

## Chapitre 2

# Recherche des modèles fréquents, corrélations et associations

Les motifs fréquents sont des motifs ou patterns (tel que les ensembles d'items, les sous séquences, ou les sous structures) qui apparaissent fréquemment dans un ensemble de données. Par exemple, un ensemble d'items tel que le lait et le pain qui apparaissent souvent dans une base de transactions dans un supermarché, est un ensemble d'items fréquent. Une sous séquence telle que acheter premièrement un PC puis une caméra numérique ensuite une carte mémoire qui se produit souvent dans la base historique des achats, est une séquence d'items fréquente. Les sous structures peuvent être des sous-graphes ou des sous-arbres qui peuvent être combinés avec des ensembles ou des séquences d'items. Trouver de tels motifs fréquents joue un rôle essentiel dans la fouille des associations et des corrélations, et représente une tâche importante en data mining et constitue toujours un thème qui attire beaucoup de recherches.

L'analyse des motifs fréquents trouve son application dans plusieurs domaines :

- L'analyse du panier du marché, pour comprendre les habitudes des clients afin de mieux organiser les rayons d'articles, organiser les promotions, ...etc.
- L'analyse d'ADN en biologie afin de comprendre les propriétés génétiques des espèces.
- L'analyse du climat en météorologie afin de mieux orienter l'agriculture ou choisir l'orientation des pistes des aéroports.
- ...

## 2.1 Concepts de base

### 2.1.1 Base de données formelle

La version de base de l'extraction de motifs fréquents permet de faire la fouille dans une table d'une base de données relationnelle dont les valeurs sont des booléens indiquant la présence ou l'absence d'une propriété. Une telle base est appelée base de données formelle.

Une base de données formelle est définie par un triplet  $(O, P, R)$  où :

- $O$  est un ensemble fini d'objets.
- $P$  est un ensemble fini de propriétés.
- $R$  est une relation sur  $O \times P$  qui permet d'indiquer si un objet  $x$  a une propriété  $p$  (noté  $xRp$ ) ou non.

Par exemple dans le cas d'analyse du panier dans un supermarché,  $O$  est l'ensemble des transactions d'achat,  $P$  est l'ensemble d'articles et  $R$  est la relation indiquant si un article  $a$  est acheté dans la transaction  $t$ .

Considérons par exemple la base de données formelle suivante :

$R$	a	b	c	d	e
$x_1$	1	0	1	1	0
$x_2$	0	1	1	0	1
$x_3$	1	1	1	0	1
$x_4$	0	1	0	0	1
$x_5$	1	1	1	0	1
$x_6$	0	1	1	0	1

- $O = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ .
- $P = \{a, b, c, d, e\}$ .
- $xRp$  si et seulement si la ligne de  $x$  et la colonne de  $p$  se croisent sur un 1 (et pas sur un 0), par exemple :  $x_1Ra, x_1Rc$  et  $x_1Rd$ .

### 2.1.2 Motif

Un motif d'une base de données formelle  $(O, P, R)$  est un sous-ensemble de  $P$ . L'ensemble de tous les motifs d'une base est donc l'ensemble des parties de  $P$ , noté  $2^P$ . On dira qu'un objet  $x \in O$  possède un motif  $m$  si  $\forall p \in m, xRp$ . Pour la base de données en exemple, on a donc :

- Motif de taille 0 =  $\emptyset$  ( $C_5^0 = 1$  motif).
- Motifs de taille 1 =  $\{a\}, \{b\}, \{c\}, \{d\}$  et  $\{e\}$ , qu'on notera, pour simplifier,  $\underline{a}, \underline{b}, \underline{c}, \underline{d}$  et  $\underline{e}$ . ( $C_5^1 = 5$  motifs).
- Motifs de taille 2 =  $\underline{ab}, \underline{ac}, \underline{ad}, \underline{ae}, \underline{bc}, \underline{bd}, \underline{be}, \underline{cd}, \underline{ce}, \underline{de}$  ( $C_5^2 = 10$  motifs)
- Motifs de taille 3 =  $\underline{abc}, \underline{abd}, \underline{abe}, \underline{acd}, \underline{ace}, \underline{ade}, \underline{bcd}, \underline{bce}, \underline{bde}, \underline{cde}$  ( $C_5^3 = 10$  motifs).
- Motifs de taille 4 =  $\underline{abcd}, \underline{abce}, \underline{abde}, \underline{acde}, \underline{bcde}$  ( $C_5^4 = 5$  motifs).
- Motifs de taille 5 =  $\underline{abcde}$  ( $C_5^5 = 1$  motifs).

Dans la base formelle précédente,  $x_1$  possède les motifs :  $\emptyset, \underline{a}, \underline{c}, \underline{d}, \underline{ac}, \underline{ad}, \underline{cd}$  et  $\underline{acd}$ . Parmi l'ensemble global de  $2^p$  motifs, on va chercher ceux qui apparaissent fréquemment. Pour cela, on introduira les notions de connexion de Galois et de support d'un motif.

