Règles Associatives

- Définition et introduction
- Indicateurs de pertinence
- Algorithme d'extraction
- Algorithmes optimisés
- *Vers des règles plus robustes*
- *Règles associatives multi-niveaux*
- Conclusion

1. Objectifs

- Découvrir des règles liant les données avec un bon niveau de probabilité
  - découverte de relations fines entre attributs (ou variables)
  - généralisation des dépendances fonctionnelles
    - NumSS → Sexe
- Règles du style si P(tid,X) alors P(tid,Y)
  - notée: P(tid, X) ➔ P(tid, Y) encore X ➔ Y
- Différents types de règles
  - origine "panier de la ménagère"
  - étendues aux tables multiples et aux attributs continus

Attributs simples

- Table normalisée

ACHAT	TID	PRODUIT
	1	pain
	1	crème
	1	eau
	2	crème
	3	pain
	3	crème
	3	vin

…

Règles mono-dimensionnelles

- simple
  - Achat(tid, "vin") ➔ Achat(tid, "pain")
- conjonctive
  - Achat(tid, "pain") & Achat(tid, "fromage") ➔ Achat(tid, "vin")
- Règles booléennes (attribut discret) mono-dimensionnelles
- notation simplifiée pour règles booléennes 1-D
  - X ➔Y où X et Y sont des ensembles d'items disjoints
  - Formellement I= {Items}, $X \subset I$, $Y \subset I$, $X \cap Y = \varnothing$
  - {"vin"} ➔ {"pain"}
  - {"pain", "fromage"} ➔ {"vin"}

Attributs multi-valués

- Chaque ligne correspond à une ménagère

ACHATS	TID	PRODUITS
	1	pain, crème, eau
	2	crème
	3	pain, crème, vin
	4	eau
	5	crème, eau

- Achats(tid, "pain") ➔ Achats(tid, "crème")
- en raccourci : {"pain"} ➔ {"crème"}

Règles multi-dimensionnelles

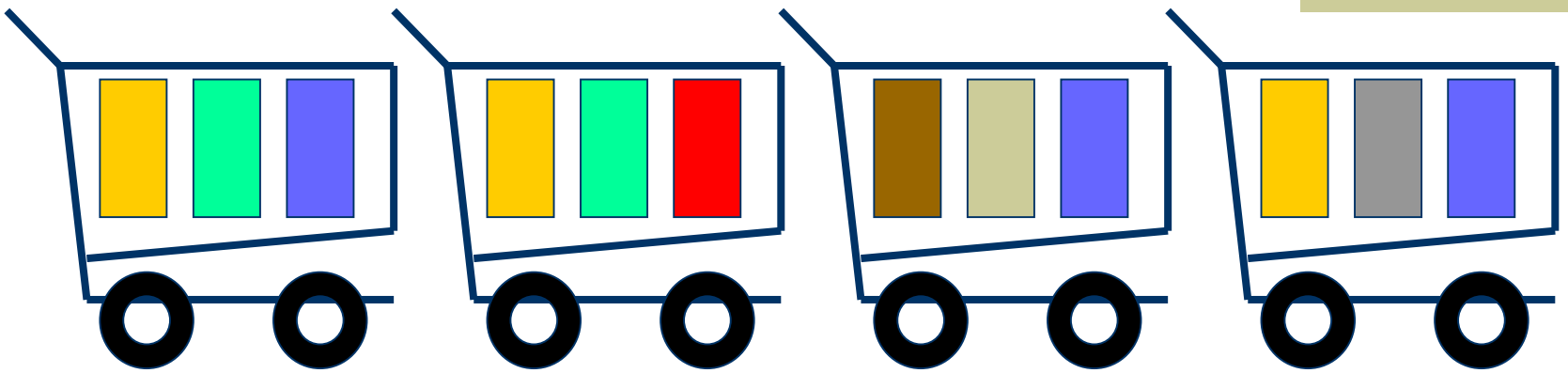
- Mettent en jeu plusieurs attributs
  - Personne(tid, age>50) & Personne(tid, salaire>10000) ➔ Achats(tid, produits="luxe")
  - il est possible de se ramener à 1 table par jointure
    - Personne ⊗ Achats
- Attributs continus (règles quantitatives)
  - Age, Salaire …
  - Possibilité de les discrétiser

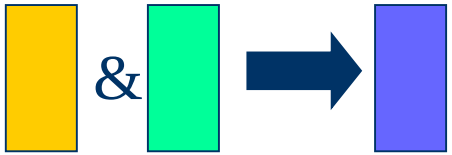
2. Indicateurs de pertinence

- Support : probabilité absolue $P(X \cup Y)$
  - $||X \cup Y||$/ $||BD||$ = % de transactions vérifiant la règle
- Confiance : probabilité conditionnelle $P(Y/X)$
  - $||X \cup Y||/||X||$ = % de transactions vérifiant l'implication

- Règles intéressantes ?
  - par exemple : Support > 0.1 et Confiance > 0.7
  - comment extraire les règles intéressantes ?
  - comment optimiser les calculs d'indicateurs sur des VLDB ?

