Suppression des Itemsets Clés Non Essentiels en Classification basée sur les Règles d'Association

Viet Phan-Luong

Université de Provence
Laboratoire d'Informatique Fondamentale de Marseille
(LIF - UMR CNRS 6166)
CMI, 39 rue F. Joliot Curie
13453 Marseille, France
viet.phanluong@lif.univ-mrs.fr

Résumé. En classification basée sur les règles d'association, les itemsets clés sont essentiels : la suppression des itemsets non clés n'affecte pas la précision du classifieur en construction. Ce travail montre que parmi ces itemsets clés, on peut s'intéresser seulement à ceux de petites tailles. Plus loin encore, il étudie une généralisation d'une propriété importante des itemsets non clés et montre que parmi les itemsets clés de petites tailles, il y a ceux qui ne sont pas significatifs pour la classification. Ces itemsets clés sont dits non essentiels. Ils sont définis via un test de χ^2. Les expériences menées sur les grands jeux de données montrent que l'optimisation par la suppression de ces itemsets est correcte et efficace.

1 Introduction

Etant donné un ensemble d'objets et un ensemble d'étiquettes de classes, le problème de classification est de chercher une fonction pour attribuer à chaque objet une étiquette de classe. Une telle fonction est appelée un classifieur. Les constructions de ces classifieurs sont en général basées sur les données d'exemples (d'entraînement). Il existe plusieurs méthodes de classification, telles que l'arbre de décision Quinlan (1993), la méthode naïve-Bayes Duda et Hart (1973), les méthodes basées sur les règles Clark et Niblett (1995); Cohen (1995). Ce papier présente une approche à la construction de classifieurs basée sur les règles classe-associations Lent et al. (1997); Liu et al. (1998); Li et al. (2001), en utilisant une structure d'arbre de préfixes pour l'extraction des itemsets fréquents et les règles d'association Agrawal et al. (1993).

Dans les approches telles que CMAR Li et al. (2001), HARMONY Wang et Karypis (2005), par optimisations, les règles d'association sont essentiellement construites sur les itemsets clés Bastide et al. (2000). Ce présent travail montre que parmi ces itemsets clés, on peut s'intéresser seulement à ceux de petites tailles. Ensuite, via un test de χ^2, il montre que parmi ces derniers, il existe encore ceux qui ne sont pas significatifs pour la classification. Ces itemsets clés sont dits non essentiels. Les résultats d'expérimentations sur les grands jeux de données de *UCI* Coenen (2004) montrent que l'optimisation par la suppression de ces itemsets est correcte et efficace.

2 Préliminaires

Un jeu de données est un triplet $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$, où $\mathcal{O}, \mathcal{I}, \mathcal{R}$ sont des ensembles finis et non vides. Un élément de $\mathcal{I}$ est appelé un *item*, un élément de $\mathcal{O}$ est appelé un objet (ou une transaction) représenté par un identifiant, et $\mathcal{R}$ est une relation binaire entre $\mathcal{O}$ et $\mathcal{I}$. Une paire (o, i) de $\mathcal{R}$ signifie que l'item i est une valeur attribuée de l'objet o. Un *itemset* est un sous-ensemble de $\mathcal{I}$. Un k-itemset est un itemset avec k items ; k est la *taille* (ou *la longueur*) de l'itemset.

La *connexion de Galois* Ganter and Wille (1999) est une paire de fonctions (f, g), où
$g(I) = \{o \in \mathcal{O} \mid \forall i \in I, (o, i) \in \mathcal{R}\}$ et $f(O) = \{i \in \mathcal{I} \mid \forall o \in O, (o, i) \in \mathcal{R}\}$.

En fait, $g(I)$ est l'ensemble des objets de $\mathcal{O}$ qui ont en commun tous les items de I, et f est la fonction duale de g. La fonction g est anti-monotone : pour tout $I_1, I_2 \subseteq \mathcal{I}$, si $I_1 \subseteq I_2$ alors $g(I_2) \subseteq g(I_1)$.

Soit un jeu de données $\mathcal{D} = (\mathcal{O}, \mathcal{I}, \mathcal{R})$ et sa connexion de Galois (f, g). Les *opérateurs de fermetures de Galois* sont les fonctions suivantes : $h = f\ o\ g$ et $h' = g\ o\ f$, où o dénote la composition de fonctions. Les fonctions h et h' sont monotones. Soit I un itemset. Alors $h(I) = f(g(I))$ est appelée la *fermeture* de I. En effet, pour tout $I \in \mathcal{I}$, $I \subseteq h(I)$ (Extension) et $h(h(I)) = h(I)$ (Idempotence). Un itemset I est dit *fermé* si $I = h(I)$. I est appelé un itemset *clé* (ou un *générateur minimal* Bastide et al. (2000)) si $\forall I' \subseteq I, h(I') = h(I)$ implique $I = I'$.

Le *support* de I est $sup(I) = card(g(I))$, où $card$ dénote la cardinalité. Il est clair que si $I \subseteq I'$ alors $sup(I) \geq sup(I')$. Soit $minsup$ un seuil de support. I est dit *fréquent* si $sup(I) \geq minsup$. I est un itemset clé (respectivement, fermé) fréquent si I est fréquent et I est aussi un itemset clé (respectivement, fermé).

