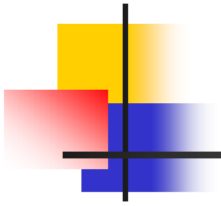


INF5081

Règles d'associations
L'algorithme Apriori

Mohamed Bouguessa

Objectifs

- Comprendre l'importance de l'extraction des règles d'associations dans la pratique
- Présentation d'algorithmes capable d'extraire des règles d'associations
- Présenter une approche efficace qui permet de distinguer entre les règles intéressantes des règles qui ne contiennent pas de l'information pertinente.

Plan

1- Motivations

2- Concepts de base

3- Règles d'associations

4- Extraction des associations

5- L'algorithme Apriori

Motivations

- Approche automatique pour découvrir des relations / corrélations intéressantes entre des objets
- Origine marketing : analyser les ventes des supermarchés
- Données en entrée : données de transaction

Caddie	Liste des items achetés
1	{pain, beurre, lait}
2	{pain,viande}
…	
n	{jus de fruit, poisson, fraises, pain}

- Données en sortie : règles d'associations

Motivations

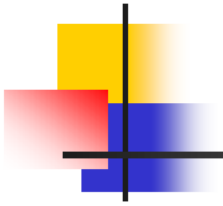
Règles de la forme: A ⇒ B [support, confiance]

A et B peuvent être composés de conjonctions

"lorsqu'un client achète du pain et du beurre, il achète 9 fois sur 10 du lait en même temps"

$$\{pain, beurre\} \Rightarrow \{lait\}$$

Ensemble d'items **A**	⇒	*Ensemble d'items* **B**	où **A ∩ B = Ø**
antécédent		conséquent	



Plan

1- Motivations

2- Concepts de base

3- Règles d'associations

4- Extraction des associations

5- L'algorithme Apriori

Concepts de base

- Soit D une base de donnée de N transactions.
- Chaque transaction est décrite par un identifiant T_{id} et une liste d'items

  $D = \{T_1, T_2, \ldots, T_N\}$; $T_{id} = \{i_1, i_2, \ldots, i_m\} \subset I$

  $I = \{i_1, i_2, \ldots, i_n\}$ l'ensemble de tout les items possibles
- Itemset : une collection d'un ou plusieurs items / un ensemble d'items
- *K*-itemset : un itemset qui contient *K* items

Exemple

5 transactions et 6 items

I ={Pain, Lait, Beurre, Fromage, Œufs, Coke}

T_{ID}	Items
1	Pain, Lait
2	Pain, Beurre, Fromage, Œufs
3	Lait, Beurre, Fromage, Coke
4	Pain, Lait, Beurre, Fromage
5	Pain, Lait, Beurre, Coke

1-itemset {{Pain}, {Lait}, {Beurre}, … {Coke}}

2-itemset {{Pain, Lait}, {Pain, Beurre}, …, {Fromage, Œufs}}

3-itemset {{Pain, Lait, Beurre}, …, {Fromage, Œufs, Coke}}

Concepts de base

Support d'un itemset

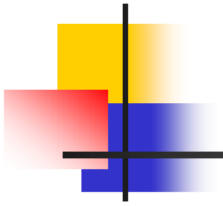
Le support d'un itemset set est le % de transactions qui contiennent l'itemset en question

$$\text{Support}(X) = \frac{\#X}{N}$$

TID	Items
1	Pain, Lait
2	Pain, Beurre, Fromage, Œufs
3	Lait, Beurre, Fromage, Coke
4	Pain, Lait, Beurre, Fromage
5	Pain, Lait, Beurre, Coke

Item	*Support*
Pain	80%
Lait	80%
Beurre	80%
Fromage	60%
Œufs	20%
Coke	40%
Pain, Lait	60%
⋮	⋮
Lait, Œufs	0%
⋮	⋮
Pain, Lait, Beurre	40%
⋮	⋮
Pain, Lait, Beurre, Coke	20%
⋮	⋮
Pain, Lait, Beurre, Coke, Fromage	0%
⋮	⋮

