Chapitre 4

Taches du Data Mining
Règles d'association

Règles d’association

Introduction

La technique d'extraction des règles d'association fait partie des modèles du Data Mining associative, cette technique permet la découverte de règles intelligibles et exploitables dans un ensemble de données, tout en exprimant des associations entre les données. L’objectif du processus d'extraction est de découvrir des règles (relations) significatives entre attributs extraits des bases de données, dont le support et la confiance sont au moins égaux à des seuils minimaux définis par l'utilisateur.

Règles d'association

Recherche de règles d'association

- **Règles de la forme**

ANTECEDENT → CONSEQUENT [Support, Confiance]

(support et confiance sont des mesures d'intérêt définies par l'utilisateur)

- ✓ Achat (x, « Beurre ») ET Achat (x, « Pain ») → Achat(x, « Lait ») [70%, 80%]
- ✓ Achat (x, « Farine ») ET Achat (x, « Gâteaux ») → Achat (x, «Couches ») [30%, 80%]
- ✓ Achat (x, « Caviar ») → Achat(x, « Champagne ») [10%, 90%]

Règles d'association

Interprétation

- R : X →Y (A%, B%)
- ✓ ***Support*** *: portée de la règle*

Proportion de paniers contenant tous les attributs A% des clients ont acheté les 2 articles X et Y

- ✓ ***Confiance*** *:*

Proportion de paniers contenant le conséquent parmi ceux qui contiennent l'antécédent

B% des clients qui ont acheté X ont aussi acheté Y

- Beurre, Pain → Lait [70%, 80%]
- Farine, Gâteaux → Couches [30%, 80%]
- Caviar → Champagne [10%, 90%]

Règles d’association

Utilisation des règles d’association

Farine, … →Couches

- **Couches** comme conséquent

  déterminer ce qu’il faut faire pour augmenter les ventes

- **Farine** comme antécédent

  quel produit serait affecté si on n’arrête de vendre de la Farine

- **Farine** comme antécédent et **Couche** comme conséquent

quels produits devraient être vendus avec la Farine *pour promouvoir la vente de couches*

Règles d'association

Définitions

- Soit un ensemble I = {I1, I2, ..., Im} d'items, une transaction T est définie comme les sous-ensembles d'items dans I ($\subseteq$ I).
  - I = {Farine, Café, Couche, Gâteaux,Moutarde,Saucisse...}
  - T1 = {Café, Moutarde, Saucisse}
- Une transaction n'a pas de duplicats
- Soit une base de données D un ensemble de n transactions et chaque transaction est nommée par un identifiant (TID).
  - D = {{T1, {Café,Moutarde,Saucisse}},
    {T2, {Farine, Café, Gâteaux}}, ...}

Règles d'association

Une base de données

- Une représentation de la base de données D

Client	Pizza	Lait	Sucre	Pommes	Café
1	1	0	0	0	0
2	0	1	1	0	0
3	1	0	0	1	1
4	0	1	0	0	1
5	1	0	1	1	1

- En fait

$$M = \begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \end{matrix}$$

Règles d'association

Définitions

- Une transaction T **supporte** un ensemble X ⊆ I si elle contient tous les items de X (X ⊆ T).
  - ✓ T1 supporte {Café, Moutarde,Saucisse}
- Support de X (**Supp(X)**) : fraction de toutes les transactions dans D qui supportent X.
- Si supp(X) ≥ minsup l'ensemble X est dit **fréquent**.
- Un ensemble d'items (*itemset*) X de cardinalité k = |X| est appelé un *k-itemset*.

  3-itemset : {Café, Moutarde, Saucisse}

Règles d'association

Propriétés des ensembles fréquents

- **Propriété 1** : *support pour les sous-ensembles*
  - ✓ Si A ⊆ B pour les itemsets A, B alors supp(A) >= supp(B) car toutes les transactions dans D qui supportent B supportent aussi nécessairement A.