### 2.1.3 Connexion de Galois

La connexion de Galois associée à une base de données formelle  $(O, P, R)$  est le couple de fonctions  $(f, g)$  définies par :

$$\left\{ \begin{array}{l} f : 2^p \rightarrow 2^o \\ m \rightarrow f(m) = \{x \in O / x \text{ possède } m\} \\ g : 2^o \rightarrow 2^p \\ X \rightarrow g(X) = \{p \in P / \forall x \in X \ xRp\} \end{array} \right.$$

$g$  est dite duale de  $f$  et  $f$  duale de  $g$ . On dit parfois que  $f(m)$  est l'image du motif  $m$ .

**Exemples :**

$$\begin{array}{ll} f(\underline{bc}) = \{x_2, x_3, x_5, x_6\} & g(x_2, x_3, x_5, x_6) = \{b, c, e\} \\ f(\underline{acd}) = \phi & g(x_1) = \{a, c, d\} \\ f(\underline{a}) = \{x_1, x_3, x_5\} & g(x_1, x_3, x_5) = \{a, c\} \end{array}$$

### 2.1.4 Support d'un motif

Soit  $m \in 2^p$ , un motif. Le support de  $m$  est la proportion d'objets dans  $O$  qui possèdent le motif :

$$\begin{array}{l} Support : 2^p \rightarrow [0, 1] \\ m \rightarrow Support(m) = \frac{|f(m)|}{|O|} \end{array}$$

Par exemple dans la base précédente, on a :

$$Support(\underline{a}) = \frac{3}{6},$$

$$\begin{aligned}
\text{Support}(\underline{b}) &= \frac{5}{6}, \\
\text{Support}(ab) &= \frac{2}{6}, \\
\text{Support}(\emptyset) &= 1, \\
\text{Support}(P) &= 0.
\end{aligned}$$

**Propriété fondamentale :** Le support est décroissant de  $(2^p, \subseteq)$  dans  $([0, 1], \leq)$ . Autrement dit, si  $m$  est un sous-motif de  $m'$  ( $m \subseteq m'$ ) alors  $\text{Support}(m) \geq \text{Support}(m')$ . Le support mesure la fréquence d'un motif : plus il est élevé, plus le motif est fréquent. On distinguera les motifs fréquents des motifs non fréquents à l'aide d'un seuil  $\sigma_s$ .

### 2.1.5 Motif fréquent

Soit  $\sigma_s \in [0, 1]$ . Un motif  $m$  est fréquent (sous-entendu, relativement au seuil  $\sigma_s$ ) si  $\text{Support}(m) \geq \sigma_s$ . Sinon, il est dit non fréquent.

## 2.2 Méthodes efficaces pour la recherche des modèles fréquents

Une approche naïve pour l'extraction des motifs fréquents consiste à parcourir l'ensemble de tous les motifs, à calculer leurs nombres d'occurrences (support) et à ne garder que les plus fréquents. Malheureusement, cette approche est trop consommatrice en temps et en ressources. En effet, le nombre de motifs est  $2^p$  ( $p$  est le nombre de propriétés), et en pratique, on veut manipuler des bases ayant un grand nombre d'attributs. L'algorithme Apriori proposé par Agrawal et ses co-auteurs en 1994 est un algorithme de base qui permet d'extraire des motifs fréquents dans une base ayant plusieurs milliers d'attributs et plusieurs millions d'enregistrements. L'idée est d'effectuer une extraction par niveaux selon le principe suivant :

- On commence par chercher les motifs fréquents de longueur 1 ;
- On combine ces motifs pour obtenir des motifs de longueur 2 et on ne garde que les fréquents parmi eux ;
- On combine ces motifs pour obtenir des motifs de longueur 3 et on ne garde que les fréquents parmi eux ;
- ... continuer jusqu'à la longueur maximale.

Cette approche s'appuie sur les deux principes fondamentaux suivants (qui reposent sur la décroissance du support) :

1. Tout sous-motif d'un motif fréquent est fréquent.
2. Tout sur-motif d'un motif non fréquent est non fréquent.

L'algorithme de référence basé sur cette approche est l'algorithme Apriori. Le pseudo-code suivant décrit l'extraction de motifs fréquents selon ce principe :

---

**Algorithme 1** Apriori

---

**ENTRÉES:** Base de données de transactions  $D$ , Seuil de support minimum  $\sigma$

**SORTIES:** Ensemble des items fréquents

$i \leftarrow 1$

$C_1 \leftarrow$  ensemble des motifs de taille 1 (un seul item)

**tantque**  $C_i \neq \phi$  **faire**

Calculer le Support de chaque motif  $m \in C_i$  dans la base

$F_i \leftarrow \{m \in C_i \mid \text{support}(m) \geq \sigma\}$

$C_{i+1} \leftarrow$  toutes les combinaisons possibles des motifs de  $F_i$  de taille  $i + 1$

$i \leftarrow i + 1$

**fin tantque**

retourner  $\cup_{(i \geq 1)} F_i$

---

**Exemple :**

L'application de l'algorithme sur la base donnée en exemple avec  $\sigma = 0.25$  se passe comme suit :

1. Génération de candidats de taille 1 :

–  $C_1 = \{a, b, c, d, e\}$

– Supports :  $\frac{3}{6}, \frac{5}{6}, \frac{5}{6}, \frac{1}{6}, \frac{5}{6}$

D'où  $F_1 = \{a, b, c, e\}$  (aucun motif fréquent ne contiendra  $d$ ).