Un itemset est fréquent si support(itemset) >= *minsup*



Plan

1- Motivations

2- Concepts de base

3- Règles d'associations

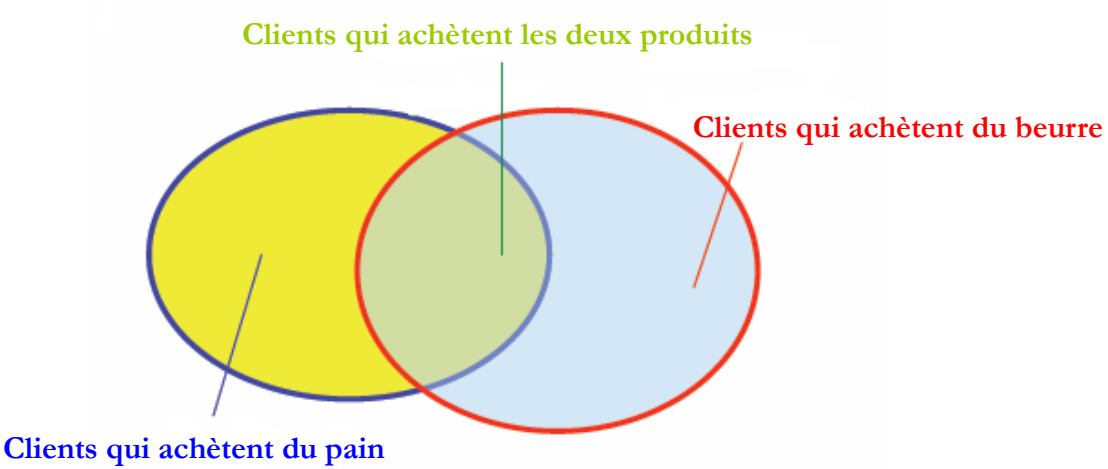
4- Extraction des associations

5- L'algorithme Apriori

Règles d'associations

Les règles d'associations expriment une corrélation entre la présence d'un item avec la présence d'un ensemble d'items.

Ex: 60% des personnes qui achètent du pain achètent du beurre



Règles d'associations

Une règle d'association est de la forme A→B

$$A \subseteq I, B \subseteq I \; et \; A \cap B = \phi$$

Pour chaque règle d'association A→B on doit estimer:

1 – **Support** : probabilité qu'une transaction contienne A et B.

2 – **Confiance** : probabilité conditionnelle P(B | A).

Règles d'associations

Support et confiance d'une règle d'association

➢ Le **support** d'une règle d'association A→B est le % des transactions qui contiennent $A \cup B$

$$\text{support} \quad (A \rightarrow B) = \frac{\#(A \cup B)}{N}$$

➢ La **confiance** d'une règle d'association A→B est le ratio du nombre de transactions qui contiennent $A \cup B$ sur le nombre de transactions qui contiennent A

$$confiance(A \rightarrow B) = p(B \mid A) = \frac{P(A et B)}{P(A)}$$

$$= \frac{\sup port(A \rightarrow B)}{\sup port(A)}$$

$$= \frac{\#(A \cup B)}{\#(A)}$$

Règles d'associations

- **Exemple : Support et confiance d'une règle d'association**

TID	Items
1	Pain, Lait
2	Pain, Beurre, Fromage, Œufs
3	Lait, Beurre, Fromage, Coke
4	Pain, Lait, Beurre, Fromage
5	Pain, Lait, Beurre, Coke

{Lait, Beurre} → {Fromage} (s = 40%, c = 67%)
{Lait, Fromage} → {Beurre} (s = 40%, c =100%)
{Beurre, Fromage} → {Lait} (s = 40%, c = 67%)
{Fromage} → {Lait, Beurre} (s = 40%, c = 67%)
{Beurre} → {Lait, Fromage} (s = 40%, c = 50%)
{Lait} → {Beurre, Fromage} (s = 40%, c = 67%)

- Remarque
  - Toutes ces règles sont issue de l'itemset : {Lait, Beurre, Fromage}
  - Les règles originaires du même itemset ont un support identique, mais différentes valeurs de confiance

Pourquoi?

Règles d'associations

Support et confiance : pourquoi deux mesures?

Support

- Le support mesure le nombre d'occurrences d'une règle dans l'ensemble de données.
- Généralement utilisé pour éliminer les règles moins intéressantes.

Expl. une règle d'association, qui représente les items achetés ensemble, qui a une valeur de support faible n'est pas intéressante du point de vue commercial, car ce n'est pas rentable de promouvoir les articles rarement achetés ensemble

{Pain, Beurre}→{Œufs} - support = 20%

Confiance

- Confiance : mesure la fiabilité et la pertinence de l'inférence faite par une règle.
- Spécifiquement : Étant donnée une règle A→B, plus la valeur de confiance est grande, plus il est probable que B soit présent dans un grand nombre de transactions qui contiennent A.

Règles d'associations

Définition formelle du problème de la recherche des règles d'association

Le problème de la recherche des règles d'associations peut êtres formulé comme suit:
Étant donné un ensemble de transactions D, identifier toutes les règles avec :

support >= *minsup* et confiance >=*minconf*,

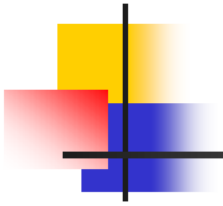
minsup et *minconf* correspondent respectivement aux seuils de support et de confiance.

Choix de la valeur de *minsup* et *minconf*

Élevée ⟹ Peu de règles mais toutes « pratiquement » pertinentes.

Réduite ⟹ Plusieurs règles, plusieurs d'entre elles sont « incertaines ».

Valeurs utilisées? → tout dépend de l'application



Plan

1- Motivations

2- Concepts de base

3- Règles d'associations

4- Extraction des associations

5- L'algorithme Apriori

Extractions des associations

Approche générale (Brute-force approach)

- Identifier toutes les règles d'associations.
- Calculer le support et la confiance de chaque règle.
- Sélectionner que les règles avec support et confiance >= *minsup* et *minconf*

⇨ Complètement non praticable sur de grandes bases de données

pour un ensemble de données avec *n* items, le nombre total des règles possible:

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

Si $n = 6$ alors $R = 602$, Plus de 80% des règles sont éliminées si *minsup* = 0.2 et *minconf=0.5*

On a besoin d'une stratégie efficace pour générer seulement que des règles intéressantes

Extractions des associations

Exemple

Les règles suivantes ont un support identique, car elles proviennent du même itemset

{Lait, Beurre, Fromage}:
{Lait, Beurre} → {Fromage} (s = 40%)
{Lait, Fromage} → {Beurre} (s = 40%)
{Beurre, Fromage} → {Lait} (s = 40%)
{Fromage} → {Lait, Beurre} (s = 40%)
{Beurre} → {Lait, Fromage} (s = 40%)
{Lait} → {Beurre, Fromage} (s = 40%)

- Intuition :

Si l'itemset {Lait, Beurre, Fromage} n'est pas fréquent
c.-à-d. support({Lait, Beurre, Fromage}) < minsup
alors toute les règles candidates peuvent êtres supprimées.

