Data Mining 2 – Modélisation Statistique et Apprentissage
Master Statistique & Econométrie
UT1 Sciences Sociales, Toulouse
E-thème 6 **16 mars 2018**
Enseignant: Gilles Cohen

Règles d'association

1 Introduction

Beaucoup d'entreprises accumulent d'énormes quantités de données à partir de leurs opérations journalières. Par exemple, de grandes quantités de données concernant les achats des clients sont recueillies quotidiennement lors de leur passage en caisse. Le tableau 1 illustre un exemple de ces données, communément appelés les transactions du panier de la ménagère. Chaque ligne de ce tableau correspond à une transaction, qui contient un identifiant unique marqué TID et un ensemble d'articles (set of items) achetés par un client donné. Les détaillants sont intéressés par l'analyse de ces données pour en apprendre davantage sur le comportement d'achat de leurs clients. Ces informations précieuses peuvent être utilisées dans une grande variété d'applications métiers telles que les promotions de marketing, la gestion des stocks et la gestion de la relation client. Cet ethème présente une méthodologie connue sous le nom d'analyse des associations, qui est utile pour la découverte de relations intéressantes cachées dans de grands ensembles de données. Les relations peuvent être représentés sous la forme de règles d'association ou ensembles fréquents d'articles.

TID	Articles
t1	{Pain, Lait}
t2	{Pain, Couches, Bière, Oeufs}
t3	{Lait, Couches, Bière, Coca-Cola}
t4	{Pain, Lait, Couches, Bière}
t5	{Pain, Lait, Couches, Coca-Cola}

TABLE 1 – Représentation binaire

Par exemple, la règle suivante peut être extraite de l'ensemble de données

indiqué dans le tableau 1 :

$$\{\text{Couches}\} \rightarrow \{\text{Bière}\}.$$

La règle suggère qu'il existe une forte corrélation entre la vente de couches et celle de la bière parce que beaucoup de clients qui achètent des couches achètent aussi de la bière. Les détaillants peuvent utiliser ce type de règles pour les aider à identifier de nouvelles opportunités pour la vente croisée de leurs produits aux clients. En dehors des données du panier de la ménagère, l'analyse des associations est également applicable à d'autres domaines tels que la bio-informatique, le diagnostic médical, le Web mining et l'analyse des données scientifiques.

Dans l'analyse des données de sciences de la Terre, par exemple, les motifs (pattern) d'association peuvent révéler des liens intéressants entre l'océan, la terre et les processus atmosphériques. Ces informations peuvent aider les géologues à développer une meilleure compréhension de la façon dont les différents éléments du système terrestre interagissent les uns avec les autres. Même si les techniques présentées ici sont généralement applicables à une plus grande variété de jeux de données, à des fins d'illustration, notre discussion se concentrera principalement sur les données du panier de la ménagère. Il y a deux problèmes essentiels dans l'analyse des associations des données du panier de la ménagère. Le premier, découvrir des motifs dans un ensemble de données de transaction de grande taille peut être coûteux en calcul. Deuxièmement, certains des modèles découverts sont potentiellement faux car ils peuvent simplement arriver par hasard. Le reste de ce chapitre est organisé autour de ces deux problèmes.
La première partie du chapitre est consacrée à l'explication des concepts de base de l'analyse des associations et les algorithmes utilisés pour exploiter efficacement ces modèles. La deuxième partie du chapitre traite de la question de l'évaluation des modèles découverts afin d'éviter la génération de résultats erronés.

2 Définition du problème

Cette section passe en revue la terminologie de base utilisée dans l'analyse des associations et présente une description formelle de la tâche. Les données du panier de la ménagère peuvent être représentés dans un format binaire, comme indiqué dans le tableau 2, où chaque ligne correspond à une transaction et chaque colonne correspond à un élément. Un article (item) peut être considéré comme une variable binaire dont la valeur vaut un si

l'article est présent dans une transaction et zéro autrement. Parce que la présence d'un article dans une transaction est souvent considéré comme plus important que son absence, un élément est une variable binaire asymétrique.