Une *règle d'association (RA)* est une expression de la forme $X \rightarrow Y$, où X et Y sont des itemsets disjoints. Soit $r = X \rightarrow Y$ une RA. Alors $LHS(r)$ et $RHS(r)$ dénotent respectivement la partie gauche et la partie droite de r. Le *support* de r est : $sup(r) = sup(X \cup Y)$. La *confiance* de r est : $conf(r) = sup(r)/sup(X)$.

En classification, on considère un ensemble $\mathcal{C}$ des éléments appelés les *étiquettes de classes*. Une *règle classe-association (RCA)* est une expression de la forme $X \rightarrow c$, où $X \subseteq \mathcal{I} - \mathcal{C}$ et $c \in \mathcal{C}$. Soit $r = X \rightarrow c$ une RCA. Un objet o est *couvert par* r si o a tous les items de X ; on dit que o *satisfait* r. Un objet o est *classifié correctement par* r si o satisfait r et o est effectivement de la classe d'étiquette c. Le *support* de r, $sup(r) = sup(X \cup \{c\})$, est le nombre d'objets de $\mathcal{D}$ qui sont classifiés correctement par r. La *confiance* de r, $conf(r) = sup(r)/sup(X)$, représente la fréquence d'applications correctes de r dans $\mathcal{D}$.

Exemple 1 *Le tableau 1 représente un jeu de données d'entraînement dans lequel* $a \rightarrow C$ *est un RCA de support* 2 *et de confiance* 0.67*, car* $sup(aC) = 2$ *et* $sup(a) = 3$*. Un autre exemple de* RCA *:* $ab \rightarrow C$*. On a* $sup(ab \rightarrow C) = 1$*, et* $conf(ab \rightarrow C) = 0,5$*.*

Sur les RCAs un ordre partiel, appelé *l'ordre de précédence* et noté $\preceq$, est défini comme suit : Soient r et r' des RCAs, $r \preceq r'$ (lire r précède r') si
– $conf(r') < conf(r)$, ou
– $conf(r) = conf(r')$ et $sup(r') < sup(r)$, ou
– $conf(r) = conf(r')$ et $sup(r) = sup(r')$ et $card(LHS(r)) < card(LHS(r'))$.

Oid	Itemsets	Cls
1	acd	C
2	abe	C
3	abd	C'
4	bce	C'

TAB. 1 – *Un jeu de données d'entraînement*

3 Travaux liés et contributions

Il existe plusieurs méthodes pour calculer les itemsets fréquents Agrawal et Srikant (1994); Park et al. (1995); Lent et al. (1997); Zaki et al. (1997); Bayardo (1998); Han et al. (2000). Ces méthodes peuvent être classées en deux catégories. En première catégorie, Apriori Agrawal et Srikant (1994) et ses variantes calculent des itemsets fréquents par niveau : les k-itemsets candidats ($k > 1$) sont générés à partir des $(k-1)$-itemsets fréquents. La seconde catégorie consiste en méthodes qui se développent sur FP-growth Han et al. (2000). Cet algorithme utilise les arbres de préfixes pour le stockage compact des jeux de données en mémoire centrale. A chaque étape, il calcule un itemset préfixe (fréquent) et le jeu de données restreint par rapport à ce préfixe, puis récursivement s'applique au jeu de données restreint. Ces méthodes sont adaptées pour le calcul des itemsets fermés (ou clés) fréquents Pasquier et al. (1999); Zaki (2000); Bastide et al. (2000); Pei et al. (2000); Stumme et al. (2002); Phan-Luong (2002). On peut trouver un intéressant état de l'art comprenant les récents algorithmes de calcul des itemsets fermés dans Yahia et al. (2006).

Le problème de classification est largement étudié Lim et al. (2000); Liu et al. (1998); Li et al. (2001); Wang et Karypis (2005). Les algorithmes classiques Quinlan et Cameron-Jones (1993); Cohen (1995); Yin et Han (2003) calculent à chaque fois une règle, en utilisant des heuristiques basées sur l'analyse statistique. En contraste, les algorithmes basés sur l'extraction des règles classe-associations, comme CBA Liu et al. (1998), CMAR Li et al. (2001), cherchent un ensemble de règles de confiances élevées construites sur les itemsets fréquents.

CBA adapte Apriori pour extraire les RCAs. Les règles sont triées dans l'ordre de précédence avant d'être sélectionnées pour le classifieur. Une règle est sélectionnée si elle classifie correctement au moins un des objets d'entraînement. Dans ce cas, tous les objets couverts par la règle sont écartés du processus de sélection. Par l'ordre de précédence, le classifieur préfère les règles formées sur les itemsets clés.

Basé sur l'idée de CBA, CMAR Li et al. (2001) adapte FP-growth pour extraire les RCAs. En plus de l'ordre de précédence, CMAR considère la corrélation entre la partie gauche de règle et l'étiquette de classe. D'ailleurs, CMAR permet à chaque objet d'être couvert par plusieurs règles et propose un schéma de classification basé sur de multiples règles.