⇨ Décomposer le problème en deux étapes

Extractions des associations

- **Approche naïve**

  Deux étapes importantes pour identifier les règles d'associations

  Étape 1 : identifier tout les itemsets fréquents

  Étape 2 : générer les règles d'associations les plus pertinentes à partir des itemsets identifiés dans l'étape1

Étape 1

- D: une base de transactions
- I: ensemble de tous les items avec |I|=n
- Algorithme 1: Extraction des ensembles fréquents

```
Fréquents = ∅
Pour chaque J ⊆ I Faire
    count(J)=0
    Pour chaque transaction t∈D Faire
        Si J ⊆ t.items Alors
            count(J)=count(J)+1
    Si count(j) ≥ minsup Alors
        Fréquents=Fréquents += J
Fin
```

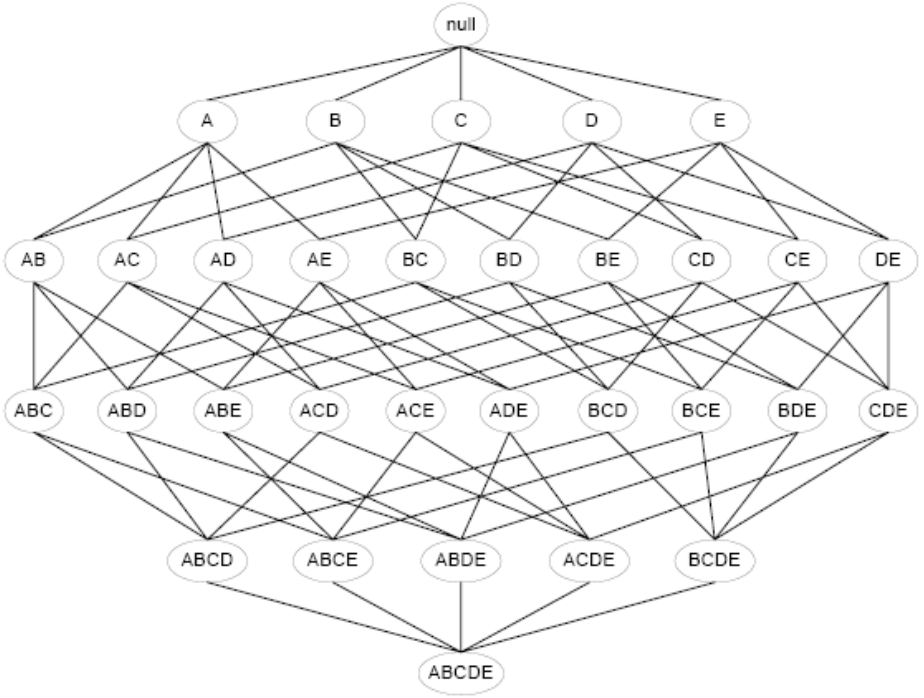
Étape 2

- D: une base de transactions
- I: ensemble de tous les items avec |I|=n
- Algorithme 2: Extraction des règles

```
Règles = ∅
Pour chaque J dans Fréquents
    Pour chaque règle r extraite de J={A1,…,Am}
        Si confiance(r) ≥ minconf Alors
            Règles= Règles+= r
Fin
```

Extractions des associations

⇨ **L'approche est toujours couteuse**



⇦ Structure de treillis qui permet d'énumérer la liste de tous les itemsets possible

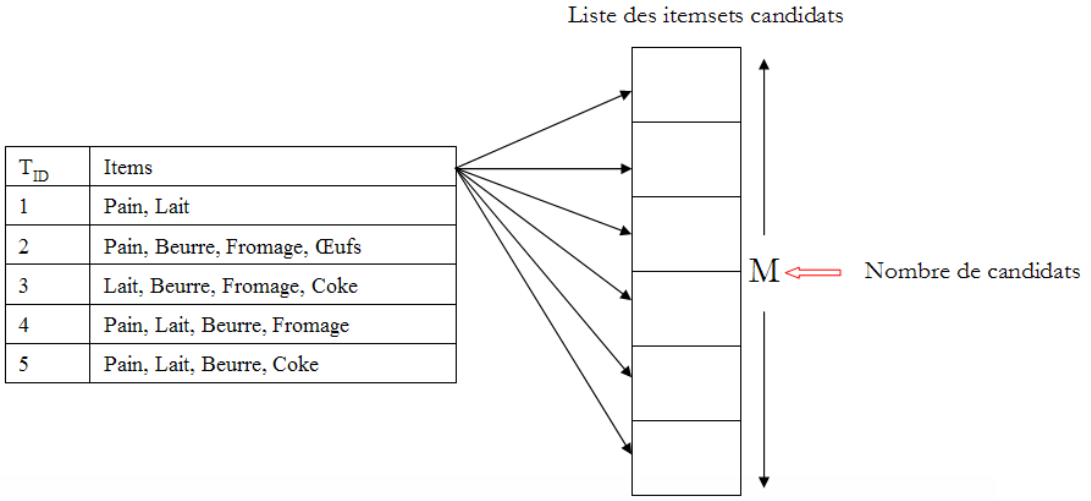
- |I| = n alors il y a 2^n-1 ensembles J (c'est le nombre de sous-ensembles de I)