TID	Pain	Lait	Couches	Bière	Oeufs	Coca-cola
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 2 – Représentation binaire

Ensemble d'articles (itemset) et support

Soit $I = \{i_1, i_2, \ldots, i_d\}$ l'ensemble de tous les articles (items) du panier de la ménagère et $T = \{t_1, t_2, \ldots, t_N\}$ l'ensemble de toutes les transactions. Chaque transaction t_i contient un sous-ensemble d'articles pris dans I.
Dans l'analyse des associations une collection de zéro ou plusieurs articles est appelé un ensemble d'articles (itemset). Si un itemset contient k éléments ou articles , il est appelé un k-itemset ou ensemble de k-articles. Par exemple, {Bière, Couches, Lait} est un exemple de 3itemset. L'ensemble vide est un itemset qui ne contient aucun éléments ou articles. La taille ou longueur de la transaction est définie comme étant le nombre d'éléments présents dans une transaction. Une transaction t_j est dite contenir un itemset X si X est un sous-ensemble de t_j.
Par exemple, la seconde transaction de la table 1 contient l'itemset {Pain, Couches} mais pas {Pain, Lait}. Une importante propriété d'un itemset est son support qui fait référence au nombre de transactions qui contiennent un itemset particulier.
Mathématiquement, le support, $\sigma(X)$, pour un itemset X peut-être définie comme :

$$\sigma(X) \quad = \quad |\{t_i | X \subseteq t_i \in T\}|$$

où le symbole |.| dénote le nombre d'éléments d'un ensemble. Dans le jeu de données du tableau 1, le support pour {Bière, Couches, Lait} est égal à deux puisqu'il n'y a que deux transactions qui contiennent les trois articles.

Règles d'association

Une règle d'association est une implication de la forme $X \rightarrow Y$, où X et Y sont des ensembles d'itemset disjoints, i.e. $X \cap Y = \emptyset$. La force d'une règle d'association peut être mesurée en termes de support et de confiance.

Le support détermine combien de fois une règle s'applique à un ensemble de données, tandis que la confiance détermine la fréquence d'apparition des items de Y dans les transactions qui contiennent X. Les définitions formelles de ces mesures sont :

$$\text{Support }, s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N} \quad (1)$$

$$\text{Confiance }, c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \quad (2)$$

Exemple : On considère la règle {Lait, Couches} $\rightarrow$ {Bière}. Puisque le nombre support pour {Lait, Couches, Bière} est 2 et que le nombre total de transactions est 5, le support de la règle est $2/5 = 0.4$. La confiance de la règle est obtenue en divisant le nombre support pour {Lait, Couches, Bière} par le nombre support pour {Lait, Couches}. Comme il y a 3 transactions qui contiennent lait et couches, la confiance pour cette règle est $2/3 = 0.67$.

2.1 Formulation du problème de la recherche des règles d'association.

Le problème de la fouille des règles d'association peut-être formellement défini comme suit :

Définition 2.1 (Découverte des règles d'association) *Etant donné un ensemble de transaction T, trouver toutes les règles avec un support $\geq$ minsup et une confiance $\geq$ minconf, où minsup et minconf sont les seuils du support et de la confiance correspondants.*

Une approche "force brute" pour la fouille de règles d'association consiste à calculer le support et la confiance pour toutes les règles possibles. Le coût d'une telle approche est prohibitif en terme de temps de calcul car il y a un nombre exponentiel de règles qui peuvent être extraites à partir d'un ensemble de données. Plus précisément, le nombre total de règles possibles extraites d'un ensemble de données qui contient d articles est

$$R = 3^d - 2^{d+1} + 1$$

Même pour le petit ensemble de données du tableau 1, appliquer cette approche revient à calculer le support et la confiance pour $3^6 - 2^7 + 1 = 602$ règles.
Plus de 80% des règles sont écartées après l'application des seuils minsup = 20% et minconf = 50%, rendant ainsi la plupart des calculs réalisés inutiles. Afin d'éviter d'effectuer des calculs inutiles , il serait utile d'élaguer