2. Génération de candidats de taille 2 : Combiner 2 à 2 les candidats de taille 1 de  $F_1$  :

–  $C_2 = \{ab, ac, ae, bc, be, ce\}$

– Supports :  $\frac{2}{6}, \frac{3}{6}, \frac{2}{6}, \frac{4}{6}, \frac{5}{6}, \frac{4}{6}$

$F_2 = C_2$  : tous les motifs de  $C_2$  sont fréquents.

3. Génération de candidats de taille 3 : Combiner 2 à 2 les candidats de taille 2 de  $F_2$

(et ne considérer que ceux qui donnent des motifs de taille 3) :

–  $C_3 = abc, abe, ace, bce$

– Supports :  $\frac{2}{6}, \frac{2}{6}, \frac{2}{6}, \frac{4}{6}$

$F_3 = C_3$  : tous les motifs de  $C_3$  sont fréquents.

4. Génération de candidats de taille 4 :

–  $C4 = \{abce\}$

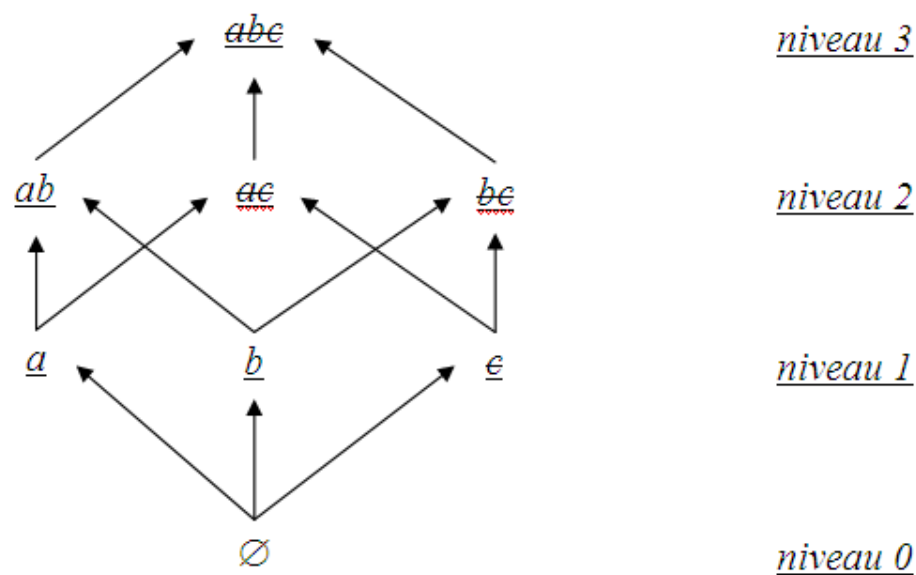
– Supports :  $\frac{2}{6}$

5. Génération de candidats de taille 5 :  $C5 = \emptyset$ . Donc,  $F5 = \emptyset$

6. L'algorithme retourne alors l'ensemble des motifs fréquents :  $F1 \cup F2 \cup F3 \cup F4$

**Remarque 1 :** On peut voir cet algorithme comme le parcours du treillis des parties de  $P$  ordonné pour l'inclusion.

Supposons par exemple que  $P = \{a, b, c\}$ . Le treillis des parties de  $P$  est :



Il est parcouru par niveau croissant à partir du niveau  $i = 1$ . Quand un motif n'est pas fréquent, tous ses sur-motifs sont non fréquents. Dans notre exemple  $c$  n'est pas fréquent (il a été barré) et, par conséquent, aucun de ses sur-motifs n'est considéré. On a ainsi élagué le parcours du treillis.

**Remarque 2 :** Un des objectifs des optimisations de cet algorithme est de diminuer le nombre d'accès à la base de données.

**Remarque 3 :** Le seuil  $\sigma$  est fixé par l'analyste. Celui-ci peut suivre une approche itérative en fixant un seuil au départ et, en fonction du résultat, changera la valeur du seuil : Si trop de motifs fréquents ont été trouvés, il augmentera le seuil ; dans le cas inverse, il le diminuera. Par ailleurs, on peut constater que le temps de calcul de l'algorithme Apriori décroît avec le seuil. Par conséquent, si l'analyste fixe une valeur de seuil trop grande, cela gaspillera moins de temps que s'il en fixe un trop petit.

