Chapitre 1

Fouille de données : Approche de règles d'association

1.1 Introduction

Nous présentons dans ce chapitre une approche assez récente de fouille de données qui est fondée sur la découverte de *règles d'association* à partir d'un ensemble de données qu'on appellera *transactions* (Agrawal et al. 1993). Ce thème est considéré aujourd'hui comme faisant parti des approches d'apprentissage symbolique non supervisé, utilisé dans le domaine de fouille de données (data mining) et d'extraction de connaissances. Un exemple d'application assez courant est l'analyse des logs web sur un serveur web afin de découvrir de comportements utilisateur (web usage mining) dans le but d'adapter ou de personnaliser le site ou de découvrir des comportements types sur certains sites (e-commerce par exemple).

Nous avons vu dans les chapitre précédents trois approches d'apprentissage de concept (espace de versions, arbre de décision et FOIL) dans un contexte d'apprentissage supervisé ; autrement dit, la classe de sortie était présentée est évaluée dans l'ensemble d'apprentissage.

Les règles d'association, étant une méthode d'apprentissage non supervisé, permettent de découvrir à partir d'un ensemble de transactions, un ensemble de règles qui exprime une possibilité d'association entre différents *items*. Une transaction est une succession d'items exprimée selon un ordre donné ; de même, l'ensemble de transactions contient des transactions de longueurs différentes.

Un exemple classique de l'utilité de cette approche est *le panier du ménagère* qui décrit un ensemble d'achats effectué au supermarché ; les règles d'association permet de découvrir de régularités dans l'ensemble de transactions comme par exemple : *Si fromage alors vin rouge*, etc. Ces règles permettent par exemple au gérant de proposer des bons de réductions significatifs sur les achats futurs des clients ! !

1.2 Description du domaine

Un domaine d'application donné doit être décrit par une liste limitée d'atomes qu'on appellera *items*. Par exemple, pour l'application du *panier de ménagère* la liste des items correspond à l'ensemble d'articles disponibles dans le supermarché $[vin, fromage, chocolat, etc.]$. *Un ensemble d'items* est une succession d'items exprimée dans un ordre donné et pré-défini. *Une transaction* est un ensemble d'items. Un ensemble D de transactions correspond à un ensemble d'apprentissage qu'on va utiliser dans la suite pour déterminer les règles d'associations.

Par exemple, deux transactions possibles qui décrivent les achats dans un supermarché sont : $T_1 = \{VinFromageViande\}$ et $T_2 = \{VinFromageChocalat\}$. Remarquer bien qu'un ordre doit être défini sur l'ensemble d'items, autrement dit, dans toutes les transactions qui contiennent Vin et Fromage, Vin doit figurer avant Fromage.

1.2.1 Définition

Un ensemble d'items est dit fréquent ssi il correspond à un motif fréquent dans la base de transactions. Nous définissons le support d'un motif comme étant le nombre de transactions dans D contenant ce motif divisé par card(D). Un seuil minimal de support $minSupp$ est fixé à partir duquel un ensemble d'items est dit fréquent.

Le but de ce cours est de trouver tous les ensembles d'items fréquents qui auront certainement des longueurs différentes, et d'en construire des règles d'association. Par exemple, supposons que ABCD soit un ensemble d'items fréquent de longueur 4, nous construisons la règle $AB \Rightarrow CD$ ssi $support(ABCD)/support(AB) \geq minConf$ nous appelons cette mesure : la confiance associée à une règle, si cette confiance dépasse un certain seuil, alors la règle est induite.

Nous présentons dans la suite deux algorithmes qui génèrent tous les ensembles d'items fréquents suivi de l'algorithme qui construit les règles d'associations à partir de ces ensembles.

1.3 Deux algorithmes pour la générations des sous-ensembles fréquents

Nous présentons deux algorithmes pour la génération des sous-ensembles d'items fréquents, le premier étant un algorithme très coûteux en terme d'accès à la base de transactions :D, le deuxième optimise le coût d'accès.

1.3.1 Terminologie

Nous adoptons dans la suite la terminologie suivante :

L_k est l'ensemble constitué des sous-ensembles d'items fréquents de longueur k.