les règles au début sans avoir à calculer les valeurs de leur support et de leur confiance. Une première étape vers l'amélioration de la performance des algorithmes d'extraction de règles d'association est de découpler les exigences du support et de la confiance. De l'équation 1, on note que le support d'une règle $X \rightarrow Y$ dépend seulement du support de ses itemsets, $X \rightarrow Y$.
Par exemple, les règles suivantes ont un support identique parce qu'elles impliquent des items du même itemset,
{Bière, Couches, Lait} :

$$\{\text{Bière, Couches}\} \rightarrow \{\text{Lait}\}, \{\text{Bière, Lait}\} \rightarrow \{\text{Couches}\},$$
$$\{\text{Couches, Lait}\} \rightarrow \{\text{Bière}\}, \{\text{Bière}\} \rightarrow \{\text{Couches, Lait}\},$$
$$\{\text{Lait}\} \rightarrow \{\text{Bière,Couches}\}, \{\text{Couches}\} \rightarrow \{\text{Bière,Lait}\}.$$

Si l'itemset est non fréquent, alors les règles candidates peuvent être élaguées immédiatement sans avoir à calculer leur valeur de confiance. Ainsi, une stratégie commune adoptée par beaucoup d'algorithmes de fouilles de règles d'association est de décomposer le problème en deux sous tâches principales :

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

2.2 Génération fréquent itemset

Une structure de treilli peut être utilisée pour énumérer la liste de tous les ensembles d'articles possibles. La Figure 1 montre un treilli d'ensemble d'articles (itemset) pour $I = \{a, b, c, d, e\}$. En général, un ensemble de données qui contient k articles (items) peut potentiellement générer jusqu'à $2^k - 1$ fréquents itemsets, excluant l'ensemble vide.
Parce que k peut être très grand dans de nombreuses applications pratiques, l'espace de recherche de l'ensemble des articles qui doit être exploré est exponentiellement grand. Une approche brute pour trouver les fréquents itemsets est de déterminer le nombre support pour chaque itemset candidat dans la structure du treilli. Pour ce faire, nous avons besoin de comparer chaque candidat contre chaque transaction.
Si le candidat fait partie d'une transaction, son nombre support sera incrémenté. Par exemple, le support pour {Pain, Lait} est incrémenté trois fois

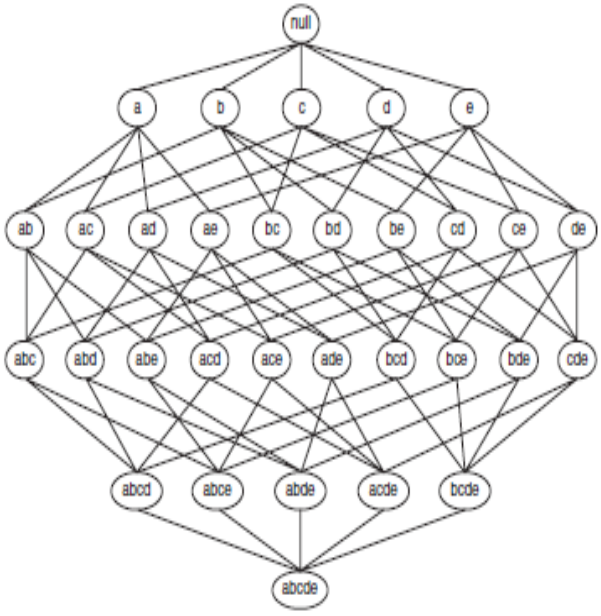

FIGURE 1 – Un treilli d'itemset

parce que l'itemset est contenue dans les transactions 1, 4, et 5. Une telle approche peut être très coûteuse, car elle nécessite $O(NMw)$ comparaisons, où N est le nombre de transactions, $M = 2^k - 1$ est le nombre d'ensembles d'articles candidats, et w est la taille maximale de la transaction. Il y a plusieurs façons de réduire la complexité du calcul de la génération de fréquents d'itemsets.

1. Réduire le nombre d'itemsets candidats (M). Le principe Apriori, décrit dans la section suivante, est un moyen efficace pour éliminer certains des itemsets candidats sans calculer leur valeur de support.
2. Réduire le nombre de comparaisons. Au lieu de faire correspondre chaque itemset candidat contre toutes les transactions, nous pouvons réduire le nombre de comparaisons en utilisant des structures de données plus avancées, que ce soit pour stocker les itemsets candidats ou pour compresser l'ensemble de données.