## 2.2.1 Optimisations

### 2.2.1.1 L'algorithme AprioriTID

L'algorithme Apriori nécessite  $N$  passes sur la base,  $N$  étant la taille du plus grand ensemble susceptible d'être fréquent. Une optimisation possible consiste à générer en mémoire, pendant la première passe, les identifiants (TID) des transactions pour chaque 1-itemset (ensemble de motifs de taille 1) fréquent. Dans la suite, les listes de TID correspondant à chaque  $k$ -itemset sont gardées. Le calcul d'un  $k$ -itemset se fait toujours à partir des deux  $(k-1)$ -itemsets contenant un élément de moins, mais le comptage se fait simplement par intersection des deux listes de TID des deux  $(k-1)$ -itemsets source. Nous construisons la liste des TID après avoir déterminé les 1-itemsets fréquents, ce qui est plus efficace en taille mémoire mais nécessite deux passes. La première passe permet d'éliminer les produits non fréquents et réduit donc les listes de TID en mémoire construites dans une deuxième passe. Générer les listes de TID en mémoire en parallèle dès la première passe est plus efficace. Cependant, il n'est possible d'éliminer des listes qu'après le comptage total de la base, lorsqu'on sait qu'un 1-itemset ne sera pas fréquent. L'algorithme nécessite alors une taille mémoire de  $N \cdot P$  TID,  $N$  étant le nombre de transactions et  $P$  le nombre de produits. Il devient très inefficace si les listes ne tiennent pas en mémoire.

### 2.2.1.2 L'algorithme apriori partitionné

L'algorithme proposé par Savasere permet de résoudre le problème de place mémoire de l'algorithme précédent. L'idée est simplement de diviser la base en  $Q$  partitions, de sorte que chaque partition tienne en mémoire. Chaque partition est traitée indépendamment et les ensembles fréquents sont découverts pour chaque partition. La validité de l'algorithme est basée sur le lemme suivant : pour être globalement fréquent, un ensemble doit être fréquent dans au moins une partition. Ayant obtenu les ensembles fréquents sur chaque partition, il suffit dans une passe finale d'explorer l'union des ensembles fréquents sur la base. L'opération se fait par un comptage simple sur la base. Les ensembles non globalement fréquents, mais localement fréquents dans une partition, sont ainsi éliminés. L'avantage de cet algorithme est qu'il nécessite deux passes au plus. Il est aussi facilement parallélisable, les partitions pouvant être traitées indépendamment. Dans ce cas, il faut cependant beaucoup de mémoire pour tenir les partitions et listes de TID en mémoire.

### 2.2.1.3 Algorithme de comptage dynamique

L'algorithme DIC a été proposé par Brin et al. Pour réduire le nombre de parcours de la base de données. DIC partitionne la base de données en blocs de  $M$  transactions. Durant le calcul des supports des  $k$ -itemsets, après le parcours d'une partition de taille  $M$  de  $D$ , on vérifie les  $k$ -itemsets candidats qui ont déjà atteint le support minimum, DIC les utilise alors pour générer des candidats de taille  $(k+1)$ , et commence à compter leurs supports. Ainsi les supports de candidats de tailles différentes sont calculés durant les mêmes parcours de  $D$ . En contrepartie de la diminution du nombre de balayages de la base de données, DIC considère des itemsets candidats de tailles différentes simultanément. Ceci pose le problème de stockage des itemsets candidats traités simultanément et du coût de calcul des supports des candidats qui est plus important que pour Apriori.

## 2.3 Types de motifs fréquents

Le nombre de motifs présents dans une base de données est généralement très grand, il est courant d'en obtenir plusieurs millions. Leur utilisation repose classiquement sur une démarche de sélection par la fréquence à fin de ne conserver que les plus représentatifs, mais aussi parce que c'est une façon pragmatique efficace d'élaguer l'espace de recherche. Néanmoins cette approche s'avère insuffisante dans de nombreux cas pratiques et montre ses limites lorsque l'espace de recherche est important ou la base très dense et corrélée.

Les représentations condensées de motifs fournissent une solution au problème de l'extraction de motifs fréquents en proposant un résumé des motifs fréquents, tout en assurant de pouvoir reconstruire l'ensemble des motifs fréquents si nécessaire. En général, on en trouve deux types :

### 2.3.1 Motif fréquent fermé

Un motif fréquent est dit fermé s'il ne possède aucun sur-motif qui a le même support.