HARMONY Wang et Karypis (2005) utilise la même stratégie que FP-growth pour extraire les RCAs. Par défaut, les items d'un jeu de données restreint sont triés dans l'ordre croissant des coefficients de corrélation entre le préfixe correspondant et ces items. Les items et les jeux de données restreints non prometteurs sont exclus de l'espace de recherche. Par ces exclusions, le classifieur préfère les règles formées sur les itemsets clés.

A la différence de CBA et CMAR, pendant l'extraction des règles, HARMONY maintient pour chaque objet une liste de règles de confiance la plus élevée qui classifient correctement l'objet. A la fin du processus de l'extraction, HARMONY regroupe les règles sélectionnées selon leurs étiquettes de classes et les trie dans l'ordre décroissant des confiances et des supports. Pour classifier un objet de test ti, HARMONY calcule, pour chaque groupe du classifieur, la somme de confiances de k premières règles de confiance la plus élevée qui couvrent ti. La classe avec la somme la plus grande est sélectionnée pour prédire la classe de ti.

Contribution : La contribution de ce travail consiste en :

– L'étude d'une relation entre les supports des itemsets qui s'incluent, la généralisation d'une propriété importante des itemsets non clés, et la notion d'itemset clé non essentiel.

– L'application de la notion d'itemset clé non essentiel pour optimiser la construction de classifieurs basée sur les RCAs.

– L'approche n'explore pas les itemsets de toutes tailles. Elle se limite aux itemsets clés de petites tailles (≤ 5). Par conséquent, elle peut exploiter les règles de supports très bas, mais avec de meilleures confiances.

Les résultats expérimentaux sur les grands jeux de données catégoriels montrent qu'en moyenne l'approche est efficace, en comparaison avec les approches importantes, et que l'optimisation sur les itemsets clés non essentiels est correcte et efficace.

4 Itemsets clés non essentiels

4.1 Itemsets non clés

Il est facile de voir que si X est un itemset clé, alors pour tout itemset $I \neq X$, si $X \subset I \subseteq h(X)$, alors $sup(X) = sup(I) = sup(h(X))$. Cette propriété permet de définir une relation d'équivalence sur les RAs : une règle $X \rightarrow Y$, $X \cap Y = \emptyset$, avec X et Y étant des itemsets clés, représente une classe d'équivalence de règles, par rapport au support et à la confiance. Ces RAs représentatives sont très intéressantes : leur nombre est plus réduit, et avec les parties gauches réduites (au sens de clés), elles peuvent être appliquées à un plus large nombre d'objets, par rapport aux règles aux itemsets non clés.

Proposition 1 *Soient $I \rightarrow J$ et $X \rightarrow Y$ les RAs telles que $I \cap J = X \cap Y = \emptyset$ et $X \subset I \subseteq h(X)$ et $Y \subset J \subseteq h(Y)$. Alors, $sup(X \rightarrow Y) = sup(I \rightarrow J)$ et $conf(X \rightarrow Y) = conf(I \rightarrow J)$.*

Preuve. En effet, $sup(X \rightarrow Y) = card(g(X \cup Y)) = card\ (g(X) \cap g(Y))$. De même, $sup(I \rightarrow J) = card(g(I) \cap g(J))$. Maintenant, comme $X \subset I \subseteq h(X)$ et $sup(X) = sup(I)$, on a $g(X) = g(I)$. De même, $g(Y) = g(J)$. Donc, $sup(X \rightarrow Y) = sup(I \rightarrow J)$. Avec $sup(X) = sup(I)$, on déduit $conf(X \rightarrow Y) = conf(I \rightarrow J)$. $\diamond$

L'application de la proposition 1 aux RCAs résulte en :

Corollaire 1 *Soient X_1 et X_2 des itemsets tels que $X_1 \subseteq X_2$ et $sup(X_1) = sup(X_2)$. Alors pour toute étiquette de classe C, $sup(X_1 \rightarrow C) = sup(X_2 \rightarrow C)$ et $conf(X_1 \rightarrow C) = conf(X_2 \rightarrow C)$.*

D'après le corollaire 1, en classification on peut construire les classifieurs avec seulement les RCAs aux itemsets clés. La fermeture de X, i.e. $h(X)$, et tout I tel que $X \subset I \subseteq h(X)$, sont des itemsets *non clés*. Ces itemsets ne sont pas intéressants pour les classifieurs. La propriété suivante est utile pour la suppression de ces itemsets Phan-Luong (2002).

Proposition 2 *Soient X, Y, Z des itemsets. Si $X \subseteq Y \subseteq Z$ et $g(X) = g(Y)$, alors $g(Z) = g((Z - Y) \cup X)$.*

Preuve. Si $X \subseteq Y \subseteq Z$, alors clairement $Z = (Z - Y) \cup X \cup Y$. D'où, $g(Z) = g(Z-Y) \cap g(X) \cap g(Y)$. Avec la condition $g(X) = g(Y)$, on a : $g(Z) = g(Z-Y) \cap g(X) = g((Z - Y) \cup X)$. Ainsi, $g(Z) = g((Z - Y) \cup X)$. $\diamond$

Bastide et al. (2000) ont montré que, avec les conditions de la proposition 2, on a $g(Z) = g(Z - (Y - X))$. Il est facile de voir que $(Z - Y) \cup X = Z - (Y - X)$.