  → 2^n-1 parcours de D (la base de données des transactions) !

Extractions des associations

- Calcule du support de chaque itemset candidat
  - Pour chaque sous-ensemble d'itemset possible *J*, il faut parcourir la base de données des transactions D pour compter le nombre de ses occurrences.

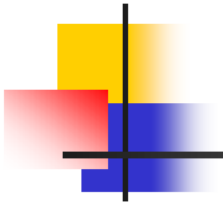
➢ Chaque itemset candidat on doit le comparer avec toutes les transactions



T_{ID}	Items
1	Pain, Lait
2	Pain, Beurre, Fromage, Œufs
3	Lait, Beurre, Fromage, Coke
4	Pain, Lait, Beurre, Fromage
5	Pain, Lait, Beurre, Coke

⇨ Temps de calcule énorme puisque M = 2^n-1

- **Comment réduire le nombre des itemsets candidats (M) ?**



Plan

1- Motivations

2- Concepts de base

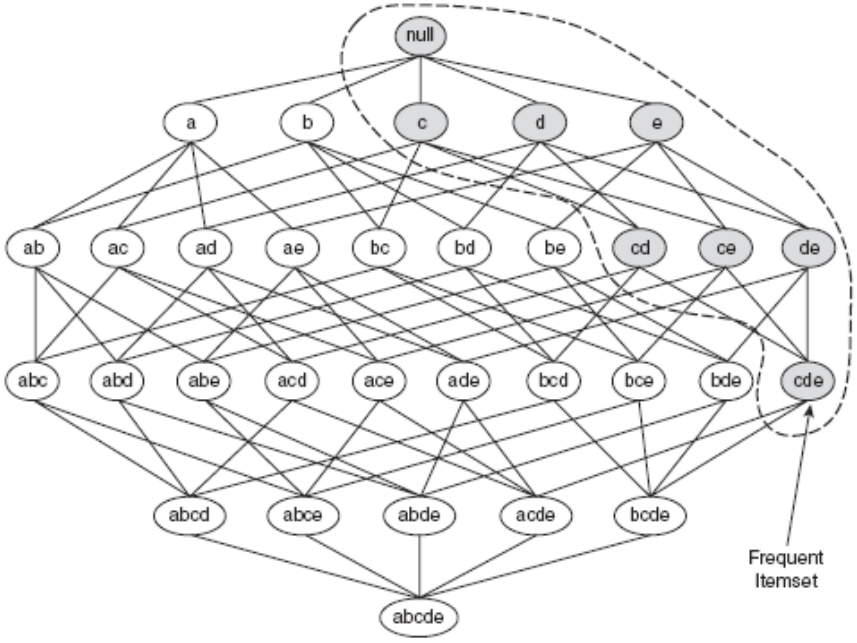
3- Règles d'associations

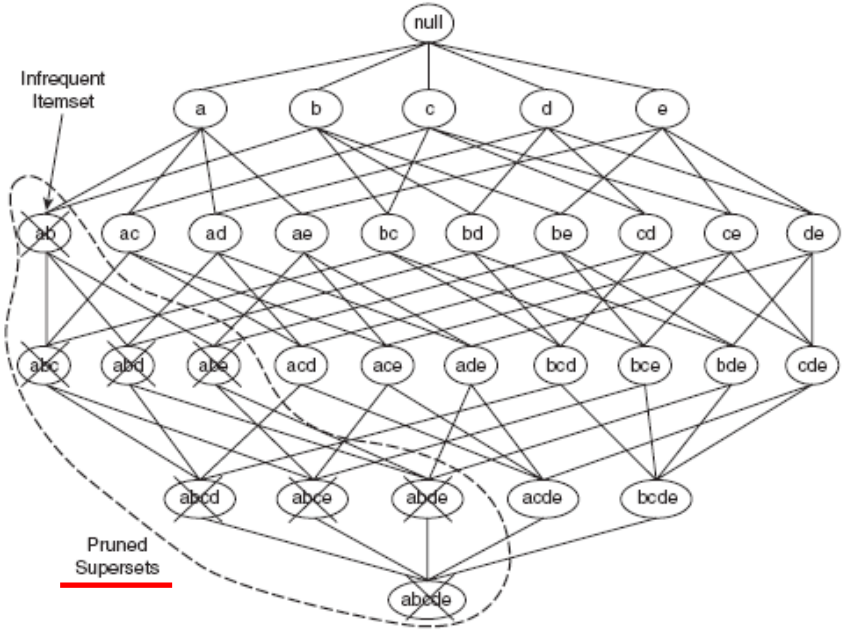
4- Extraction des associations

5- L'algorithme Apriori

Extractions des associations : Apriori

- **Principe**
  - Si un itemset est fréquent alors tout ses sous-ensemble sont aussi fréquents
  - Si un itemset est non fréquent, alors tout ses sur-ensemble ne sont pas fréquents
- **Illustration**





Principe de L'algorithme Apriori

L'idée

Le support d'un itemset ne peut jamais dépasser le support de ces sous-ensembles.