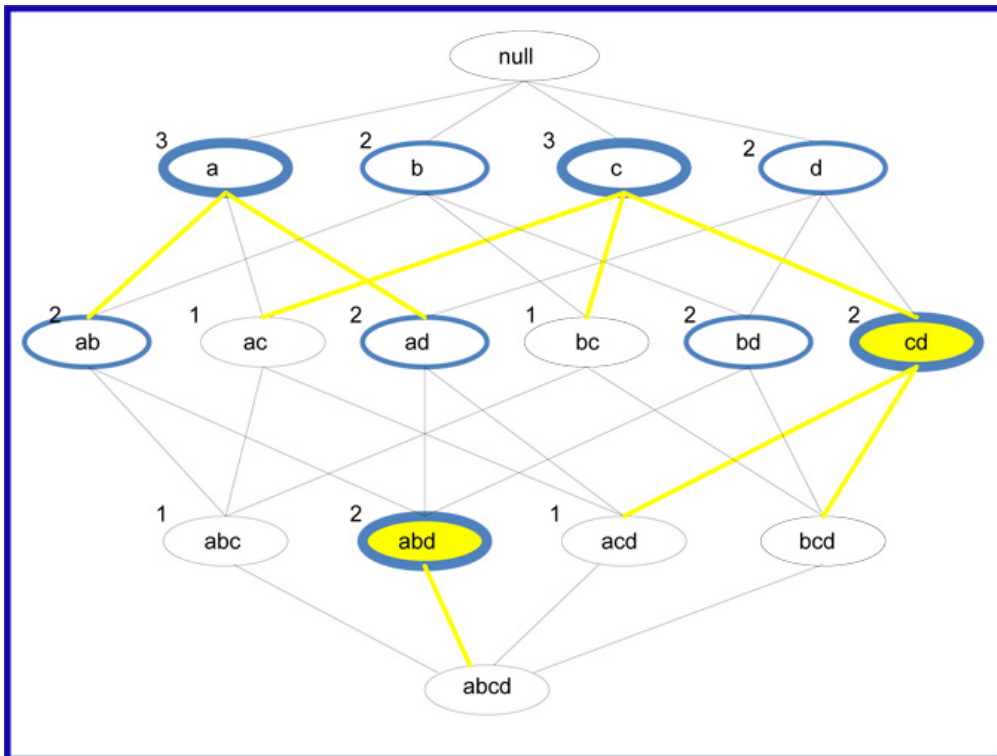
### 2.3.2 Motif fréquent maximal

Un motif fréquent est dit Maximal si aucun de ses sur-motifs immédiats n'est fréquent.

#### Exemple

Le schéma suivant illustre la relation entre les motifs fréquents, fréquents fermés et fréquents maximaux :





- Les motifs encerclés par les lignes minces ne sont pas fréquents, les autres le sont.
- Les motifs encerclés par des lignes plus épais sont fermés.
- Les motifs colorés sont maximaux.

Il est clair que :

Les motifs maximaux  $\subset$  Les motifs fermés  $\subset$  Les motifs fréquents

## 2.4 Passage aux règles d'association

La phase qui suit la phase de recherche des motifs fréquents est la phase de découverte des règles d'association basées sur tous les items fréquents trouvés. Cette phase est relativement simple, elle consiste à trouver toutes les règles qui existent entre les items fréquents. Par exemple pour une règle telle que  $\{x_1, x_2, x_3\} \Rightarrow x_4$ , il faut premièrement que  $\{x_1, x_2, x_3\}$  et  $x_4$  soient fréquents. Ensuite, il faut que la confiance de la règle dépasse un certain seuil. La confiance est calculée comme suit :

$$\begin{aligned}
 \text{Confidence}(\{x_1, x_2, x_3\} \Rightarrow x_4) &= P(x_4 / \{x_1, x_2, x_3\}) \\
 &= \frac{\text{Support}(\{x_1, x_2, x_3\} \cup x_4)}{\text{Support}(\{x_1, x_2, x_3\})}
 \end{aligned}$$

Où le  $Support(x)$  est le nombre d'enregistrements où apparaît l'item  $x$ ,  
 et le  $Support(\{x_1, x_2, x_3\})$  est le nombre d'enregistrement où apparaissent les items  $x_1, x_2, x_3$   
 ensemble.

L'ensemble des règles d'association peut être trouvé en calculant la confiance de toutes  
 les combinaisons possibles des items fréquents puis prendre celles dont la confiance est  
 importante. Cette opération peut être accélérée en enregistrant la liste des items fréquents  
 avec leurs fréquences dans une table qui peut être accédée rapidement.