C_k est un ensemble constitué des sous-ensembles d'items candidats de longueur k, notons bien que $L_k \subset C_k$

Propriété des sous-ensembles fréquents Soit X_k un sous-ensemble d'items fréquent, tous les sous-ensembles d'items contenus dans X_k et qui soient de longueurs inférieurs à k sont fréquents.

Par exemple si $ABCD$ est un sous-ensemble d'items fréquents, alors, les sous ensembles :ABC,ABD, BCD, AB,AC,BC,BD,CD,A,B,C,D les sont aussi.

1.3.2 L'algorithme Apriori

L'algorithme Apriori est donné par la suite d'instructions suivantes :

Algorithm 1 L'algorithme Apriori

Calculer L_1
$k \leftarrow 2$
TANTQUE $L_{k-1} <> \phi$ **FAIRE**
  $C_k \leftarrow apriori - gen(L_{k-1})$
  TANTQUE $t \in D$ **FAIRE**
    $C_t = sousensemble(C_k, t)$
    TANTQUE $c \in C_t$ **FAIRE**
      c.count++
    FIN TANTQUE
  FIN TANTQUE
  $L_k \leftarrow \{c \in C_k | c.count \geq minSup\}$
  $k \leftarrow k + 1$
FIN TANTQUE
RETOURNER $\bigcup_k L_k$

Noter bien que la donnée C_k (ainsi que L_k) est *un ensemble d'enregistrements* contenant deux champs :

itemset ce champs contient le sous ensemble d'items,

count ce champs contient la fréquence de cet ensemble dans la base de transactions.

L'algorithme Apriori permet de découvrir les sous-ensembles d'items fréquents en partant de ceux dont la longueur est 1 et en augmentant la longueur au fur et à mesure. Cet algorithme est fondé sur la propriété des sous ensembles d'items fréquents. Il fait appel à deux algorithmes :

apriori-gen L'algorithme apriori-gen est constitué de deux phases la première phase nommé *Joindre* trouve tous les candidats possibles de longueur k à partir de l'ensemble L_{k-1}, La deuxième phase *Effacer* efface de C_k les éléments qui ne vérifient pas la propriété des sous ensemble fréquents. Nous donnons le code de joindre en SQL comme suit : Soint $p, q \in L_{k-1}$

1. insert into C_k
2. select $p[1], p[2], ..p[k-1], q[k-1]$

3. from p,q
4. Where $p[1] = q[1]..p[k-2] = q[k-2]$,$p[k-1] < q[k-1]$

sous-ensemble L'algorithme sous-ensemble calcule le sous ensemble $C_t \subseteq C_k$ qui correspond à des sous-ensembles présents dans les transactions contenues dans D.

Par exemple si $L_3 = \{\{123\}, \{124\}, \{134\}, \{135\}, \{234\}\}$, la phase $joindre$ donne comme résultat $C_4 = \{\{1234\}, \{1345\}\}$ ensuite la phase $effacer$ donne le résultat : $C_4 = \{\{1234\}$ car l'élément $\{145\}$ n'est pas dans L_3 et donc $\{1345\}$ est effacé.

L'inconvénient majeur de cet algorithme est le nombre considérable de l'accès à la base de données D. Une amélioration qui consiste à intégrer les identificateurs des transactions est proposée dans la sous-section suivante.

1.3.3 L'algorithme AprioriTid

L'algorithme AprioriTid est donné par la suite d'instructions suivantes :

Algorithm 2 L'algorithme AprioriTid

Calculer L_1
$\hat{C_1} \leftarrow D$
$k \leftarrow 2$
TANTQUE $L_{k-1} <> \phi$ **FAIRE**
  $C_k \leftarrow apriori - gen(L_{k-1})$
  $\hat{C_k} \leftarrow \phi$
  TANTQUE $t \in \hat{C_{k-1}}$ **FAIRE**
    $C_t = \{c \in C_k | (c[1].c[2]..c[k-1]) \in t.ensemble \wedge (c[1].c[2]..c[k-2].c[k]) \in t.ensemble\}$
    TANTQUE $c \in C_t$ **FAIRE**
      c.count++
    FIN TANTQUE
    SI $C_t <> \phi$ **ALORS**
      $\hat{C_k} \leftarrow + < t.TID, C_t >$
    FIN SI
  FIN TANTQUE
  $L_k \leftarrow \{c \in C_k | c.count \geq minSup\}$
  $k \leftarrow k + 1$
FIN TANTQUE
RETOURNER $\bigcup_k L_k$