A={Café, Moutarde}, B ={Café, Moutarde, Saucisse}

- **Propriété 2** : *les sous-ensembles d'ensembles fréquents sont fréquents*
- **Propriété 3** : *les sur-ensembles d'ensembles non fréquents sont non fréquents* (anti-monotonie)

Règles d'association

Définition des Règles d'association

- Une règle d'association est une implication de la forme
  R : X → Y où X et Y sont des itemsets disjoints : X, Y ⊆ I et X ∩ Y = ∅.
  Farine, Gâteaux → Couches
- Confiance (*confidence*) dans une règle R
- Si une transaction supporte X, elle supporte aussi Y avec une certaine probabilité appelée **confiance** de la règle (conf(R)).

conf(R) = p(Y ⊆T | X ⊆ T)
= p(Y⊆T ∧ X⊆T) / p(X ⊆ T)
= support (X U Y) / support(X)

$$conf(R) = \frac{Supp(Farine,Gâteaux,Couches)}{Supp(Farine, Gâteaux)} \geq \text{confiance ?}$$

Règles d'association

Propriétés des règles d'association

- **Propriété 4** : *pas de composition des règles*
  - Si X → Z et Y → Z sont vrais dans D, X U Y → Z n'est pas nécessairement vrai.
  - ✓ Considérons le cas où X ∩ Y = Ø et les transactions dans D supportent Z si et seulement si elles supportent X ou Y, alors l 'ensemble X U Y a un support de 0 et donc X U Y → Z a une confiance de O%.
- **Propriété 5** : *décomposition des règles*
  - Si X U Y → Z convient, X → Z et Y → Z peut ne pas être vrai.
- **Propriété 6** : *pas de transitivité*
  - Si X → Y et Y → Z , nous ne pouvons pas en déduire que X → Z.
- **Propriété 7** : *déduire si une règle convient*
- Si A → (L-A) ne vérifie pas la confiance alors nous n'avons pas B → (L-B) pour les itemsets L, A, B et B ⊆ A.

Règles d'association

En résumé

- Itemsets : A, B ou B, E, F
- Support pour un itemset
  - Supp (A,D)=1
  - Supp (A,C) = 2
- Itemsets fréquents (minSupp=50%)
  {A,C} est un itemset fréquent
- Pour minSupp = 50% et minConf = 50%, nous avons les règles suivantes :
  - A → C [50%, 50%]
  - C → A [50%, 100%]

Trans. ID	Items
1	A, D
2	A, C
3	A, B, C
4	A, B, E, F

Règles d’association

algorithmique de base

- La plupart des approches utilise le même schéma algorithmique
- Pour construire les règles d ’association, le support de tous les itemsets fréquents dans la base doit être calculé
- L ’algorithme procède en deux phases :
  1) Génération de tous les ensembles fréquents
  2) Génération des règles d ’association
- **Comptage des itemsets**
- Une première approche

  I = {A, B,C}
- Génération de tous les cas possibles :

  {Ø},{A}, {B}, {C},
  {A,B}, {A,C}, {B,C}
  {A,B,C}
- Comptage du support

Règles d’association

Génération des ensembles fréquents

- Le nombre d ’ensemble fréquent potentiel est égal à la taille du produit cartésien de tous les items …. qui croit exponentiellement en fonction du nombre d ’items considérés.
- Approche naïve : recherche exhaustive et test de tous les ensemble du produit cartésien pour savoir s ’ils sont fréquents
- 1000 items => 2^{1000} ensembles à considérer

Règles d’association

Construction des règles

- Pour chaque ensemble fréquent X, chaque sousensemble est choisi comme antécédent de la règle, le reste devenant la partie conséquent.
- Comme X est fréquent, tous les sous-ensembles sont fréquents (Propriété 3) donc leur support est connu. La confiance d’une règle est calculée et une règle est conservée ou pas selon la confiance minimale.
- Amélioration : (Propriété 7) quand une règle échoue, aucun sous ensembles de l ’antécédent n ’est à considérer.

Règles d'association

L'algorithme APRIORI

- But : minimiser les candidats
- Principe : générer seulement les candidats pour lesquels tous les sous-ensembles ont été déterminés fréquents
- Génération des candidats réalisée avant et de manière séparée de l'étape de comptage

Input : C_k: itemsets candidats de taille k
Output : L_k : itemsets fréquents de taille k
L_1 = {items fréquents}; for (k = 1; L_k !=∅; k++) do
  C_{k+1} = candidats générés à partir de L_k;
  Pour chaque transaction t de la base de données, incrémenter le compteur de tous les candidats dans C_{k+1} qui sont contenus dans t
  L_{k+1} = candidats dans C_{k+1} avec minSupp
  return $\cup_k L_k$;

Règles d'association

Détails d'APRIORI

- Comment générer les candidats ?
  - Etape 1: auto-jointure sur L_k
  - Etape 2: élagage
- Comment compter le support des candidats ?