### Définition d'une règle d'association

Soit  $m_1$  et  $m_2$  deux motifs. Une règle d'association est une implication de la forme :

$$m_1 \Rightarrow m_2$$

$$\text{Où } m_1, m_2 \in 2^P, \text{ et } m_1 \cap m_2 = \emptyset$$

#### 2.4.0.1 Support d'une règle

La règle  $m_1 \Rightarrow m_2$  est vérifiée dans la base de donnée  $D$  avec un support  $s$ , où  $s$  est le  
 pourcentage d'objets dans  $D$  contenant  $m_1 \cup m_2$  :

$$Support(m_1 \Rightarrow m_2) = \frac{\text{Nombre de transactions contenant}(m_1 \cup m_2)}{\text{Nombre total de transactions}}$$

#### 2.4.0.2 Qualité d'une règle

La qualité de la règle représente une mesure de la coexistence des deux parties (gauche  
 et droite) de la règle. Plusieurs métriques sont utilisées :

- **Confidence** : La confiance de la règle  $m_1 \Rightarrow m_2$  est définie par le pourcentage de  
 transactions qui contiennent  $m_1 \cup m_2$  dans les transaction contenant  $m_1$ .

$$Conf(m_1 \Rightarrow m_2) = \frac{Support(m_1 \cup m_2)}{Support(m_1)} = \frac{\text{Nombre de transactions contenant}(m_1 \cup m_2)}{\text{Nombre de transactions contenant } m_1}$$

- **Lift** : c'est l'amélioration

$$Lift(m_1 \Rightarrow m_2) = \frac{Support(m_1 \cup m_2)}{Support(m_1) \times Support(m_2)}$$

C'est le rapport entre le support observé et celui attendu si  $m_1$  et  $m_2$  étaient indé-  
 pendants.

Si le Lift d'une règle est égal à 1, cela signifie que les occurrences de ses deux parties sont totalement indépendantes. Si par contre le lift est supérieur à 1, il nous informe du degré de dépendance des deux parties et leur candidature pour les futures prédictions.

Le lift considère à la fois la confiance de la règle et la base toute entière.

- **Leverage** : Il représente la différence entre la fréquence observée d'apparition des deux parties ensemble et leur apparition ensemble attendues s'il étaient indépendants.

$$Leverage(m_1 \Rightarrow m_2) = Support(m_1 \cup m_2) - Support(m_1) \times Support(m_2)$$

- **Conviction** :

La conviction est calculé par la formule :

$$Conv(m_1 \Rightarrow m_2) = \frac{1 - Support(m_2)}{1 - Conf(m_1 \Rightarrow m_2)}$$

Elle représente le rapport entre la fréquence d'apparition de X sans Y (taux d'erreur de la règle), et le taux des prédictions erronés.

Les règles qui dépassent un minimum de support et un minimum de qualité sont appelées règles **solides**.

Une fois les items fréquents dans une base de donnée sont extraits, il devient simple de générer les règles d'association qui vérifient un minimum de support et un minimum de confiance, comme suit :

- Pour chaque motif fréquent l, générer tous les sous ensembles non vides de l ,
- Pour chaque sous-ensemble non vide s de l, enregistrer la règle (s  $\Rightarrow$  l-s) si :

$$Confidence(s \Rightarrow l - s) \geq Min\_Conf$$

Où Min\_Conf est un seuil minimum de confiance.

Puisque les règles sont générées des motifs fréquents, chacune vérifie automatiquement le support minimum.

## 2.5 Analyse des corrélation

La plupart des algorithmes de recherche des règles d'association utilise un seuil minimum de confiance. Cependant, plusieurs règles intéressantes peuvent être trouvées en

utilisant un seuil très faible. Bien que l'utilisation d'un support minimum empêche l'exploration d'un grand nombre de règle intéressantes, plusieurs règles ainsi trouvées restent inutiles pour l'utilisateur.

On s'intéresse maintenant à la recherche de paires d'items dont le support est faible mais dont la présence est fortement corrélée. Les données peuvent être vues comme une matrice  $X = (x_{i,j})$  dont chaque ligne représente un individu et chaque colonne représente un item. Typiquement la matrice  $x_{i,j}$  est très creuse : pour chaque individu, il n'y qu'une faible proportion d'items présents (souvent bien moins qu'1 %). On cherche des paires de produits souvent présents ensemble, *i.e.* des colonnes similaires. Le problème à résoudre est donc de trouver le plus efficacement possibles ces paires de colonnes similaires, en faisant l'hypothèse que les items apparaissent rarement (support faible). On peut essayer d'utiliser l'algorithme A-Priori vu plus haut mais il est mal adapté à la recherche d'items de support faible ; on peut aussi essayer d'imaginer des méthodes plus spécifiques. C'est ce que nous décrivons ci-dessous. On suppose par ailleurs que :

- P est suffisamment petit pour que l'on puisse stocker une information (un nombre) concernant chaque colonne en mémoire centrale, mais on ne peut pas stocker en mémoire centrale pour chaque paire d'attributs ;
- O est tellement grand que l'on ne peut pas stocker toute la matrice en mémoire centrale, même en tenant compte du fait que la matrice est creuse, et même en la compressant ;
- on cherche une alternative à l'utilisation des critères utilisés plus haut (support, confiance) car l'ensemble de paires candidates est trop grand pour être énuméré en un temps raisonnable.