Une conséquence directe de la proposition 2 est la suivante : si $X \subseteq Y \subseteq Z$ et $sup(X) = sup(Y)$, alors $sup(Z) = sup((Z - Y) \cup X)$. On peut en déduire que si Y n'est pas une clé, alors tout super ensemble de Y ne l'est pas. Donc, pour la construction de classifieurs, il est plus intéressant de commencer par les itemsets de petites tailles. Quand on ajoute un nouvel item dans un itemset courant, si le résultat n'est pas un itemset clé, alors il n'est plus intéressant de continuer la recherche avec ce résultat. Dans la suite, nous étudions une généralisation de cette propriété pour pousser encore l'optimisation sur les itemsets clés.

4.2 Itemsets clés non essentiels

Proposition 3 *Si $X \subseteq Y$, alors $g(X) - g(Y) = g(X) - g(Y - X)$.*

Preuve. Supposons que $X \subseteq Y$. Alors $g(Y) \subseteq g(X)$. Soit $o \in g(X) - g(Y)$. Alors $o \in g(X)$ et $o \notin g(Y)$. Or $o \in g(X)$ ssi o a tous les items de X, et $o \notin g(Y)$ ssi o n'a pas au moins un item i de Y. Donc, $o \in g(X) - g(Y)$ ssi o a tous les items de X et o n'a pas au moins un item i de $Y - X$, ssi $o \in g(X)$ et $o \notin g(Y - X)$. $\diamond$

Corollaire 2 *Si $X \subseteq Y \subseteq Z$, alors $g((Z-Y) \cup X) - g(Z) = g((Z-Y) \cup X) - g(Y - X)$.*

Proposition 4 *Si $X \subseteq Y \subseteq Z$, alors $g((Z - Y) \cup X) - g(Z) \subseteq g(X) - g(Y)$*

Preuve. D'après le corollaire 2, $g((Z - Y) \cup X) - g(Z) = g((Z - Y) \cup X) - g(X - Y)$. Comme $g((Z - Y) \cup X) \subseteq g(X)$, en remplaçant l'expression $g((Z - Y) \cup X)$ sur la droite de l'égalité par $g(X)$, nous avons $g((Z - Y) \cup X) - g(Z) \subseteq g(X) - g(X - Y)$. D'après la proposition 3, $g((Z - Y) \cup X) - g(Z) \subseteq g(X) - g(Y)$. $\diamond$

A cette étape on peut voir que la proposition 2 est une directe conséquence de la proposition 4. En effet, lorsque $g(X) = g(Y)$, on a $\emptyset \subseteq g((Z-Y) \cup X) - g(Z) \subseteq g(X) - g(Y) = \emptyset$. D'où, $g((Z-Y) \cup X) - g(Z) = \emptyset$. Comme $g(Z) \subseteq g((Z-Y) \cup X)$, on a $g((Z-Y) \cup X) = g(Z)$.

Corollaire 3 *Si $X \subseteq Y \subseteq Z$, alors $sup((Z - Y) \cup X) - sup(Z) \leq sup(X) - sup(Y)$.*

Parmi les itemsets clés, il existe ceux qui ne sont pas très différents entre eux, aux niveaux de supports et d'étiquettes de classes. Ces itemsets peuvent être considérés comme la même chose, dans le sens où les RCAs construites avec eux portent les informations très similaires sur le support et la confiance. Ces itemsets sont définis via un test χ^2 comme suit.

Définition 1 *Soient X, Y des itemsets tels que $X \subseteq Y$. Soit C_X (respectivement, C_Y) l'ensemble des étiquettes de classes telles que $\forall A \in C_X, sup(XA) \neq 0$ (respectivement, $\forall A \in C_Y, sup(YA) \neq 0$). Définir $X \simeq_{C_X} Y$ ssi*

$$\frac{(sup(X) - sup(Y))^2}{sup(X)} + \Sigma_{A \in C_Y \cap C_X} \frac{(sup(XA) - sup(YA))^2}{sup(XA)} +$$

$$\Sigma_{A \in C_X - C_Y} sup(XA) < \chi_{|C_X|,\alpha} \quad (1)$$

où $|C_X|$ dénote la cardinalité de C_X et $\chi_{|C_X|,\alpha}$ est l'écart χ^2 au degré de liberté $|C_X|$ et au risque d'erreur α.

Pour les itemsets X et Y satisfaisant la définition 1, en construction d'un classifieur, on peut s'intéresser seulement aux itemsets clés comme X avec la taille la plus petite et oublier les itemsets comme Y. Les itemsets clés comme Y sont appelés *itemsets clés non essentiels*.

