Un algorithme de génération des itemsets fermés pour la fouille de données

Huaiguo Fu, Engelbert Mephu Nguifo

CRIL-CNRS FRE2499, Université d'Artois-IUT de Lens
Rue de l'université SP 16, 62307 Lens cedex. France
{fu,mephu}@cril.univ-artois.fr

Résumé. Le traitement de grand volume de données est un problème pour l'extraction de connaissances. La fouille de données nécessite des méthodes de résolution efficaces. Le treillis de concepts (treillis de Galois) est un outil utile pour l'analyse de données. Des travaux en classification et sur les règles d'association ont permis d'accroître son intérêt. Plusieurs algorithmes de génération ont été proposés, parmi lesquels NextClosure est l'un des meilleurs pour traiter des données de grande taille. Mais la complexité de NextClosure reste malgré tout très élévé. Aussi nous proposons un nouvel algorithme efficace nommé **ScalingNextClosure**, et basé sur une méthode de partitionnement de données pour générer de manière indépendante les itemsets fermés de chaque partition. Les résultats expérimentaux montrent que cette technique de partitionnement améliore efficacement NextClosure.

1 Introduction

Dans le domaine des bases de données et de l'analyse de données, les données de grande taille reste difficile à analyser. Par exemple, dans le cas de la fouille de règles d'association [Agrawal *et al.*, 1996], trouver tous les itemsets fréquents est un problème de complexité exponentielle par rapport au nombre d'items. La génération des itemsets fréquents est l'étape la plus importante dans le processus de recherche des règles d'association. Il est donc nécessaire de développer des algorithmes efficaces pour traiter cette étape. La structure de treillis de concepts est un outil intéressant pour la génération des itemsets fermés fréquents qui est l'étape la plus importante dans la recherche de règles d'association. Ses fondements théoriques repose sur la théorie mathématique des treillis [Birkhoff, 1967]. Elle est constituée de concepts formels générés à partir d'un contexte de données pour montrer les relations entre les objets et les attributs de ce contexte. Le treillis des itemsets fermés est inclus dans le treillis des concepts. Le problème de génération des itemsets fréquents peut être réduit à la génération des itemsets fermés fréquents avec le treillis des itemsets fermés. Et il est possible d'élaguer le nombre de règles générées sans perte d'information à partir du treillis des itemsets fermés. Un itemset fermé est l'intension d'un concept formel, et est un itemset maximal pour la recherche des règles d'association [Pasquier *et al.*, 1999].

Plusieurs algorithmes ont été proposés pour générer les concepts formels et/ou le treillis de concepts formels. Les comparaisons expérimentales des temps de calcul de ces algorithmes montrent que l'algorithme Nextclosure [Ganter et Wille, 1999] est

l'un des meilleurs pour les contextes denses et volumineux [Mephu Nguifo *et al.*, 2002, Fu et E., 2004]. Néanmoins le temps de génération reste très élevé pour ces cas. Ici, nous proposons un nouvel algorithme, **ScalingNextClosure**, pour générer les concepts formels et les itemsets fermés en présence de données de grande taille. Cet algorithme s'appuie sur une méthode de partitionnement de données différente de celles existantes. Notre algorithme utilise une méthode permettant de décomposer l'espace de recherche en plusieurs partitions, à condition que dans chaque partition il existe au moins un itemset fermé.

Nous avons testé notre algorithme sur des données de la base du UCI et dans le pire des cas. Les comparaisons expérimentales avec NextClosure, montre que notre algorithme séquentiel, ScalingNextClosure, est meilleur sur les données volumineuses. En outre nous avons réussi à générer des itemsets fermés dans des situations de pire des cas qui étaient impossible de traiter avec d'autres algorithmes en notre possession. Par exemple, avec des données de plus de 20 items, NextClosure n'a pas réussi à générer les itemsets fermés sur un ordinateur PC PIII450 avec 128Mo de RAM.

Le reste du papier est organisé comme suit : les notions de base du treillis des itemsets fermés sont décrites dans la section suivante. L'algorithme ScalingNextClosure est présenté dans la section 3. La section 4 décrit les comparaisons expérimentales.