- Cette propriété est connue aussi sous le nom de la propriété anti-monotone.

❑ **Définition 1 (la propriété monotone)** soit I un ensemble d'item, et J = 2^I l'ensemble des itemsets possibles. Une mesure *f* est monotone si

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

❑ **Définition 2 (la propriété anti-monotone)**

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

- Une mesure qui possède une propriété anti-monotone peut être utilisée peut réduire la taille exponentielle de l'espace de recherche des itemsets candidats.

L'algorithme Apriori – Exemple

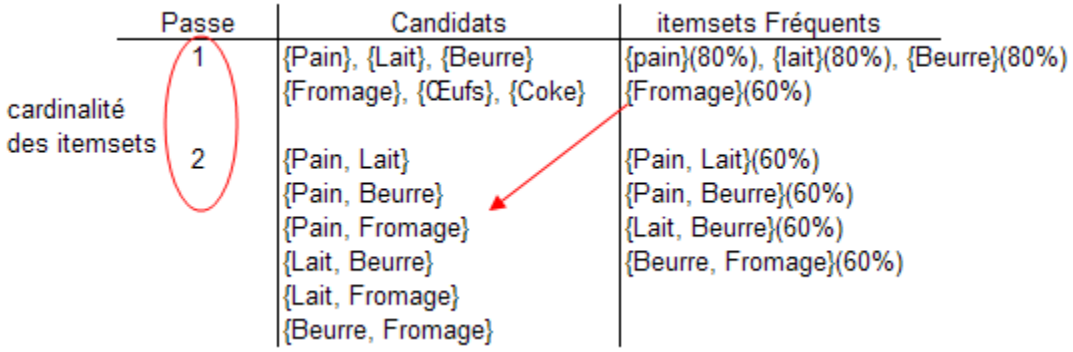
Générer les itemsets candidats dont la cardinalité = k

1- Scanner la BDD pour identifier les itemsets fréquents

2- Utiliser seulement les itemsets identifiés dans l'étape 1 pour générer les itemsets candidats de cardinalité k = k+1

minsup = 50%

TID	Items
1	Pain, Lait
2	Pain, Beurre, Fromage, Œufs
3	Lait, Beurre, Fromage, Coke
4	Pain, Lait, Beurre, Fromage
5	Pain, Lait, Beurre, Coke



L'algorithme Apriori

L'algorithme

$k = 1$
L_k : itemsets frequents de taille k

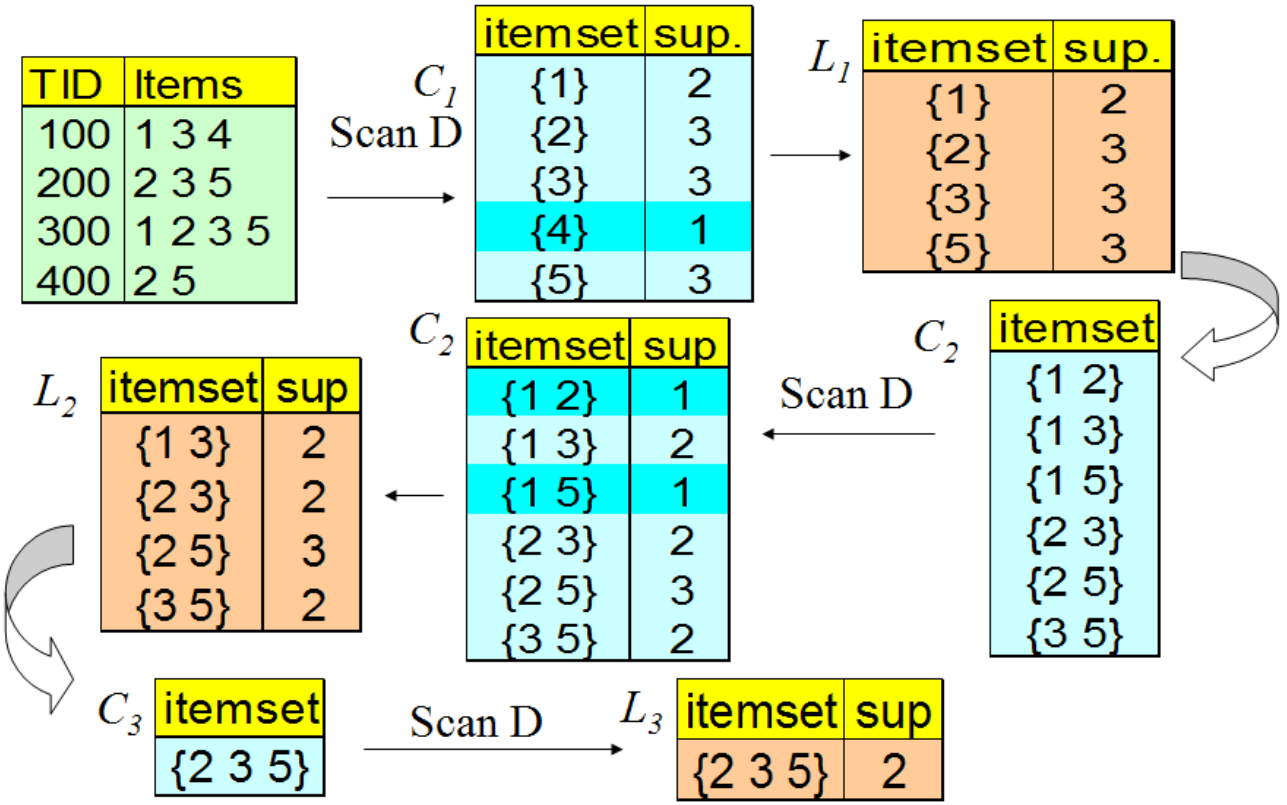
pour ($k = 2$; $L_k != \phi$; k++) //répéter jusqu'à aucun itemset ne sera identifier