Pour élaguer les itemsets comme Y, on peut espérer une propriété similaire à celle spécifiée en proposition 2. C'est-à-dire, commençant par les itemsets clés de petites tailles, et lors de l'ajout d'un nouvel item dans l'itemset courant, si le résultat est une clé et, en comparaison avec l'itemset courant, la définition 1 est satisfaite, alors on peut arrêter la recherche sur cet itemset courant. Précisément, si $X \subseteq Y \subseteq Z$ et $X \simeq_{C_X} Y$, alors on attend que

$$(Z - Y) \cup X \simeq_{C_{(Z-Y)\cup X}} Z.$$

Pour cette conjecture, on peut avoir trois arguments importants. D'abord, bien que

$$\chi_{|C_{(Z-Y)\cup X}|,\alpha} \leq \chi_{|C_X|,\alpha},$$

la différence entre $\chi_{|C_{(Z-Y)\cup X}|,\alpha}$ et $\chi_{|C_X|,\alpha}$ ne soit pas importante, surtout quand on considère les itemsets de petites tailles. Ensuite, par le corollaire 3, on a :

$$(sup((Z - Y) \cup X) - sup(Z))^2 \leq (sup(X) - sup(Y))^2.$$

Et dernièrement, d'après la définition 1 :

$$\frac{(sup(X) - sup(Y))^2}{sup(X)} + \Sigma_{A \in C_Y \cap C_X} \frac{(sup(XA) - sup(YA))^2}{sup(XA)} +$$

$$\Sigma_{A \in C_X - C_Y} sup(XA) < \chi_{|C_X|,\alpha}.$$

D'où,

$$(sup((Z - Y) \cup X) - sup(Z))^2 \leq (sup(X) - sup(Y))^2$$

$$< sup(X) * \chi_{|C_X|,\alpha}.$$

Nous n'avons pas prouvé formellement cette conjecture. Cependant, nous l'appliquerons dans la méthode de construction de classifieurs, basée sur une structure d'arbre de préfixes pour l'extraction des itemsets.

5 Extraction de classifieurs

5.1 Extraction de règles de classe-association

L'approche utilise une technique d'énumération de sous-ensembles Rymon (1992) sur un arbre de préfixes pour l'extraction de RCAs. Commençant par un arbre de préfixes p vide, on lit successivement les objets d'un jeu de données f pour mettre à jour p. Pour chaque objet $o = (l : c)$, où l est la liste d'items de o, et c son étiquette de classe, les sous-itemsets de l sont énumérés dans l'ordre lexicographique et stockés avec l'étiquette c, dans l'arbre p (fonction $Build$). Leurs supports ainsi que les occurrences de leurs étiquettes de classes sont mis à jour le long de la lecture du jeu de données. L'approche peut adopter le calcul des itemsets fréquents par niveau comme Apriori. La fonction $LevelBuild$ est un exemple spécifique de ce calcul. Par contraste avec Apriori : (i) l'approche ne génère pas de candidats, (ii) le calcul de supports se fait pendant la construction de l'arbre, et (iii) l'approche peut commencer par les i-itemsets et passer de k-itemsets aux $(k + j)$-itemsets, avec $i, j \geq 1$.

```
// Construire l'arbre p avec les sous-itemsets de taille maximale max du jeu de données f.
function LevelBuild(f, p, max) {
    for (i = 1 ; i ≤ max ; i++) {
        Build(f, p, i) ;              // énumérer les itemsets de taille maximale i
        PruneInfrq(p, minsup) ;       // élaguer les itemsets non fréquents
    }
}
```

L'approche applique la contrainte de support aux i-itemsets, seulement pour $i \leq 2$. Cependant, quand elle sélectionne les règles pour le classifieur, seules les règles dont la confiance et le support sont maximaux, par rapport à chaque objet d'entraînement, sont retenues.

Exemple 2 *La figure 1 représente le résultat de la mise à jour de l'arbre vide avec les objets* $(acd : C)$ *et* $(abe : C)$. *A un noeud* N *est associée une paire* $(sup(N), lc(N))$. *Un élément de* $lc(N)$ *est noté* $x : k_x$, *où* x *est une étiquette de classe et* k_x *le nombre d'occurrences de* x.

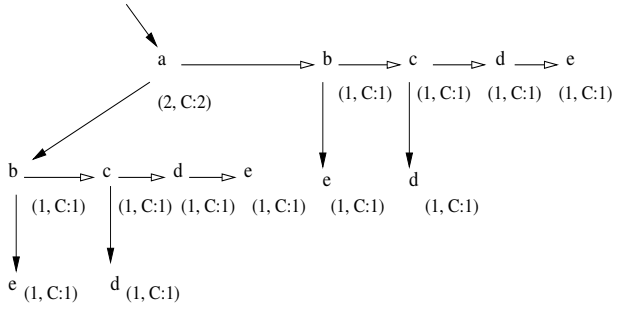

FIG. 1 – *L'arbre de préfixes des objets* $(acd : C)$ *et* $(abe : C)$.

5.2 Réduction de l'arbre de préfixes

Nous avons vu dans la section 4 que les RCAs formées sur les itemsets clés sont suffisantes pour les classifieurs (Corollaire 1). Nous avons aussi défini la notion d'itemset clé non essentiel et pensé que ces itemsets se comporteront comme des itemsets non clés (Corollaire 3) et auront la même propriété spécifiée en proposition 2.