2 Treillis des itemsets fermés

Définition 2.1 *Un* **contexte** *est un triplet (G, M, R), où G (resp. M) un ensemble d'objets (resp. d'items ou attributs), et R est une relation entre G et M.*

Définition 2.2 *Soient $A \subseteq G$ et $B \subseteq M$, on définit l'opérateur ' comme engendrant l'ensemble A' (resp. B') des items (resp. objets) communs à tous les objets de A (resp. tous les items de B).*

$A' := \{m \in M \mid gRm$ pour tout $g \in A\}$. $B' := \{g \in G \mid gRm$ pour tout $m \in B\}$.

Ces deux opérateurs constituent la **correspondance de Galois** *de (G, M, R). On a les propriétés suivantes : $B \subseteq M \Rightarrow B'' \subseteq M$; et $A \subseteq G \Rightarrow A'' \subseteq G$.*

Nous avons 2 **opérateurs de fermeture** *: $B \rightarrow B''$ sur M et $A \rightarrow A''$ sur G.*

Définition 2.3 *Un* **concept formel** *de (G, M, R) est un couple (A, B) avec $A \subseteq G$, $B \subseteq M, A = B'$ et $B = A'$. A est l'extension, et B est l'intension.*

Définition 2.4 *Un itemset $C \subseteq M$ est un* **itemset fermé** *si et seulemnt si $C'' = C$.*

Tous les itemsets fermés muni de la relation d'ordre partiel, constituent un treillis complet appelé **treillis des itemsets fermés**.

3 L'algorithme ScalingNextClosure

Définition 3.1 *Un contexte est dit* **ordonné** *si ses items sont rangés suivant leur nombre d'objets de chaque item, en partant du plus petit au plus grand, et les items vérifiés par les mêmes objets sont fusionnés pour constituer un seul item.*

La fusion des items ayant les mêmes objets est une précondition pour que ScalingNextClosure génère correctement les itemsets fermés. Le treillis des itemsets générés à partir du contexte ordonné est identique au treillis des itemsets générés à partir du contexte initial. Le contexte ordonné peut être engendré en un temps polynomial, qui de surcroît est infiniment petit par rapport au temps de génération du treillis.

3.1 L'espace de recherche des itemsets fermés

En effet chaque itemset fermé est un sous-ensemble de M, aussi tous les sous-ensembles de M sont des éléments de l'espace de recherche. Ainsi la taille de l'espace de recherche pour l'énumération de tous les itemsets fermés est 2^m. Cette espace de recherche est la couverture de sous-ensembles d'items de M. Nous considérons que l'espace de recherche est la couverture des sous-ensembles que nous nommons $fold$, où $fold_i$ contient tous les sous-ensembles de $\{a_i, \ldots, a_{m-3}, a_{m-2}, a_{m-1}, a_m\}$ qui inclut a_i. Chaque $fold$ est un sous-espace de recherche. L'espace de recherche entier sera décomposé en fonction des éléments de génération de chaque $fold$.

Définition 3.2 *Soit un item $a_i \in M$, l'ensemble F_{a_i} est appelé* **ensemble partiel** *de a_i, et défini par : $F_{a_i} := \{a_j \in M |$ pour tout $a_j \in M, i < j \leq m\}$*

En d'autres termes, l'ensemble partiel de a_i est l'ensemble $\{a_{i+1}, a_{i+2}, \ldots, a_{m-1}, a_m\}$.

Définition 3.3 *Un item est joint à chaque sous-ensemble de son ensemble partiel pour générer les nouveaux itemsets. Ceux-ci forment un sous-espace de recherche appelé* **sous-espace de recherche partiel d'un item** (*que nous notons* **F3S**).

Par exemple, F3S de a_{m-1} est : $\{a_{m-1}\}$; $\{a_{m-1}, a_m\}$.

Nous fusionons les items ayant exactement les mêmes objets, de manière à assurer qu'il existe au moins un itemset fermé dans le sous-espace de recherche partiel de chaque item. Ceci est une précondition matérialisée par la propriété suivante.

Propriété 3.1 *Etant donné un contexte ordonné, il existe des itemsets fermés dans le sous-espace de recherche partiel d'un item.*

Preuve : Soit le contexte ordonné (G, M, R), $\forall a_i \in M$ et $a_j \in M (1 \leq j < i)$, nous avons $\{a_i\}' \not\subseteq \{a_j\}'$, ainsi $\{a_i\}''$ est dans le sous-espace de recherche partiel de l'item a_i, sinon $\exists a_j (1 \leq j < i), \{a_i\}' \subseteq \{a_j\}'$.