- C_k = apriori-gen(L_k-1) ; // C_k : l'ensemble des itemsets candidat générés à partir de L_k-1
- pour chaque transaction T dans D
  - C_t = sous-ensemble(C_k, T) ; //identifier les candidats qui appartiennent à T
  - pour chaque itemset candidat c $\in C_t$
    - cpt(c) = cpt(c) + 1 ;
- $L_k = \{c \mid c \in C_k \; et \; cpt(c) \geq N * \min \sup\}$

return $\bigcup_k L_k$

L'algorithme Apriori

- Exemple (*minsup = 2*)



$\{\{1, 2, 3\}, \{1, 2, 5\}, \{1, 3, 5\}\} \notin C_3$

L'algorithme Apriori

- **Génération des itemset candidats: apriori-gen**

Un itemset est considéré comme candidat seulement si tous ces sous-ensembles sont fréquents.

La génération des itemsets candidats se fait en deux étapes

- **Etape de jointure :** C_k est généré en joignant L_{k-1}
- **Etape d'élimination:** Chaque (k-1)-itemset qui n'est pas fréquent ne peut être un sous-ensemble d'un k-itemset fréquent

⇨ **La procédure de générations des itemset candidats de l'algorithme Apriori fusionne une pair de (k-1)- itemset fréquent seulement si leur première k-2 itemset sont identique.**

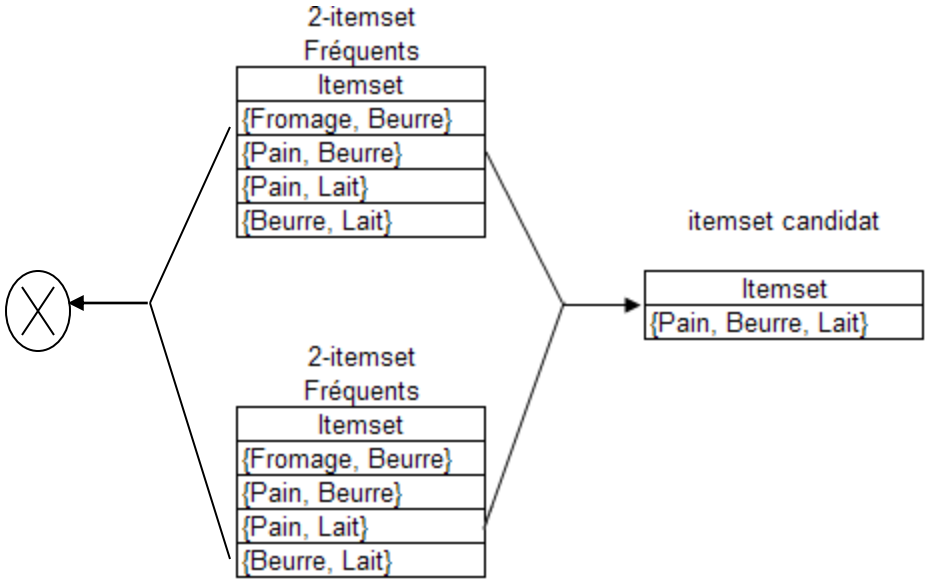
- **Formellement**

Soit $A=\{a_1, a_2, \ldots, a_{k-2}, a_{k-1}\}$ et $B=\{b_1, b_2, \ldots, b_{k-2}, b_{k-1}\}$ deux (k-1)-itemsets fréquents. A et B sont fusionné si:

$$\mathbf{a_i = b_i\ (i = 1, 2, \ldots, k-2)\ et\ a_{k-1} \neq b_{k-1}}$$

L'algorithme Apriori

- Génération des itemset candidats



L'algorithme Apriori

Exemple

- Given: 20 clothing transactions; *s*=20%, c=50%
- Generate association rules using the Apriori algorithm