On va se limiter au principe Apriori qui fait l'objet de la section suivante.

2.2.1 Le principe d'Apriori

Théorème 2.2 (Principe Apriori) *Si un itemset est fréquent, alors tous ses sous-ensembles sont également fréquents.*

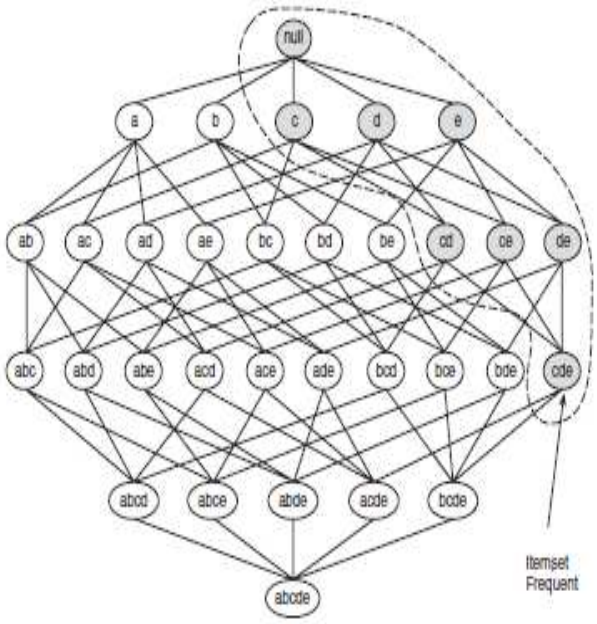

FIGURE 2 – Illustration du principe de l'approche Apriori. Si $\{c, d, e\}$ est fréquent, alors tous les sous-ensemble de cet itemset sont fréquents.

Pour illustrer l'idée derrière le principe Apriori, on considère le treillis des itemsets de la figure 1. On suppose que $\{c, d, e\}$ est un fréquent itemset. Clairement, chaque transaction qui contient $\{c, d, e\}$ doit aussi contenir ses sous-ensembles, $\{c, d\}, \{c, e\}, \{d, e\}, \{c\}, \{d\}, et\{e\}$. Et donc comme résultat, si $\{c, d, e\}$ est fréquent, alors tous les sous-ensembles de $\{c, d, e\}$ (i.e., les itemsets grisés dans la figure) doivent également être fréquent. Inversement, si un itemset tel que a, b est non fréquent ou rare, alors l'ensemble de ses sur-ensembles doit être non fréquent aussi. Comme le montre la figure 2, le sous-graphe entier contenant les sur-ensembles de $\{a, b\}$ peut être élagué immédiatement après que $\{a, b\}$ ait été trouvé non fréquent. Cette stratégie de taille de l'espace de recherche exponentielle en fonction de la mesure de support est connu comme élagage basé sur le support. Une telle stratégie d'élagage est rendue possible par une propriété clé de la mesure support,

à savoir que le support pour itemset ne dépasse jamais le support de ses sous-ensembles. Cette propriété est également connue comme la propriété antimonotone de la mesure de support.

Définition 2.3 (Propriété de monotonicité) *Soit I un ensemble d'articles, et $J = 2^I$ l'ensemble puissance de I. Une mesure f est monotone si*

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

ce qui signifie que si X est un sous-ensemble de Y, alors $f(X)$ ne doit pas excéder $f(Y)$. D'un autre coté, f est antimonotone si

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

ce qui signifie que si X est un sous-ensemble de Y , alors $f(Y)$ ne doit pas excéder $f(X)$.

Toute mesure qui possède une propriété antimonotone peut-être incluse directement dans l'algorithme de fouille afin d'élaguer efficacement l'espace de recherche exponentiel des itemsets candidats.