La suppression des itemsets non clés et les itemsets clés non essentiels se fait en deux étapes. La première étape s'applique à l'arbre de préfixes (fonction $TreeReduction$). Si un noeud N a le même support que son prédécesseur ou si l'inéquation 1 de la définition 1 pour N et son prédécesseur est satisfaite, alors le sous-arbre avec N à la racine sera coupé. Pour tout noeud restant N, on réduit la liste $lc(N)$ en enlevant les paires (x, kx) telles que la valeur de kx n'est pas maximale dans $lc(N)$ (fonction $RedCls$).

Les paramètres de $TreeReduction$ sont les suivants : N est un noeud de l'arbre, k et $precls$ représentent respectivement le support du prédécesseur de N et la liste des étiquettes de classes associées au prédécesseur de N avec leurs nombres d'occurrences, et $chi[\,]$ est un tableau de valeurs χ^2 pour le test de clés non essentiels.

Function **TreeReduction(**N**,** k**,** $precls$**,** $chi[\,]$**)***{*
if N *is not null then {* *// $chd(N)$: fils de N ; $sib(N)$: frère de N.*
  if N *is not treated then {*
    if $sup(N) = k$ *or evaluation of expression 1 (Definition 1),*
      using N, k, $precls$, $chi[\,]$, *is true,*
    then { prune N *; TreeReduction(*$sib(N)$, k, $precls$, $chi[\,]$*); }*
    else { mark N *as treated ;* $RedCls(lc(N))$ *;*
      TreeReduction($chd(N)$, $sup(N)$, $lc(N)$, $chi[\,]$*);*
      TreeReduction($sib(N)$, k, $precls$, $chi[\,]$*); }*
    }
  else { TreeReduction($chd(N)$, $sup(N)$, $lc(N)$, $chi[\,]$*);*
    TreeReduction($sib(N)$, k, $precls$, $chi[\,]$*); }*
  }
}

5.3 Construction de Classifieurs

La deuxième étape de réduction est appliquée pendant la construction de classifieurs. Après la réduction utilisant $TreeReduction$, le jeu de données est lu à nouveau. Pour chaque objet, la fonction $Match$ cherche dans l'arbre les noeuds correspondants aux RCAs qui classifient correctement l'objet. Ces noeuds sont mis dans une liste temporaire nommée lnd (initialisée à vide pour chaque objet), dans l'ordre de précédence des règles correspondantes. Quand l'arbre est entièrement visité, les règles formées des noeuds de lnd sont rangées (Fonction $AddRule$) dans le classifieur, noté lrc, d'après leurs étiquettes de classes. Cependant, soient r une règle en cours de considération et rc la règle de lrc en cours d'être comparée avec r, si r et rc ont la même étiquette de classe et $LHS(rc) \subset LHS(r)$ et $conf(r) \leq conf(rc)$ alors r est rejetée.

La méthode pour classifier un objet suit le schéma de test de HARMONY : pour chaque étiquette de classe c, chercher dans lrc les règles qui couvrent l'objet. La somme de confiances de ces règles est calculée. L'objet est prédit de la classe dont la somme est maximale.

Function **BuildClassifier($\mathcal{D}$, p)** *{*
$lrc = \emptyset$ *;*
For each object (l : c) of the training dataset f do {
lnd = empty ; Match(p, l, c, lnd) ;
For each node N *of* lnd *do { build a CAR* $r(N)$ *with class label* c *;*
$lrc = AddRule(r(N), lrc)$ *; }*
}
return lrc *;*
}

6 Résultats Expérimentaux

L'approche proposée, appelée SIM (pour méthode SIMple), est implémentée en C (Linux version 9) et expérimentée sur un ordinateur portable doté d'un processeur mobile Pentium 4, 1.7 GHz et de 768 Mo de mémoire. Nous comparerons les résultats d'expérimentation avec ceux de HARMONY, une approche intéressante à laquelle SIM semble très similaire.

Le tableau 2 rappelle les caractéristiques des 10 jeux de données de *UCI* Coenen (2004) et les résultats d'expérimentation, sur les précisions de HARMONY et d'autres méthodes, rapportés dans Wang et Karypis (2005), utilisant le schéma de test "10-fold cross validation".

TAB. 2 – *Comparaisons de précision de HARMONY, rapport de Wang et Karypis 2005.*

Database	#objects	#items	#class	FOI	CPA	SVM	HARMONY
adult	48482	131	2	82,5	76,7	**84,2**	81,9
chess	28056	66	18	42,6	32,8	29,8	**44,9**
connect	67557	66	3	65,7	54,3	**72,5**	68,0
led7	3200	24	10	62,3	71,2	73,8	**74,6**
letRecog	20000	106	26	57,5	59,9	67,8	**76,8**
mushroom	8124	127	2	99,5	98,8	99,7	**99,9**
nursery	12960	32	5	91,3	78,5	91,3	**92,8**
pageBlocks	5473	55	5	**91,6**	76,2	91,2	**91,6**
penDigits	10992	90	10	88,0	83,0	93,2	**96,2**
waveform	5000	100	3	75,6	75,4	**83,2**	80,5
Moyenne				75,7	70,7	78,7	**80,7**

Maintenant nous présentons les résultats des expérimentations menées par ce travail (Tableau 3). Le programme exécutable de HARMONY est fourni gracieusement par les auteurs de Wang et Karypis (2005). Les paramètres sont configurés d'après la description par ces auteurs : $minsup = 50$ et les items sont ordonnés d'après l'ordre des coefficients de corrélation (l'ordre avec lequel HARMONY atteint la meilleure performance). En particulier, pour *connect*, seuls les items de supports < 20000 sont considérés. Les mêmes considérations sont appliquées pour SIM. La configuration des paramètres pour SIM :

(i) L'extraction des itemsets fréquents commence directement par les 2-itemsets, et tous k-itemsets considérés satisfont $k \leq 5$.