### 2.5.1 Calcul de la corrélation

On commence par préciser ce que l'on entend par "corrélation" de deux colonnes. Informellement, deux colonnes sont similaires si elles ont généralement des 1 aux mêmes lignes. On mesure la corrélation de deux colonnes  $x_{.,j}$  et  $x_{.,k}$  par le rapport entre le nombre de lignes où  $x_{i,j}$  et  $x_{i,k}$  sont égaux à 1 en même temps par le nombre de lignes où l'un des deux seulement vaut 1 :

$$Cor(x_{.j}, x_{.k}) = \frac{|x_{i,j} \wedge x_{i,k}|}{|x_{i,j} \vee x_{i,k}|}$$

La Complexité du calcul de Cor : (O) pour deux colonnes données ; il y a  $O(P^2)$  paires de colonnes ; donc  $O(O \times P^2)$  pour l'ensemble des données.

En prenant  $O = 106$  et  $P = 105$ ,  $OP^2 = 10^{16}$ . Sur un ordinateur capable d'effectuer 1 million de comparaisons par seconde, le calcul de Cor demande 1010 secondes, soit 10 milliards de secondes, soit ... 300 ans !

La signature permet de surmonter ce problème de complexité. On définit le type de deux colonnes pour une ligne donnée par a, b, c ou d comme suit :

Type	$x_{i,j}$	$x_{i,k}$
a	1	1
b	1	0
c	0	1
d	0	0

Pour une paire de colonnes, On note respectivement a, b, c et d le nombre de lignes de type a, b, c et d. On a :

$$Cor(x_{.,j}, x_{.,k}) = \frac{|x_{i,j} \wedge x_{i,k}|}{|x_{i,j} \vee x_{i,k}|} = \frac{a}{a + b + c + d}$$

Les applications de cette recherche de paires fortement corrélées à faible support sont :

- lignes et colonnes sont des pages web et  $x_{i,j} = 1$  si la page  $i$  pointe sur la page  $j$ .  
Dans ce cas, des colonnes similaires peuvent être des pages traitant d'un même sujet : ces pages sont pointées par les mêmes pages ;
- lignes et colonnes sont des pages web et  $x_{i,j} = 1$  si la page  $j$  pointe sur la page  $i$ .  
Dans ce cas, des colonnes similaires sont des pages miroirs ;
- les lignes sont des pages web ou des documents, les colonnes sont des mots. Des colonnes similaires indiquent des mots qui apparaissent souvent ensemble dans les mêmes pages ;
- les lignes sont des pages web ou des documents, les colonnes sont des phrases. Les colonnes similaires indiquent des pages miroir ou des plagiats.

## 2.6 Motifs rares

Les motifs rares représentent les motifs qui apparaissent rarement dans un ensemble de données tel que les symptômes non usuels ou les effets indésirables exceptionnels qui peuvent se déclarer chez un patient pour une pathologie ou un traitement donné.

De même que pour les motifs fréquents, certains phénomènes rares dans les bases de données peuvent également véhiculer des connaissances.

La découverte des motifs rares peut se révéler très intéressante, en particulier en médecine et en biologie. Prenons d'abord un exemple simulé d'une base de données médicale où nous nous intéressons à l'identification de la cause des maladies cardio-vasculaires (MCV). Une règle d'association fréquente telle que "niveau élevé de cholestérol  $\rightarrow$  MCV" peut valider l'hypothèse que les individus qui ont un fort taux de cholestérol ont un risque élevé de MCV. A l'opposé, si notre base de données contient un grand nombre de végétariens, une règle d'association rare "végétarien  $\rightarrow$  MCV" peut valider l'hypothèse que les végétariens ont un risque faible de contracter une MCV. Dans ce cas, les motifs végétarien et MCV sont tous deux fréquents, mais le motif végétarien, MCV est rare. Un autre exemple est en rapport avec la pharmacovigilance, qui est une partie de la pharmacologie dédiée à la détection et l'étude des effets indésirables des médicaments. L'utilisation de l'extraction des motifs rares dans une base de données des effets indésirables des médicaments pourrait contribuer à un suivi plus efficace des effets indésirables graves et ensuite à prévenir les accidents fatals qui aboutissent au retrait de certains médicaments.

### 2.6.1 Définitions

Un motif est dit rare ou infrequent si son support est inférieur ou égal à un support maximum (noté  $max\_supp$ ).