3.2 Un algorithme de décomposition pour de grands contextes

Nous choisissons les items du contexte ordonné pour former un ensemble ordonné P. Si le nombre d'éléments de P est T, nous avons $a_{P_1} < a_{P_2} < \ldots < a_{P_k} < \ldots < a_{P_T}$. Nous notons $[a_{P_k}, a_{P_{k+1}}]$ l'espace de recherche de l'item a_{P_k} à l'item $a_{P_{k+1}}$. A partir de $\{a_{P_k}\}$ ($\{a_{P_k}\}$ est le premier sous-ensemble de $[a_{P_k}, a_{P_{k+1}}]$), nous générons les prochains itemsets fermés jusqu'à $\{a_{P_{k+1}}\}$, ainsi nous pouvons engendrer tous les itemsets fermés dans l'intervalle $[a_{P_k}, a_{P_{k+1}}]$. Tous les itemsets fermés du contexte sont inclus dans :

$$\bigcup_{1<k<T} [a_{P_k}, a_{P_{k+1}}] \cup [a_{P_T})$$

ScalingNextClosure décompose l'espace de recherche, et génère tous les itemsets fermé de chaque sous-espace de recherche. Pour chaque partition, nous utilisons NextClosure pour parcourir le sous-espace de recherche et générer les itemsets fermés. ScalingNextClosure comprend 2 étapes : déterminer les partitions (algorithme 1) et générer tous les itemsets fermés de chaque partition (algorithme 2).

Le paramètre DP permet de déterminer le début et la fin de chaque partition. Sa valeur est comprise entre 0 et 1 ($0 < DP < 1$).

Algorithm 1 Première étape de ScalingNextClosure : determiner les partitions

saisir un paramètre DP ($0 < DP < 1$)
générer le contexte ordonné (engendrer la liste ordonnée des items)
retourner l'ordre des items du contexte ordonné
m = cardinal de l'ensemble des items du contexte ordonné
$min := m$
$k := 0$
while ($min >= 1$) **do** {determiner les partitions}
  $k = k + 1$
  $P_k := min$
  retourner P_k
  $min := int(min * DP)$
end while
$T := k$ //T est le nombre de partitions

Algorithm 2 ScalingNextClosure : génération des itemsets fermés pour chaque partition

entrer l'ordre des items du contexte ordonné
entrer P_k et P_{k+1} //entrée de partition
$A \leftarrow \{a_{P_k}\}$
$END \leftarrow a_{P_{k+1}}$
$stop := false$
while (!$stop$) **do**
  $A \leftarrow$ générer la prochaine fermeture de A pour le contexte ordonné : NextClosure
  if $END \in A$ durant la génération de la prochaine fermeture **then**
    $stop := true$
  end if
end while

Nous utilisons tous les items P_k du contexte ordonné pour construire les partitions $[a_{P_k}, a_{P_{k+1}}]$ et $[a_{P_T})$, avec $1 \leq k \leq T$. P_k est la position d'un item dans le contexte ordonné, et représente l'item a_{P_k} ; a_{P_k} correspond à un ou plusieurs items quelconques du contexte initial. Quand nous recherchons les itemsets fermés, Ø n'est pas pris en considération.

4 Expérimentations

Nous avons implémenté une version séquentielle de notre algorithme en JAVA. Les résultats de notre implémentation séquentielle sur un ordinateur PIII450 avec 128Mo

de RAM montre un gain en temps de calcul d'un facteur 10 par rapport à NextClosure. Par exemple avec l'ensemble Agaricus (8124 objets et 124 items), NextClosure a un temps total supérieur 60 fois à celui de ScalingNextClosure.

Nous présentons les résultats (voir figure 1) sur 11 ensembles de données de la base UCI, choisis en fonction du nombre élévé de concepts (Les 11 contextes ($|G|, |M|$) : 1 : soybean-small (47,79), 2 : SPECT (187,23), 3 : flare2 (1066,32), 4 : audiology(26,110), 5 : breast (699,110), 6 : lung-cancer (32,228), 7 : promoters (106,228), 8 : soybean-large (307,133), 9 : dermatology (366,130), 10 : nursery (12960,31), 11 : agaricus (8124,124)). En effet sur des données de petite taille, ScalingNextClosure n'est pas nécessairement utile. La valeur par défaut du paramètre DP est fixée à 0.5.