(ii) L'élagage des itemsets non fréquents est effectué seulement pour les i-itemsets pour $i \leq 2$, et avec $minsup = 50$. Les étapes suivantes sont développées sur les 2-itemsets fréquents, mais l'élagage n'est plus effectué, sauf pour les itemsets de support 1.

(iii) Les tests χ^2 sont effectués avec le risque d'erreurs de $0,5\%$.

Dans le tableau 3, on utilise les notations suivantes.

– Ts : le temps d'exécution total en secondes des dix exécutions du test "10-fold cross validation" par jeu de données, pour construire les dix classifieurs et pour les tester.

– Acc. : La précision moyenne de prédiction, en pourcentage.

– L : la longueur maximale des itemsets considérés pendant la construction de l'arbre.

– #Rules : le nombre moyen de règles dans les classifieurs.

Notons que dans les expérimentations de SIM, les objets sont lus toujours du disque. D'ailleurs, bien que l'on limite la longueur des itemsets en construction de l'arbre de préfixes, on ne limite pas la longueur des objets dans les jeux de données.

TAB. 3 – *HARMONY comparé à deux versions de SIM.*

	HARMONY		SIM sans élagage des clés non essentiels				SIM avec élagage des clés non essentiels		
Data	Ts	Acc.	L	Ts	Acc.	#Rules	Ts	Acc.	#Rules
adult	801	83,4	3	**183**	**84,23**	1676	**145**	**84,13**	1448
chess	**14**	44,9	5	70	**60,62**	11738	91	**58,86**	10727
connect	**181**	77,3	4	1142	**77,68**	19497	1133	**77,65**	19363
led7	**2**	**74,4**	5	3	74,37	232	**2**	74,22	196
letRecog	2100	**70,8**	4	**852**	**71,28**	8321	**823**	68,78	7644
mushroom	**9**	**100**	3	66	99,94	206	41	99,96	184
nursery	**8**	92,9	5	29	**98,33**	2108	29	**98,33**	2108
pageBlocks	8	**91,2**	4	8	90,93	12	**3**	90,69	7
penDigits	**244**	96,0	4	299	**96,99**	5194	297	**96,92**	5038
waveform	7048	77,9	3	**55**	**79,78**	1418	**56**	**79,30**	1360
Total/Moy.	10416	80,9		**2707**	**83,41**	5040	**2620**	**82,88**	4807

7 Discussions et conclusion

Les résultats obtenus par les tests menés dans ce travail (Tableaux 3) confirment les résultats rapportés dans Wang et Karypis (2005) (Tableau 2). Une remarque : pour *adult* et *connect*, le tableau 3 montre que la précision de HARMONY est meilleure que celle rapportée par ses auteurs. Cependant, pour *letRecog* et *waveform*, il est inverse.

Les comparaisons entre HARMONY et SIM :

– En temps d'exécution : pour les jeux de données des objets courts, SIM est en général plus lent. En particulier, pour *connect* SIM est environ six fois plus lent que HARMONY. Ceci s'explique par le fait que SIM lit toujours les données du disque et que le nombre de règles dans le classifieur est plus grand que celui de HARMONY. Cependant, pour les jeux de données des objets longs, HARMONY est en général beaucoup plus lent. Par exemple, pour

waveform HARMONY est environ 130 fois plus lent que SIM. La raison est que HARMONY peut considérer les itemsets de toute taille (pour *waveform*, la plupart des itemsets sont de tailles de 5 à 9, tandis que SIM ne considère que des itemsets de taille maximale 3).

– En précision de classification : les deux approches sont comparables, sauf pour les jeux de données *chess, nursery* et *waveform*, SIM est plus précis. Ceci peut s'expliquer par la sélection de RCAs de confiances et de supports maximaux parmi celles ayant de petits supports.

Bien que SIM soit similaire à HARMONY sur plusieurs points, les résultats expérimentaux des deux approches sont très différents. En général, SIM est meilleur en temps d'exécution et en précision de classification. Avec un temps d'exécution total environ quatre fois plus court, la prédiction par SIM est environ $2,5\%$ plus précise que celle de HARMONY, en moyenne.

Quand SIM implémente la suppression des itemsets clés non essentiels, en moyenne, le nombre de RCAs diminue d'environ 5%, le temps d'exécution est amélioré d'environ 3%, et la précision baisse de $0,53\%$. Cependant, par rapport à HARMONY, cette précision est encore de $1,98\%$ plus grande.