Exemple 1



Règle	Support	Confiance
	2/4	2/3
	1/4	1/2

Exemple 2

- { "crème" } ➔ { "pain" }

ID	PRODUITS
1	pain, crème, eau
2	crème
3	pain, crème, vin
4	eau
5	crème, eau

Calculs d'indicateurs

- Support = Prob. (crème et pain)

$$\text{Sup} = \frac{\text{nom(tran.contenant crème et pain)}}{\text{nom\_total(tran.)}} = \frac{2}{5} = 0.4$$

- Confiance = Prob(crème et pain / crème)

$$\text{Conf} = \frac{\text{nom(tran. contenant crème et pain)}}{\text{nom(tran. contenant crème)}} = \frac{2}{4} = 0.5 = \frac{\text{sup(crème et pain)}}{\text{sup(crème)}}$$

Support et Confiance

- La confiance se déduit du support
  - conf (X-->Y) = sup(XY) / sup(X)
- Il est donc intéressant de calculer les supports d'abord
- Un ensemble de produits de support plus grand que le support minimum (minsup) est dit fréquent.

Ensembles fréquents

- Un ensemble de taille k est appelé un k-ensemble.
- Tout k-ensemble fréquent est composé de (k-1)-ensembles fréquents
  - en effet, un ensemble ne peut être fréquent si certains sous-ensembles ne le sont pas
  - tout sous-ensemble d'un ensemble fréquent est fréquent

3. Recherche des règles intéressantes

- La détermination des ensembles fréquents permet de trouver un sur-ensemble des règles intéressantes
- La confiance permet de filtrer les règles lors de la génération
- Nécessité de calculer les supports
  - de tous les produits
    - ➔ 1-ensembles fréquents
  - de tous les ensembles susceptibles d'être fréquents
    - ➔ 2-ensembles fréquents, 3-ensembles fréquents, etc.

Algorithme Apriori [Agrawal94]

- Première passe :
  - recherche des 1-ensembles fréquents
  - un compteur par produits
- L'algorithme génère un candidat de taille k à partir de deux candidats de taille k-1 différents par le dernier élément
  - procédure apriori-gen
- Passe k :
  - comptage des k-ensemble fréquents candidats
  - sélection des bons candidats

Apriori – Fréquents itemsets

Apriori(T, ϵ)

$L_1 \leftarrow \{\text{large } 1-\text{itemsets}\}$

$k \leftarrow 2$

while $L_{k-1} \neq \emptyset$

$C_k \leftarrow \{a \cup \{b\} \mid a \in L_{k-1} \wedge b \in \bigcup L_{k-1} \wedge b \notin a\}$

for transactions $t \in T$

$C_t \leftarrow \{c \mid c \in C_k \wedge c \subseteq t\}$

for candidates $c \in C_t$

$count[c] \leftarrow count[c] + 1$

$L_k \leftarrow \{c \mid c \in C_k \wedge \ count[c] \geq \epsilon\}$

$k \leftarrow k + 1$

return $\bigcup_k L_k$

Exemple

SUPPORT = 2

Données en entrée

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Ck

ItemSet	Support
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

ItemSet	Support
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

ItemSet	Support
{1 2 3}	1
{1 3 5}	1
{2 3 5}	2
{1 2 3 5}	1

Lk

ItemSet	Support
{1}	2
{2}	3
{3}	3
{5}	3

ItemSet	Support
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

ItemSet	Support
{2 3 5}	2

L= { {1}, {2}, {3}, {5}, {1 3}, {2 3}, {2 5}, ... }

Pour l = {1 3} génère 2 règles :

R1 : 1 →3 (100%)
R2 : 3 →1 (66%)

R1 plus solide que R2

Apriori – Génération des règles

// Entrée : MinConf, Lk ensembles d'items fréquents

// Sortie : ensemble R de règles d'associations

Rules = ∅ ;

for (k = 2 ; Lk-1 ≠ ∅ ; k++) do {

  Foreach subset S ≠ ∅ of Lk do

    { Conf (S → Lk-S) = Sup(I)/Sup(S)

      If Conf >= MinConf then {

          rule = " S → (Lk-S) " ;

          Rules = Rules ∪ {r} ;}

    }

Answer = Rules ;

4. Comment évaluer efficacement ?

- N passes sur la base
  - une pour 1, 2, …N-ensembles, N étant la taille du plus grand ensemble fréquent
  - comptage des ensembles fréquents par transactions en parcourant la table
- Trouver les produits d'une transaction peut nécessiter de la mémoire si table normalisée

Apriori-tid

- Optimisation de Apriori
  - chaque transaction à un tid
  - liste de tid par k-ensemble
  - Calcul d'un k-ensemble
    - Intersection des deux listes de tid des deux (k-1) ensembles sources
    - La première passe permet d ’éliminer les produits non fréquents (moins de tid)
- Inconvénient
  - les listes sont lourdes et doivent être gardées en mémoire
  - inefficace si les listes ne tiennent pas en mémoire

Bilan Règles Booléennes

- De nombreux algorithmes qui passent à l'échelle
- Les applications restent difficiles
  - le monde réel est plus complexe
    - recherche de séquences
    - recherche de séries temporelles
  - interprétation difficile
    - trop de règles sorties, coefficients ?
- Questions ?
  - Quelles valeurs pour minsup et minconf ?

7. Conclusion

- De nombreuses techniques de recherche de règles
- La plupart passe difficilement à l'échelle
  - limitées à quelques milliers d ’objets
  - échantillonner puis valider sur le reste
- Un bon outil doit proposer plusieurs techniques
- Les problèmes :
  - comment explorer de volumineuses bases de données ?
  - trouver des structures d'indexation intéressantes ?
  - maintenance incrémental des règles ?
  - exploration de types de données complexes ?