Transaction	Items	Transaction	Items
t_1	Blouse	t_{11}	TShirt
t_2	Shoes, Skirt, TShirt	t_{12}	Blouse, Jeans, Shoes, Skirt, TShirt
t_3	Jeans, TShirt	t_{13}	Jeans, Shoes, Shorts, TShirt
t_4	Jeans, Shoes, TShirt	t_{14}	Shoes, Skirt, TShirt
t_5	Jeans, Shorts	t_{15}	Jeans, TShirt
t_6	Shoes, TShirt	t_{16}	Skirt, TShirt
t_7	Jeans, Skirt	t_{17}	Blouse, Jeans, Skirt
t_8	Jeans, Shoes, Shorts, TShirt	t_{18}	Jeans, Shoes, Shorts, TShirt
t_9	Jeans	t_{19}	Jeans
t_{10}	Jeans, Shoes, TShirt	t_{20}	Jeans, Shoes, Shorts, TShirt

- Scan1: Find all 1-itemsets. Identify the frequent ones.

  Candidates: ~~Blouse~~, Jeans, Shoes, Shorts, Skirt, Tshirt

  Support: ~~3/20~~ 14/20 10/20 5/20 6/20 14/20

  Frequent (Large): Jeans, Shoes, Shorts, Skirt, Tshirt

  Join the frequent items – combine items with each other to generate candidate pairs

L'algorithme Apriori

- Exemple

Scan	Candidates	Large Itemsets
1	Blouse, Jeans, Shoes, Shorts, Skirt, T-Shirt	Jeans, Shoes, Shorts, Skirt, T-Shirt
2	{Jeans, Shoes}, {Jeans, Shorts}, {Jeans, Skirt}, {Jeans, T-Shirt}, {Shoes, Shorts}, {Shoes, Skirt}, {Shoes, T-Shirt}, {Shorts, Skirt}, {Shorts, T-Shirt}, {Skirt, T-Shirt}	{Jeans, Shoes}, {Jeans, Shorts}, {Jeans, T-Shirt}, {Shoes, Shorts}, {Shoes, T-Shirt}, {Shorts, T-Shirt}, {Skirt, T-Shirt}
3	{Jeans, Shoes, Shorts}, {Jeans, Shoes, T-Shirt }, {Jeans, Shorts, T-Shirt}, {Shoes, Shorts, T-Shirt}	{Jeans, Shoes, Shorts}, {Jeans, Shoes, T-Shirt}, {Jeans, Shorts, T-Shirt}, {Shoes, Shorts, T-Shirt}
4	{Jeans, Shoes, Shorts, T-Shirt}	{Jeans, Shoes, Shorts, T-Shirt}
5	empty	empty

L'algorithme Apriori

Exemple

- The next step is to use the large itemsets and generate association rules
- c=50%
- The set of large itemsets is

  L={{Jeans},{Shoes}, {Shorts}, {Skirt}, {TShirt}, {Jeans, Shoes}, {Jeans, Shorts}, {Jeans, TShirt}, {Shoes, Shorts}, {Shoes, TShirt}, {Shorts, TShirt}, {Skirt, TShirt}, {Jeans, Shoes, Shorts}, {Jeans, Shoes, TShirt}, {Jeans, Shorts, TShirt},{Shoes, Shorts, TShirt}, {Jeans, Shoes, Shorts,TShirt} }

- We ignore the first 5 as they do not consists of 2 nonempty subsets of large itemsets. We test all the others, e.g.:

$$confiance(Jeans \rightarrow Shoes) = \frac{\sup port(\{Jeans, Shoes\})}{\sup port(\{Jeans\})}$$

$$= \frac{7/20}{14/20} = 50\% \geq c$$

L'algorithme Apriori

Génération des règles

- Soit L un itemset fréquent, identifier tout les sous-ensembles tel que confiance(l → L-l) ≥ *minconf*

- Les règles candidates de l'itemset fréquent{A, B, C, D}

ABC →D,	ABD →C,	ACD →B,	BCD →A,
A →BCD,	B →ACD,	C →ABD,	D →ABC
AB →CD,	AC → BD,	AD → BC,	BC →AD,
BD →AC,	CD →AB,		

- Si |L|=k, alors le nombre des règles candidates = 2^k-2

(en ignorant $L \to \phi et \phi \to L$)

L'algorithme Apriori

- **Calcule de la valeur de confiance (l → L-l) ?**

  ➢ Le calcule de la confiance d'une règle d'association ne nécessite pas des lectures additionnelles de la base de données

  confiance(X→Y) = support(X→Y)/support(X);

  Expl. confiance(ABC→D) = support(ABC→D)/support(ABC)

➢ On a pas besoin de calculer le support de ABC→D, car = support{A,B,C,D}

➢ On n'a pas besoin de calculer le support de ABC, car il a été déjà calculé

Pourquoi? la propriété anti-monotone du support assure que {ABC} est fréquent → donc on a la valeur de son support

On n'a pas besoin de passer à travers toute la BDD pour calculer la confiance d'une règle

L'algorithme Apriori

Génération des règles

Comment générer des règles d'association de façons efficace à partir des itemsets fréquents ?

⇨ **Propriété anti-monotone de la confiance ?**

- En général, confiance n'a pas de propriété anti-monotone

La confiance de X→Y peut être plus grande, plus petite ou égale à la confiance d'une autre règle X` →Y`, avec X`⊆ X et Y`⊆ Y

L'algorithme Apriori

Génération des règles

On considère les règles issues de l'itemset fréquent Y

Théorème

Si confiance(X→Y-X)<*minconf* alors confiance(X`→Y-X`)<*minconf*

X` est un sous-ensemble de X

Preuve

Discussion en classe !

