par

$$\sigma(X) = Card\left(\{t_i | X \subseteq t_i, t_i \in T\}\right)$$

où $Card(A)$ représente le cardinal de l'ensemble A.

Exemple - Dans les données de la table 5.1, le support $\{Biere, Couches, Lait\}$ est égal à 2.

5.2.3 Règles d'association

Une règle d'association est une application de la forme $X \rightarrow Y$ où X et Y sont des ensembles d'items disjoints.

1. en anglais : width

Une règle d'association traduit seulement une co-occurrence et non une causalité.

La force d'une règle d'association peut être mesurée en utilisant son support et sa *confiance*[2].

$$Support, s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}$$

$$Confiance, c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

Exemple Considérons la règle $\{Lait, Couches\} \rightarrow \{Biere\}$. Le support de l'ensemble $\{Biere, Couches, Lait\}$ étant égal à 2 et le nombre total de transactions étant égal à 5, le support de la règle est 2/5 = 0.4. La confiance est obtenue en divisant le support de l'ensemble $\{Biere, Couches, Lait\}$ par le support de l'ensemble $\{Lait, Couches\}$. Comme il y a 3 transactions contenant $\{Lait, Couches\}$, la confiance de cette règle est 2/3 = 0.67.

Le support est important parce qu'une règle qui a un support faible peut être observée seulement par hasard. Le support est souvent utilisé pour éliminer les règles inintéressantes. La confiance mesure la pertinence de l'inférence faite par une règle. Plus la confiance de $X \rightarrow Y$ est élevée plus la probabilité d'observer Y avec X est forte. La confiance donne aussi une estimation de la probabilité conditionnelle de Y sachant X.

5.2.4 Recherche de règles d'association

Le problème de la recherche de règles d'association dans une base de données peut se formuler comme suit.

Définition 1 *Etant donné un ensemble de transactions T, trouver tout les règles ayant un support $\geq minsup$ et une confiance $\geq minconf$ où $minsup$ et $minconf$ sont des seuils pour le support et la confiance.*

Il n'est pas envisageable de chercher toutes les règles d'associations pour ensuite sélectionner celles qui ont un support et une confiance suffisants. Les coûts de calcul seraient prohibitifs. Par exemple pour un ensemble de d items, le nombre total de règles possibles est $R = 3^d - 2^{d+1} + 1$. Si $d = 6$, on a $R = 602$. En général, plus de 80% de règles sont éliminées en appliquant $minsup = 20\%$ et $minsup = 50\%$.

Un premier pas permettant d'améliorer les performances d'un algorithme de recherche de règle consiste à découpler les exigences sur le support et la confiance. La définition du support montre que le support d'une règle $X \rightarrow Y$ ne dépend que du support de son ensemble d'items $X \cup Y$. Par exemple les règles suivantes ont le même support car elles sont toutes construites à partir du même ensemble $\{Biere, Couche, Lait\}$:

$$\{Biere, Couche\} \rightarrow \{Lait\},\ \{Biere, Lait\} \rightarrow \{Couche\}$$
$$\{Biere\} \rightarrow \{Lait, Couche\},\ \{Lait\} \rightarrow \{Biere, Couche\}$$

2. en anglais : confidence

Si l'ensemble d'item est rare toutes les règles associées seront rares. Et elles peuvent être éliminées avant même de calculer leur confiance.

Ainsi une stratégie adoptée par la plupart des algorithmes de recherche de règles d'association consiste à décomposer le problème en deux étapes :

1. **Génération des ensembles d'items fréquents**, dont l'objectif est de trouver tous les ensembles d'items qui satisfont le seuil $minsup$.
2. **Génération de règles**, dont l'objectif est d'extraire toutes les règles de grande confiance à partir des ensembles d'items fréquents trouvés dans l'étape précédente. Ces règles sont appelées règles fortes.

La première étape est en générale plus coûteuse numériquement.

5.3 Génération d'ensembles d'items fréquents

On peut construire un graphe pour énumérer tous les ensembles d'items possible (voir figure 5.1) . Une approche consiste ensuite à calculer le support de tous les ensembles d'items du graphe. Pour le faire, il faut confronter chaque ensemble d'items à toutes les transactions observées. Dès qu'on trouve une transaction contenant l'ensemble le support de celui ci est incrémenté. Par exemple le support de $\{Pain, Lait\}$ est incrémenté 3 fois. Cette approche peut être très coûteuse car elle requiert $O(NM\omega)$ comparaison avec N le nombre de transactions, M le nombre d'ensembles d'items et ω la plus grande taille de transaction.

Il y a plusieurs façons de réduire le coût de la recherche d'ensembles d'items fréquents.

1. **Réduire le nombre d'ensemble d'items candidats (M)**. Le principe `Apriori`, décrit dans la section suivante est un moyen efficace d'éliminer certains candidats sans évaluer leur support.
2. **Réduire le nombre de comparaisons**. Au lieu de comparer chaque ensemble d'items candidat à toutes les transactions, on peut réduite le nombre de comparaison en utilisant des structures de données plus complexes, soit pour stocker les ensembles d'items soit pour compresser la base de données.