2.2.2 Génération des itemsets fréquents dans l'algorithme Apriori

Le pseudo-code de l'algorithme Apriori pour la partie génération des itemsets est illustré dans l'Algorithme 1.
Soit C_k l'ensemble des candidats k-itemsets et F_k l'ensemble des k-itemsets fréquents :

- L'algorithme réalise initialement une simple passe sur l'ensemble des données pour déterminer le support de chaque item. Après la fin de cette étape, l'ensemble de tous les fréquent 1-itemsets, F_1 , sera connu (étapes 1 et 2).
- Ensuite, l'algorithme générera itérativement les k-itemsets candidats en utilisant les fréquents $(k-1)$itemsets trouvés à l'itération précédente (étape 5). La génération des candidats est implementée en utilisant une fonction appelé apriori gen, qui est décrite plus loin.
- Pour compter le support des candidats, l'algorithme a besoin de faire une passe additionnelle sur l'ensemble de données (étape 6–10).
- Après avoir calculé leur support, l'algorithme élimine tous les itemsets candidats dont le nombre support est inférieur au seuil minsup (étape 12).

— L'algorithme termine lorsqu'il n'y a plus de fréquents itemsets générés, i.e., $F_k = \emptyset$ (étape 13).

La partie génération de l'algorithme Apriori des fréquents itemsets a deux caractéristiques importantes. Premièrement, c'est un algorithme par niveau (levelwise) ; i.e., il traverse le treilli des itemset un niveau à la fois, depuis les fréquents 1-itemsets jusqu'à la taille maximale des fréquents itemsets. Deuxièmement, il emploie une stratégie de génération et test pour trouver les fréquents itemsets. A chaque itération, les nouveaux itemsets candidats sont générés à partir des fréquents itemsets trouvés dans la précédente itération. Le support pour chaque candidat est alors compté et testé par rapport au seuil minsup. Le nombre total d'itérations nécessaires pour l'algorithme est $k_{max} + 1$, où k_{max} est la taille maximum des fréquents itemsets.

Alg. 1 Génération des itemsets fréquents dans l'algorithme Apriori

1: $k = 1$
2: $F_k = \{i | i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$. {Trouver tous les 1-itemsets fréquents}
3: **répéter**
4: $k = k + 1$
5: $C_k =$ apriori-gen(F_{k-1}) {Générer les itemsets candidats}
6: **pour** chaque transaction $t \in T$ **faire**
7: $C_t =$ sous-ensemble$(C - k, t)$ { Identifier tous les candidats qui appartiennent à t }
8: **fin pour**
9: **pour** chaque itemset candidat $c \in C_t$ **faire**
10: $\sigma(c) = \sigma(c) + 1$ { Incrémenter le nombre support }
11: **fin pour**
12: **jusqu'à** $F_k = \emptyset$
13: **return** $\bigcup F_k$

2.2.3 Génération des candidats et élagage

La fonction apriori-gen montré dans l'étape 5 de l'algorithme 1 génère les itemsets candidats en effectuant les deux opérations suivantes :

1. Génération des candidats. Cette opération génère des nouveaux k-itemsets candidats basés sur les fréquents $(k-1)$itemsets trouvés dans l'itération précédente.
2. Elagage des Candidats. Cette opération élimine certain des candidats

Pour illustrer l'opération d'élagage des candidats, on considère un k-itemset candidat, $X = \{i_1, i_2, \ldots, i_k\}$. L'algorithme doit déterminer si tous les sous-ensembles, $X - \{i_j\}(\forall j = 1, 2, \ldots, k)$, sont fréquent. Si un d'entre eux n'est pas fréquent, alors X est immédiatement élagués. Cette approche peut effectivement réduire le nombre des itemsets candidats considérés durant le comptage du support. La complexité de cette opération est O(k) pour chaque k–itemset candidat.

La procédure $F_{k-1} \times F_{k-1}$ de génération de candidats dans la fonction apriori-gen fusionne une paire de fréquent (k-1)itemsets seulement si leurs premiers $k-2$ items sont identiques. Soit $A = \{a_1, a_2, \ldots, a_{k-1}\}$ et $B = \{b_1, b_2, \ldots, b_{k-1}\}$ une paire de $(k-1)$itemsets fréquents. A et B sont fusionnés si ils satisfont les conditions suivantes : $a_i = b_i$ (pour $i = 1, 2, \ldots, k-2$) et $a_{k-1} \neq b_{k-1}$.