➢ Génération des candidats

- Les items de L_{k-1} sont ordonnés par ordre lexicographique
- Etape 1: auto-jointure sur L_{k-1}

  INSERT INTO C_k
  SELECT $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$
  FROM L_{k-1} p, L_{k-1} q
  WHERE $p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$
- Etape 2: élagage

  For each itemset c in C_k do
  For each (k-1)-subsets s of c do if (s is not in L_{k-1}) then delete c from C_k

Règles d'association

Génération des candidats : exemple

- L_3={*abc, abd, acd, ace, bcd*}
- Auto-jointure : $L_3 * L_3$
  - ✓ *abcd* à partir de *abc* et *abd*
  - ✓ *acde* à partir de *acd* et *ace*
- Élagage :
  - ✓ *acde* est supprimé car *ade* n'est pas dans L_3
- C4={abcd}

Règles d’association

Comptage du support des candidats

- Parcourir la base. Pour chaque tuple extrait t, compter tous les candidats inclus dedans
  - ✓ Rechercher toutes les feuilles qui peuvent contenir les candidats
  - ✓ Hachage sur chaque item du tuple et descente dans l ’arbre des candidats
- Dans les feuilles de l’arbre vérifier ceux effectivement supportés par t
- Incrémenter leur support

Règles d’association

Illustration

CID	Items
1	A B
2	A B C D E F
3	B D G
4	B E G
5	D F G
6	DEG
7	B E
8	B D E F

Support minimal = 1

Règles d’association

Illustration

C1	Support
A	2
B	6
C	1
D	5
E	5
F	3
G	4

L1 = {{A},{B},{C},{D},{E},{F},{G}} 1-itemsets fréquents

Règles d'association

Illustration

C2	Support	C2	Support
AB	2	CD	1
AC	1	CE	1
AD	1	CF	1
AE	1	CG	0
AF	1	DE	3
AG	0	DF	3
BC	1	DG	3
BD	3	EF	2
BE	4	EG	2
BF	2	FG	1
BG	2		

2-itemsets fréquents

{{A,B},{A,C},{A,D},{A,E},{A,F},{B,C},{B,D},{B,E},{B,F},{B,G},{C,D},{C,E},{C,F},{D,E},{D,F},{D,G},{E,F},{E,G},{F,G}}

Règles d'association

Illustration

C3	Support	C3	Support
ABC	1	BDE	2
ABD	1	BDF	2
ABE	1	BDG	1
ABF	1	BEF	2
ACD	1	BEG	1
ACE	1	BFG	0
…	…	…	…
BCF	1	EFG	0

L3 = {{A,B,C},{A,B,D},{A,B,E},{A,B,F},{A,C,D}, … {D,F,G}}
{B,C,G} élagué par Apriori-Gen car {C, G} n 'appartient pas à L2

Règles d'association

Illustration

C4	Support	C4	Support
ABCD	1	ACEF	1
ABCE	1	ADEF	1
ABCF	1	BCDE	1
ABDE	1	BCDF	1
ABDF	1	BCEF	1
ABEF	1	BDEF	2
ACDE	1	BDEG	0
ACDF	1	CDEF	0

L4 = {{A,B,C,D},{A,B,C,E},{A,B,C,F}, … {C,D,E,F}}
{B,D,F,G}, {B,E,F,G} élagués car {B,F,G} n 'appartient pas à L3
{D,E,F,G} élagué car {E,F,G} n 'appartient pas à L3