Noter bien que la donnée $\hat{C_k}$ est *un ensemble d'enregistrements* contenant deux champs :

ensemble des itemsets ce champs contient un ensemble des sous ensemble d'items,

TID ce champs contient l'identificateur de la transaction dans D qui contient cet ensemble d'itemsets.

L'amélioration qu'apporte cet algorithme par rapport au précédent est le fait de stocker à chaque itération les identificateurs des transactions contenant les sous-ensembles fréquents dans l'ensemble $\hat{C_k}$. La

propriété des sous-ensembles fréquents nous permet de voir clairement que les transactions sollicitées à l'itérations $k+1$ seront parmi celles identifiées à l'itération k. Par conséquent, l'accès à D est effectuée seulement pour l'itération 1.

1.4 Génération des règles

Nous adoptons une approche descendante de recherche de règles d'association à partir des ensembles d'items fréquents trouvés par l'un des algorithmes précédents. Cette approche est justifiée par deux propriétés de redondance sur l'ensemble de règles d'association qu'on définira dans la suite.

La première propriété est celle de la *redondance simple* : soient les deux règles :

R1 $A \Rightarrow B, C$

R2 $A, B \Rightarrow C$

Nous remarquons que confiance(R1)=support(ABC)/support(A), confiance(R2)=support(ABC)/support(AB) et par conséquent, confiance(R2)>confinace(R1). Autrement dit, il est intéressant pour un ensemble fréquent donné de commencer par rechercher les règles ayant le nombre de conditions minimal ; si une telle règle est acceptée, les autres règles dérivées du même sous-ensemble fréquent ayant plus de conditions incluant la condition minimale, auront de taux de confiance plus élevés. Autrement dit si $A \Rightarrow B, C$ alors $A, B \Rightarrow C$ et $A, C \Rightarrow B$.

La deuxième propriété est celle de la *redondance stricte* ; soient les deux règles :

R1 $A \Rightarrow B, C, D$

R2 $A \Rightarrow B, C$

Nous remarquons que confiance(R1)=support(ABCD)/support(A), confiance(R2)=support(ABC)/support(A) et par conséquence, confiance(R2)>confinace(R1). Autrement dit, il est judicieux de commencer la recherche de règles d'association dans les ensembles d'items fréquents les plus grands, si une telle règle est acceptée, toutes les règles inclues seront encore considérées.

La stratégie de recherche de règles que nous adoptons est la suivante :

1. Trouver les plus grands ensembles d'items fréquents.
2. En extraire les règles de confiance supérieure au seuil, ayant des conditions minimales.
3. Pour les configurations n'ayant pas engendré des règles d'association, explorer d'une façon analogue, les sous-ensembles immédiats.

1.5 Exercices

1.5.1 Génération des règles d'association

Soit D la base de transactions suivante :

100 1 3 4

200 2 3 5

300 1 2 3 5

400 2 5

Exercice 1 Appliquer l'algorithme Apriori pour extraire les sous-ensembles d'items fréquents avec $minSupp = 2$.

Exercice 2 Appliquer l'algorithme AprioriTid pour extraire les sous-ensembles d'items fréquents avec $minSupp = 2$.

Exercice 3 Générer les règles d'association avec $minConf = 1$

1.5.2 Amélioration de la prédiction

Soit D la base de transactions contenant un ensemble de transaction décrivant des achats de produits dans l'ensemble $\{A, B, C, D, E\}$

1 A D

2 A B C

3 A E

4 A D E

5 B D

Exercice 4 Considérons seulement l'ensemble $\{A, B, C\}$; Trouver l'ensemble de règles qui permettent de prédire l'achat de deux produits tout en améliorant la prédiction par rapport à la mesure statistique induite par la base de données.

Exercice 5 Répéter la même question pour trouver l'ensemble de règles qui permettent de prédire l'achat d'un produit.