5.3.1 Le principe Apriori

Dans cette section nous décrivons comment la mesure de support aide à réduire le nombre d'ensembles d'items candidats considérés durant la génération des ensembles d'items fréquents.

Théorème 2 ***Principe Apriori*** *Si un ensemble d'items est fréquent, alors tous ses sous-ensembles sont aussi fréquents.*

Donner des exemples (voir figures 6.3 et 6.4)

La réciproque est que si un ensemble $\{a, b\}$ est peu fréquent, alors ses super-ensembles sont aussi peu fréquents (c'est à dire tous les ensembles d'items contenant $\{a, b\}$. Ainsi tous si on sait que $\{a, b\}$ est peu fréquent, on peut éliminer a priori tous les ensembles le contenant. Cette stratégie s'appelle **l'élagage basé sur le support**. Cette stratégie est rendue possible par une propriété clé de la mesure du support : le support d'un ensemble d'items n'est jamais supérieur au support de ses sous ensembles.

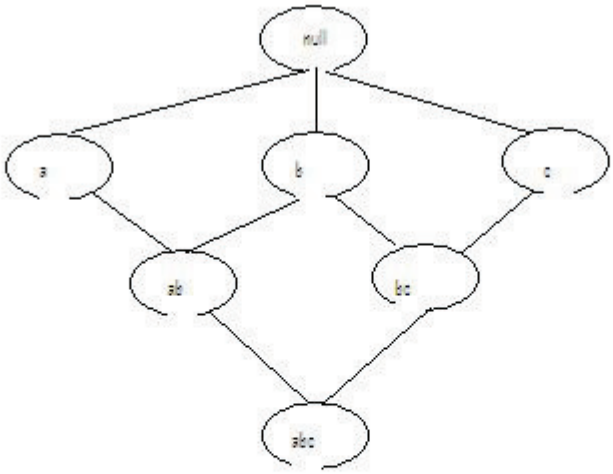

FIGURE 5.1 – Un graphe d'ensemble d'items

Définition 2 *(Propriété de monotonie) Soit I un ensemble d'items, et $J = 2^I$ la puissance de l'ensemble I. Une mesure f est monotone si*

$$\forall X, Y \in J \ : X \subseteq Y \rightarrow \ f(X) \leq f(Y),$$

Une mesure f est anti-monotone si

$$\forall X, Y \in J \ : X \subseteq Y \rightarrow \ f(X) \geq f(Y),$$

Toute mesure admettant une propriété d'anti-monotomie peut être intégrée dans un algorithme de recherche d'ensembles d'items fréquents.

5.3.2 Algorithme Apriori

L'algorithme Apriori (Agrawal et Srikant, 1994) est le premier algorithme de recherche de règles d'association incluant des étapes d'élagage pour tenir compte de la croissance exponentielle du nombre de d'ensemble d'items candidats.

Notons C_k l'ensembles des ensembles d'items candidats de taille k et F_k l'ensemble des ensemble d'items fréquents de taille k.

- L'algorithme commence par déterminer le support de chaque item : intialisation de F_1.
- L'agorithme génére itérativement les ensembles d'item fréquents de taille k à partir des ensembles d'items de taille $(k-1)$ obtenus à l'étape précédente.

Algorithme A priori

k=1
$F_k = \{i | i \in I$ et $\sigma(i) \geq N \times minsup\}$
Répéter jusqu'à $F_k =$
    $k = k + 1$
    $C_k =$ apriori-gen(f_{k-1})
    Pour toute transaction $t \in T$ faire
        $C_t =$ subset(C_k, t) %*Extrait tous les ensembles d'item contenant* t
            Pour tout ensemble de candidats $c \in C_t$ faire
                $\sigma(c) = \sigma(c) + 1$
            Fin de pour
        Fin de pour
    $F_k = \{c | c \in C_k$ et $\sigma(c) \geq N \times minsup\}$
Fin de répéter Résultat $= \cup F_k$

L'implémentation des fonctions `apriori-gen` et `subset` est décrite ci-dessous.

5.3.3 Génération des candidats et élagage

La fonction `apriori-gen` génère les ensembles d'items candidats en réalisant les deux opérations ci-dessous :

1. Génération des candidats.
2. Elagage des candidats.

Elle utilise la méthode $F_{k-1} \times F_1$. L'idée est d'étendre chaque ensemble d'items fréquents de profondeur $k-1$ en leur ajoutant d'autres items fréquents. Cette procédure est rapide et elle permet de retrouver tous les ensembles d'items fréquents de taille k, cependant elle peut générer plusieurs ensembles identiques. On ajoute alors une étape d'élagage (classement des items par ordre alphabétique, puis comparaison des ensembles).