Règles d’association

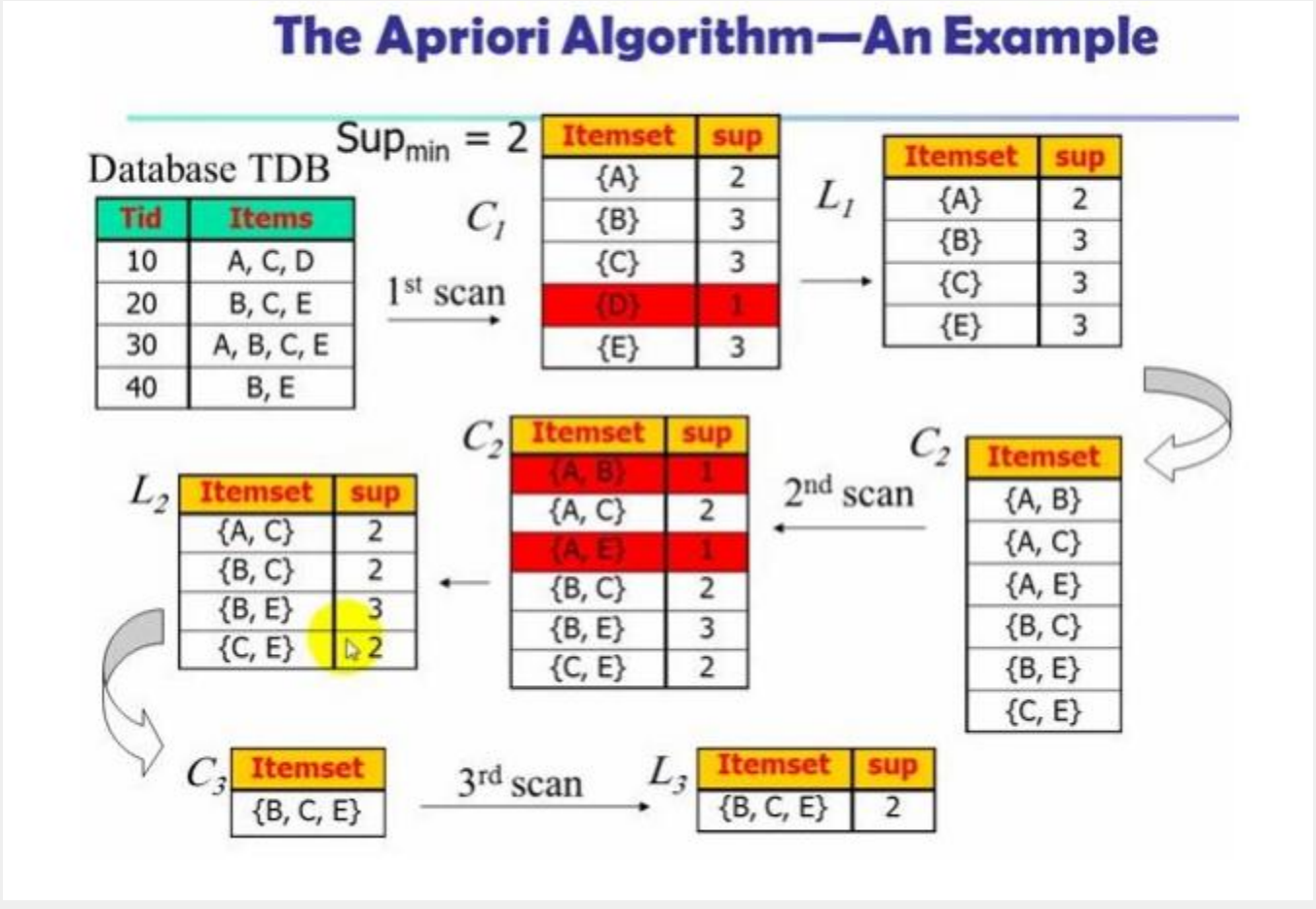
Illustration

C6	Support
ABCDEF	1

6-itemsets fréquents L6 = {{A,B,C,D,E,F}}
C7 = {Ø} => l ’algorithme se termine.

Règles d'association

Illustration



Règles d'association

Illustration

Extraction des règles d'association (I)

Démarche

Paramètres : Fixer un degré d'exigence sur les règles à extraire
>> Support min. (ex. 2 transactions)
>> Confiance min. (ex. 75%)

→ L'idée est surtout de contrôler (limiter) le nombre de règles produites

Démarche : Construction en deux temps
>> recherche des itemsets fréquents (support >= support min.)
>> à partir des itemsets fréquents, produire les règles (conf. >= conf. min.)

Quelques définitions :
>> item = produit
>> itemset = ensemble de produits (ex. {p1,p3})
>> sup(itemset) = nombre de transactions d'apparition simultanée des produits (ex. sup{p1,p3} = 4)
>> card(itemset) = nombre de produits dans l'ensemble (ex. card{p1,p3} = 2)