2.2.4 Comptage du support

Le comptage du support est le processus qui consiste à déterminer la fréquence d'occurrence pour chaque itemset candidat qui passe l'étape de l'élagage des candidats de la fonction apriori-gen. Le comptage de support est implémenté dans les étapes 6 à 11 de l'algorithme 1. Une approche pour faire cela est de comparer chaque transaction contre tous les itemsets candidats (voir figure) et de mettre à jour le nombre support des candidats contenus dans la transaction. Cette approche est coûteuse en temp de calcul, spécialement lorsque le nombre de transactions et d'itemsets candidats est élevé. Une approche alternative est d'énumerer les itemsets contenues dans chaque transaction et de les utiliser pour mettre à jour le nombre support de leur itemsets candidats respectifs. Pour illustrer, on considère la transaction t qui contient 5 items, $1, 2, 3, 5, 6$. I y a $(5\ 3)^T = 10$ itemsets de taille 3 contenu dans cette transaction. Certain de ces itemsets peuvent correspondre au candidat 3-itemset étudié, dans quel cas, leur nombre support sont incrémentés.

3 Génération des règles

Chaque fréquent kitemset, Y , peut produire jusqu'à $2k-2$ règles d'association, en ignorant les règles qui ont des antécédents ou conséquents vide ($\emptyset \rightarrow Y$ou $Y \rightarrow \emptyset$). Une règle d'association peut être extraite en partitionant l'itemset Y en deux sous-ensembles non vides, X et $Y - X$, tel que $X \rightarrow Y - X$ satisfait le seuil de confiance.

Exemple : Soit $X = \{1, 2, 3\}$ un fréquent itemset. Il y a six règles d'association candidates qui peuvent être générées à partir de X : $\{1, 2\} \rightarrow \{3\}, \{1, 3\} \rightarrow \{2\}, \{2, 3\} \rightarrow \{1\}, \{1\} \rightarrow \{2, 3\}, \{2\} \rightarrow \{1, 3\}, et \{3\} \rightarrow \{1, 2\}$. Comme leur support est identique au support de X, les règles doivent satisfaire le seuil support. Le calcul de la confiance d'une règle d'association ne nécessite pas de passe additionnelle des données de transactions. Soit la règle $\{1, 2\} \rightarrow \{3\}$, qui est générée à partir du fréquent itemset $X = \{1, 2, 3\}$. La confiance pour cette règle est $\sigma(\{1, 2, 3\})/\sigma(\{1, 2\})$. Comme $\{1, 2, 3\}$ est fréquent, la propriété d'antimonotonie du support assure que $\{1, 2\}$ doit être fréquent, également. Puisque le nombre support pour les itemsets a déjà été trouvé durant la génération des fréquents itemset, il n'y a pas besoin de relire encore une fois les données.

3.0.1 Elagage basée sur la confiance

Contrairement à la mesure de support, la confiance n'a pas de propriété de monotonie. Par exemple, la confiance pour $X \rightarrow Y$ peut être plus grande, plus petite, ou égale à la confiance d'une autre règle $\bar{X} \rightarrow \bar{Y}$ où $\bar{X} \subseteq X$ et $\bar{Y} \subseteq Y$ Cependant, si on compare les règles générées à partir du frqeuent itemset Y , le théorème suivant s'applique pour la mesure de confiance.

Théorème 3.1 *Si une règle $X \rightarrow Y - X$ ne satisfait pas le seuil de confiance, alors toutes règles $X \rightarrow Y - X'$, où X' est un sous-ensemble de X, ne doit pas satisfaire le seuil de confiance.*