Ces résultats valident l'idée d'utilisation des RCAs construites sur les itemsets clés de petites tailles et la possibilité de sélection des RCAs de confiances et de supports maximaux parmi celles ayant de petits supports. D'ailleurs, ils montrent que la notion d'itemset clé non essentiel est applicable et utile. Pour perspective, on peut penser que cette notion peut être développée pour l'apprentissage dans le contexte d'existence de bruits.

Références

Agrawal, R., T. Imielinski, et A. Swami (1993). Mining association rules between sets of items in very large databases. In *ACM SIGMOD Conference Proceedings*, pp. 207–216.

Agrawal, R. et R. Srikant (1994). Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases*, pp. 487–499.

Bastide, Y., R. Taouil, N. Pasquier, G. Stumme, et L. Lakhal (2000). Mining frequent patterns with counting inferences. In *ACM SIGMOD Explorations*, pp. 66–75.

Bayardo, R. J. (1998). Efficiently mining long patterns from databases. In *ACM SIGMOD Conference Proceedings*, pp. 85–93.

Clark, P. et T. Niblett (1995). The CN2 induction algorithm. *Machine Learning 3*(4), 261–283.

Coenen, F. (2004). The lucs-kdd implementations of the FOIL, PRM, and CPAR algorithms. In *http ://www.csc.liv.ac.uk/˜frans/KDD/Software/FOIL_PRM_CPAR/foilPrmCpar.html, Computer Science Department, University of Liverpool, UK*.

Cohen, W. (1995). Fast effective rule induction. In *ICML'95*.

Duda, R. et P. Hart (1973). *Pattern Classification and Scene Analysis*. John Wiley & Sons.

Ganter, B. et R. Wille (1999). *Formal concept Analysis : Mathematical Foundations*. Springer.

Han, J., J. Pei, et Y. Yin (2000). Mining frequent patterns without candidate generation. In *ACM-SIGMOD International Conference (SIGMOD'00)*, pp. 1–12.

Lent, B., A. Swami, et J. Widom (1997). Clustering association rules. In *ICDE'97*, pp. 220–231.

Li, W., J. Han, et J. Pei (2001). CMAR : Accurate and efficient classification based on multiple class-association rules. In *ICDM'01*, pp. 369–376.

Lim, T. S., W. Y. Loh, et Y. S. Shih (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning 39*.

Liu, B., W. Hsu, et Y. Ma (1998). Integrating classification and association rule mining. In *KDD'98*, pp. 80–86.

Park, J., M. Chen, et P. Yu (1995). An effective hash-based algorithm for mining association rules. In *SIGMOD'95*, pp. 175–186.

Pasquier, N., Y. Bastide, R. Taouil, et L. Lakhal (1999). Discovering frequent closed itemsets for association rules. In *ICDT 1999, LNCS vol. 1540*, pp. 398–416. Springer-Verlag.

Pei, J., J. Han, et R. Mao (2000). Closet : An efficient algorithm for mining frequent closed itemsets. In *Proc. of Workshop on Research Issues on Data Mining and Knowledge Discovery (DMDK)*, pp. 21–30.

Phan-Luong, V. (2002). The closed keys base for frequent itemsets. In *DaWak'02*.

Quinlan, J. (1993). C4.5 : Programs for machine learning. In *Morgan Kaufmann*.

Quinlan, J. et R. Cameron-Jones (1993). FOIL : A midterm report. In *ECML'93*.

Rymon, R. (1992). Search through systematic set enumeration. In *International Conference on Principles of Knowledge Representation and Reasoning*.

Stumme, G., R. Taouil, Y. Bastide, N. Pasquier, et L. Lakhal (2002). Computing iceberg concept lattices with titanic. *Data Knowledge Engineering 42*(2), 189–222.

Wang, J. et G. Karypis (2005). HARMONY : Efficiently mining the best rules for classification. In *SIAM'05*.

Yahia, S. B., T. Hamrouni, et E. M. Nguifo (2006). Frequent closed itemset based algorithmes : A thorough structural and analytical survey. *ACM-SIGKDD Explorations 8*(1), 93–104.

Yin, X. et J. Han (2003). CPAR : Classification based on predictive association rules. In *SDM'03*, San Francisco, CA, pp. 369–376.

Zaki, M. J. (2000). Generating non-redundant association rules. In *ACM SIGMOD KDD'00*, pp. 33–43.

Zaki, M. J., S. Parthasarathy, M. Ogihara, et W. Li (1997). New algorithms for fast discovery of association rules. In *KDD'97*, pp. 283–286.

Summary

In classification based on class-association rules, key itemsets (minimal generators) are essential in the built classifiers: non key itemsets can be pruned without affecting the accuracy of the classifiers. This work studies the generalization of a property of non key itemsets and shows that among the small size key itemsets, there still exist those which are not significant to the built classifiers, and can also be pruned. Those key itemsets are defined based on a χ^2 test. We apply this pruning to a method for building classifiers based on class-association rules, using a prefix tree structure for mining the frequent itemsets. Experiences on large datasets show that the pruning method is actually efficient and sound.