Question : **démonter que**

- confiance(ABC→D) peut être plus petite ou plus grande que confiance(AB→C)

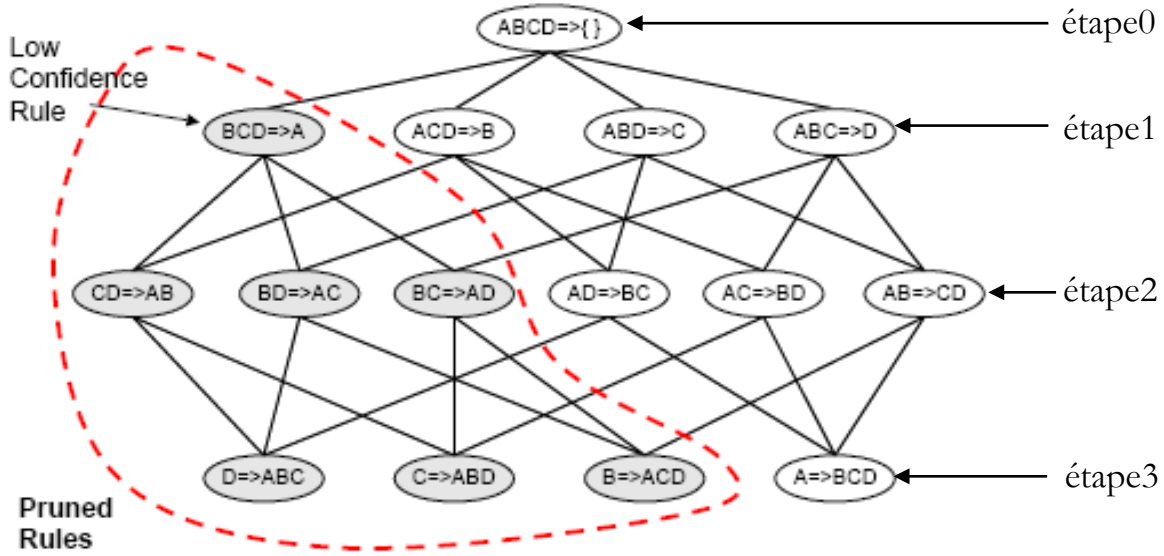
- L={A, B, C, D}

confiance(ABC→D) ≥ confiance(AB→CD) ≥ confiance(A→BCD)

L'algorithme Apriori

Génération des règles

- La génération des règles se fait par étape;
- Le numéro de chaque étape correspond au nombre d'éléments dans la partie droite de la règle;
- On commence par générer des règles avec un nombre d'éléments égal à 1 dans la partie droite;



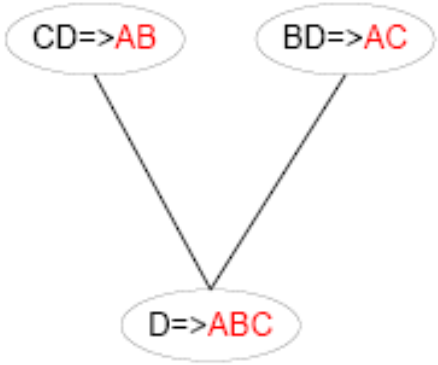
Si la valeur de la confiance de la règle BCD → A est petite **Alors** toutes les règles qui contiennent l'item A dans la partie « conséquent » peuvent être éliminées.

L'algorithme Apriori

Génération des règles

- Les règles candidates sont générées en fusionnant

  Chaque deux règles qui partagent le même préfixe.



Exercice 1

Appliquer l'algorithme Apriori sur l'ensemble des transactions suivant

minsup = 2

TID	items
1	a, b, f
2	b, c,d
3	a, c, d, e, m
4	a, d, e
5	a, b, c
6	a, b, c, d
7	a, l
8	a,b, c
9	a,b, d
10	b, c, e

Exercice 2

Soit les 3 règles suivantes :

R1 : p → q

R2 : p → q, r

R3 : p, r → q

Soit C1, C2 et C3 les valeurs de confiance de R1, R2 et R3 respectivement.

1. Quelle est la relation, en termes d'inégalité, qui existe entre C1, C2 et C3 ?
2. Quelle est la règle qui a la plus petite valeur de confiance ?
3. On suppose que la valeur de support des trois règles sont égaux. Quelle est la règle avec la valeur de confiance la plus élevée ?

L'algorithme Apriori

Avantages et limites de Apriori

- Un algorithme très simple
- Nombre minimum de candidats ⇨ Nombre minimum de calcul du support

Limite majeure

- Nombre de lectures important

• 1re lecture: **Calcul du support** des items ? ensembles fréquents de longueurs 1
• **Génération des candidats** de longueur 2 à partir des ensembles fréquents de longueurs 1
• 2e lecture: **Calcul du support** des candidats de longueur 2 ? Ensembles fréquents de longueurs 2
…
• Arrêt lorsqu'aucun nouvel ensemble fréquent n'est trouvé