3.0.2 Génération des règles dans l'algoritme apriori

L'algorithme Apriori utilise une approche de par niveau pour générer les règles d'association, où chaque niveau correspond au nombre d'items qui appartiennent à la conséquence de la règle. Initialement, toutes les règles à confiance élevés qui n'ont qu'un seul item dans le conséquent de la règle sont extraites. Ces règles sont ensuite utilisées pour générer de nouvelles règles candidates.
Par exemple, si $\{acd\} \rightarrow \{b\}$ and $\{abd\} \rightarrow \{c\}$ sont des règles à confiance élevée, alors la règle candidate $\{ad\} \rightarrow \{bc\}$ est générée en fusionnant les conséquents des deux règles. La figure 3 montre une structure de treilli pour les règles d'association générées à partir de l'itemset fréquent $\{a, b, c, d\}$. Si il y a noeud dans le treilli qui a une confiance faible, alors selon le théorème 6.2, le sous-graphe entier engendré par le noeud peut être élagué immédiatement. On suppose que la confiance pour $\{bcd\} \rightarrow \{a\}$ est faible. Toutes les règles contenant l'item a dans leur conséquent, incluant $\{cd\} \rightarrow \{ab\}$, $\{bd\} \rightarrow \{ac\}$,

$\{bc\} \rightarrow \{ad\}$, et $\{d\} \rightarrow \{abc\}$ peuvent être écartées. Le pseudo-code pour l'étape de la génération de règles figure dans l'algorithme 2 et 3 . On peut noter la similarité entre la procédure apgenrules décrite dans l'algorithm 3 et la procédure de génération d'itemset fréquent décrite dans l'algorithme 1. La seule difference est que, dans la génération de règles, on ne doit pas faire de passes additionnelles sur les données pour le calcul de la confiance des règles candidates. A la place, on détermine la confiance de chaque règle en utilisant les nombres support calculés durant la génération des fréquents itemset.

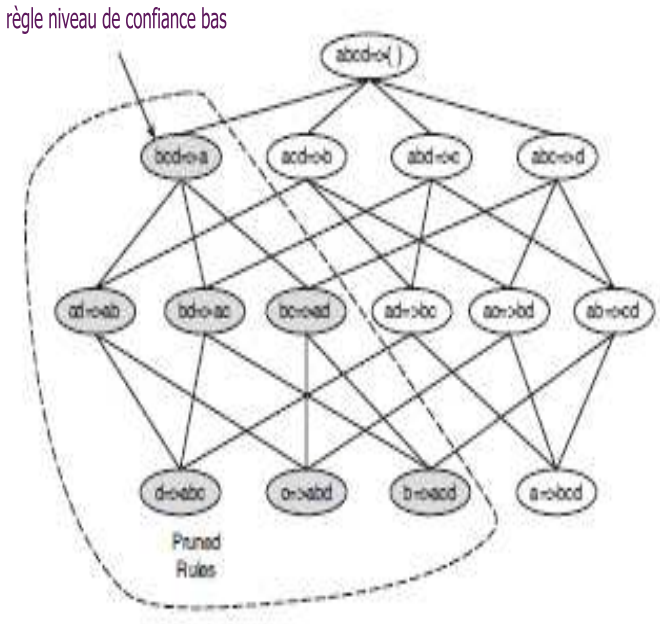

FIGURE 3 – Elagage des règles d'association en utilisant la mesure de confiance.

Alg. 2 Génération des règles algorithme Apriori

1: **pour** chaque fréquent k-itemset $f_k, k \geq 2$ **faire**
2: $H_1 = \{i, i \in f_k\}$ { 1-item conséquent de la règle }
3: appel app-genrules(f_k, H_1)
4: **fin pour**
5: **return** $\bigcup F_k$

Alg. 3 Génération des itemsets fréquents dans l'algorithme Apriori

1: $k = f_k$ {Taille fréquent itemset}
2: $m = H_m$. {Taille du conséquent de la règle}
3: **si** $k \geq m + 1$ **alors**
4: $H_{m+1} = apriori - gen(H_m)$
5: **pour** $h_{m+1} \in H_{m+1}$ **faire**
6: $conf = \sigma(f_k)/\sigma(f_k - h_{m+1})$
7: **si** $conf \geq minconf$ **alors**
8: **sortir** la règle $f_k - h_{m+1} \leftarrow h_{m+1}$
9: **sinon**
10: **retirer** la règle $h_{m+1} de H_{m+1}$
11: **fin si**
12: **fin pour**
13: appel app-genrules(f_k, H_{m+1})
14: **fin si**