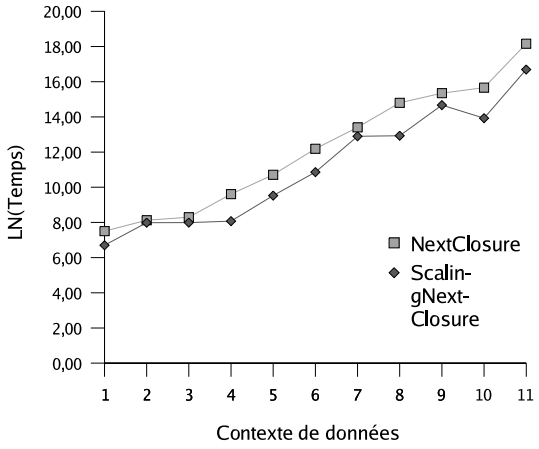

FIG. 1 – Temps de calcul en LN(Temps), en millisecondes.

Pour nos expérimentations, nous avons utilisé différentes valeurs de DP pour tester l'effet de ce paramètre sur notre algorithme. En effet la taille des partitions varie en fonction de ce paramètre. Plus DP est élévé, plus il y a de partitions. Le temps de calcul est inversément proportionnel à DP dans le cas de données à forte densité, comme l'ensemble Agaricus. Pour les autres données, la variation du temps de calcul reste très faible, même si elle est toujours légèrement décroissante si on augmente le nombre de partitions. Le choix de DP peut être variable d'un contexte à l'autre.

5 Conclusion

Dans ce papier nous avons présenté un nouvel algorithme de génération d'itemsets fermés, ScalingNextClosure, pour traiter des données volumineuses. Cet algorithme s'appuie sur une méthode de décomposition de l'espace de recherche et sur l'algorithme NextClosure pour générer les itemsets fermés. Les données sont partitionnées suivant un ordre d'ordonnancement des items et d'un paramètre fixé par l'utilisateur.

Les résultats expérimentaux obtenus montrent un gain d'un facteur supérieur en

moyenne de 10 de ScalingNextClosure par rapport à NextClosure sur des implémentations séquentielles. En outre ScalingNextClosure permet de traiter des données dans le pire des cas, que l'on ne pouvait traiter avec les autres implémentations d'algorithmes.

ScalingNextClosure utilise un paramètre DP pour créer les partitions. L'optimisation du nombre de partitions pour un contexte donné est nécessaire.

Remerciements

Nous remerçions les relecteurs anonymes pour leurs remarques constructives. Ce travail est partiellement financé par l'IUT de Lens, l'université d'Artois et la région Nord-Pas-de-Calais.

Références

[Agrawal *et al.*, 1996] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, et A.I. Verkamo. *Advances in Knowledge discovery and Data Mining*, chapter Fast discovery of association rules, pages 307–328. AAAI/MIT Press, 1996.

[Birkhoff, 1967] G. Birkhoff. *Lattice Theory.* American Mathematical Society, Providence, RI, 3rd edition, 1967.

[Fu et E., 2004] Huaiguo Fu et Mephu Nguifo E. How well go lattice algorithms on currently used machine learning testbeds ? In *Proc. EGC*, 2004.

[Ganter et Wille, 1999] B. Ganter et R. Wille. *Formal Concept Analysis. Mathematical Foundations.* Springer, 1999.

[Mephu Nguifo *et al.*, 2002] E. Mephu Nguifo, M. Liquiere, et V. Duquenne. *JETAI Special Issue 14(2/3).* Taylor and Francis, 2002.

[Pasquier *et al.*, 1999] N. Pasquier, Y. Bastide, R. Taouil, et L. Lakhal. Efficient mining of association rules using closed itemsets lattices. *Journal of Information Systems*, 24(1) :25–46, 1999.

Summary

Large data still bothers us in the field of machine learning and data mining. The mining of very large database still needs more efficient algorithm. Concept lattice is an effective tool for data analysis and knowledge discovery. Many research works in classification or association rules increase the interest of Concept lattices structures for data mining. Several algorithms were proposed to generate concepts of lattices, among which NextClosure algorithm is one of the bests for large context. However, it's also hard to face the complexity of large data with NextClosure algorithm. So we propose a new efficient scalable algorithm : **ScalingNextClosure** which creates partitions within data and builds independently closed itemsets in each partition. There is no other communication among each partition, so that **ScalingNextClosure** algorithm can be easily parallelized. The experimental results show that the algorithm has efficient performance compared to NextClosure algorithm.