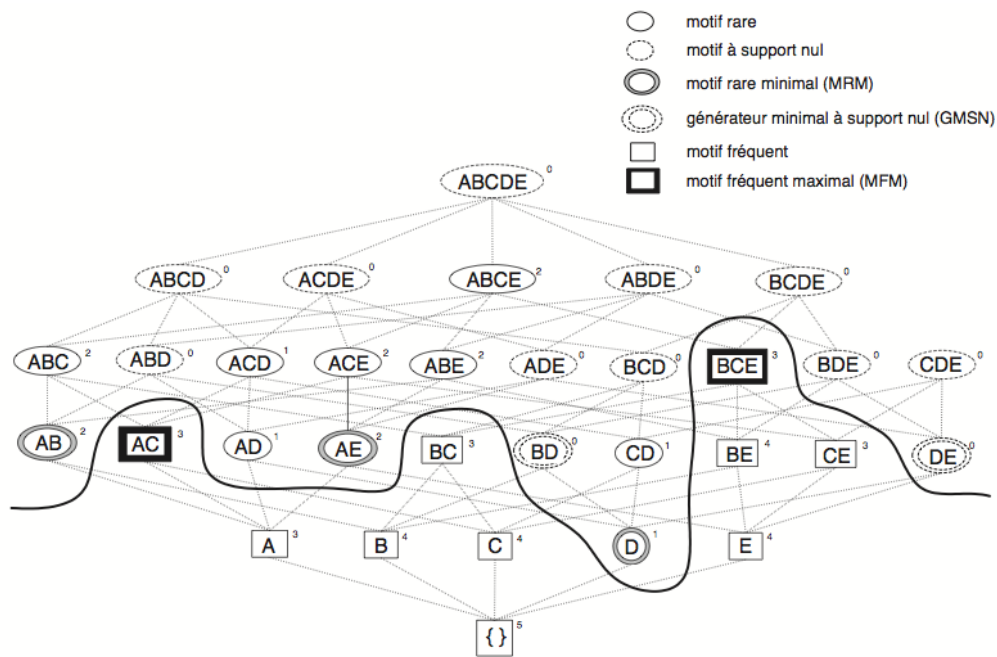
Généralement, on considère que  $max\_supp = min\_supp - 1$ , c'est-à-dire qu'un motif est rare s'il n'est pas fréquent. Cela implique l'existence d'une seule frontière entre motifs rares et fréquents.

### 2.6.2 Recherche des motifs rares

Une première approche pour la recherche des motifs rares est celle de treillis. Prenons l'exemple de la base suivante :

	A	B	C	D	E
1	X		X	X	
2		X	X		X
3	X	X	X		X
4	X				X
5	X	X	X		X

En fixant  $min\_supp$  à 3 ( $max\_supp = 2$ ), On obtient le treillis suivant :



Les motifs peuvent être séparés en deux ensembles formant une partition : les motifs rares et les motifs fréquents. Une frontière peut être dessinée entre ces deux ensembles. L'ensemble des motifs rares et l'ensemble des motifs fréquents ont tous deux un sous-ensemble minimal générateur. Dans le cas des motifs fréquents, ce sous-ensemble est appelé ensemble des motifs fréquents maximaux (MFM).

Ces motifs sont dits maximaux, parce qu'ils n'ont pas de sur-motifs fréquents. Du point de vue du nombre de ces motifs ils sont minimaux, c'est-à-dire qu'ils forment un ensemble générateur minimal à partir duquel tous les motifs fréquents peuvent être retrouvés.

Les motifs rares minimaux (MRM) sont définies en tant que complémentaires des MFMs. Un motif est un MRM s'il est rare (et ainsi tous ses sur-motifs sont rares) et si tous ses sous-motifs ne sont pas rares. Ces motifs forment un ensemble générateur minimal à partir duquel tous les motifs rares peuvent être retrouvés.

Ces motifs forment un ensemble générateur minimal à partir duquel on peut retrouver les motifs rares. Nous devons d'abord générer tous les sur-motifs possibles des motifs rares minimaux, puis calculer le support des motifs rares grâce à un passage sur la base de données.

### 2.6.3 Apriori-Rare

De manière surprenante, les motifs rares minimaux peuvent être trouvés simplement à l'aide de l'algorithme bien connu Apriori. Il est conçu pour trouver les motifs fréquents,

mais, puisque nous sommes dans le cas où non fréquent signifie rare, cela a pour "effet collatéral" d'explorer également les motifs rares minimaux. Quand Apriori trouve un motif rare, il ne générera plus tard aucun de ses sur-motifs car ils sont de manière sûre rares. Puisque Apriori explore le treillis des motifs niveau par niveau du bas vers le haut, il comptera le support des motifs rares minimaux. Ces motifs seront élagués et plus tard l'algorithme peut remarquer qu'un candidat a un sous-motif rare. En fait Apriori vérifie si tous les  $(k-1)$ -sous-motifs d'un  $k$ -candidat sont fréquents. Si l'un d'entre eux n'est pas fréquent, alors le candidat est rare. Autrement dit, cela signifie que le candidat a un sous-motif rare minimal. Grâce à cette technique d'élagage, Apriori peut réduire significativement l'espace de recherche dans le treillis des motifs. Une légère modification d'Apriori suffit pour conserver les MRM. Si le support d'un candidat est inférieur au support minimum, alors à la place de l'effacer nous l'enregistrons dans l'ensemble des motifs rares minimaux

Tous les motifs rares sont retrouvés à partir des motifs rares minimaux. Pour cela nous avons besoin de générer tous les sur-motifs possibles des MRM.