Réduction du coût d'évaluation d'une règle relationnelle

Agnès Braud*, Teddy Turmeaux**

*CENTRIA, FCT/UNL, 2829-516 Caparica, Portugal
braud@fct.unl.pt
**LIFO, Université d'Orléans, rue Léonard de Vinci, BP 6759,
F-45067 Orléans Cedex 2, France
Teddy.Turmeaux@lifo.univ-orleans.fr

Résumé. De nombreuses tâches en Fouille de Données visent à extraire des connaissances exprimées sous la forme d'un ensemble de règles. Les algorithmes dédiés à ces tâches engendrent des règles dont l'adéquation aux données doit être évaluée. On se place dans le cadre où cette évaluation est réalisée directement en lançant des requêtes de dénombrement sur la base de données, et où cette base est relationnelle. Les requêtes comptent les données qui s'apparient avec la règle, calcul qui peut être extrêmement coûteux. Dans cet article, nous étudions l'impact d'une approche d'échantillonnage visant à réduire le coût de l'évaluation des règles relationnelles en tenant compte des spécificités structurelles des requêtes induites.

1 Introduction

Nous nous intéressons à l'apprentissage/extraction de règles basées sur un formalisme du premier ordre, dont l'utilisation en Apprentissage est étudiée dans le cadre de la Programmation Logique Inductive (PLI) [Muggleton et Raedt, 1994]. Dans ce contexte, un des problèmes principaux provient du coût du test d'appariement d'une clause avec les données qui permet de mesurer l'accord aux données. En Fouille de Données (FD), le premier ordre permet de traiter les bases de données relationnelles sans les aplatir et donc sans perdre d'informations sur la structure. Cependant, le problème de coût est alors encore plus critique étant donné les volumes de données à traiter, et les systèmes de PLI rencontrent des difficultés pour passer à l'échelle.

La plupart des systèmes de PLI procèdent à une reformulation des données sous la forme utilisée habituellement par le système (en général des clauses Prolog). Une fois les données reformulées dans le formalisme adéquat, le système peut être appliqué sans modification. Nous voyons cependant plusieurs avantages au fait de laisser la charge de l'évaluation au SGBD. Premièrement, il n'est pas nécessaire de passer par une représentation intermédiaire pour traiter les données. Deuxièmement, les SGBD sont sans cesse améliorés et disposent de techniques pour accélérer les accès aux données (index, plans d'exécution par exemple). Ensuite, les SGBD restent la seule solution viable pour traiter des volumes de données importants. Enfin, cela permet d'envisager l'intégration de modules de FD au sein même des SGBD existants.

Nous nous plaçons donc dans ce cadre et nous étudions une méthode visant à accélérer l'éxecution par un SGBD des requêtes permettant d'évaluer les règles relationnelles engendrées par les systèmes. Différentes optimisations ont déjà été proposées, à la fois dans le domaine

des bases de données et dans celui de l'apprentissage. On peut distinguer différents types d'optimisation :

- réduction de la règle. On entend par cela des modifications de la règle qui engendrent une règle équivalente mais plus simple à évaluer (minimisation, simplification sémantique et décomposition en plusieurs requêtes),

- réduction des données. Il s'agit de construire et travailler sur un synopsis des données,

- traitement des règles par lots.

Dans cet article nous nous concentrons sur la réduction des données et plus particulièrement l'échantillonnage. Nous commençons par formaliser l'énoncé du problème d'évaluation d'une règle. Dans la section 3, nous proposons une technique d'échantillonnage visant à accélérer l'exécution d'une requête d'évaluation de la règle. La section 4 discute des résultats obtenus lors des expérimentations.

2 Problématique

Le problème qui nous intéresse est la réduction du coût du comptage des données qui s'apparient avec une règle r lorsque ces données sont stockées dans une base de données relationnelle et que le comptage est réalisé en envoyant des requêtes au SGBD. Dans ce qui suit, nous parlons donc non pas de clause mais de requête.

Nous travaillons sur des requêtes basées sur une restriction du SQL standard et appelées requêtes conjonctives [Chandra et Merlin, 1977].

Ce sont des formules du premier ordre de la forme $\exists\tilde{y}(p_1(\tilde{x_1},\tilde{y_1}) \wedge \ldots \wedge p_n(\tilde{x_n},\tilde{y_n}))$ avec $\forall i\ \tilde{x_i} \cap \tilde{y} = \emptyset$ et $\tilde{y} = \overset{i}{\cup}\ \tilde{y_i}$, que l'on notera (en datalog) $q(\tilde{x}) : -p_1(\tilde{x_1},\tilde{y_1}), \ldots, p_n(\tilde{x_n},\tilde{y_n})$ avec $\tilde{x} = \overset{i}{\cup}\ \tilde{x_i}$. $\tilde{x}$ et $\tilde{y}$ sont des ensembles de variables, p_i sont des prédicats et q correspond à la projection du résultat sur les variables $\tilde{x}$.

Notre problème peut alors être défini comme suit. Etant donné :

- une base de données relationnelle $\mathcal{D}$,

- une règle r,

- une relation $label$ qui établit un lien entre la règle r et une entité d'intérêt (un objet, un exemple...). C'est cette relation qui permet de compter les données qui s'accordent avec r.

On cherche à qualifier l'intérêt de r pour le problème, en comptant le nombre d'éléments de la relation $label$ vérifiant les critères spécifiés par r. Ceci est réalisé en évaluant une ou plusieurs requêtes $q_i(\tilde{x}) : -label(\tilde{x}),C_i(\tilde{x},\tilde{y})$ sur $\mathcal{D}$, où $C_i(\tilde{x},\tilde{y})$ est une conjonction de prédicats créée à partir de r ($C_i = f_i(r)$) et $\tilde{x} \cap \tilde{y} = \emptyset$.

Par exemple, pour un problème de classification, $label$ est la relation qui étiquette les exemples : le nombre d'exemples couverts par une hypothèse comme $attribut > 10$ sera déterminé par une requête telle que
$q(idExemple) : -exemple(idExemple,attribut),exemple\_positif(idExemple)$,
$attribut > 10$. Dans cet exemple simple, la relation $exemple\_positif$ est la relation $label$ car on veut compter les exemples positfs.

Ce type de requêtes et le problème de calcul de la taille de son résultat est présent dans divers cadres d'apprentissage de règles relationnelles ou de FD, comme dans la recherche de requêtes fréquentes [Dehaspe, 1998], l'induction de clauses de Horn [Muggleton et Raedt, 1994],

la découverte de requêtes étendues fréquentes [Dehaspe et Raedt, 1997], ou encore pour l'induction d'arbres de régression ou de classification du premier ordre [Kramer, 1996], [Blockeel et Raedt, 1998].

Les requêtes que nous considérons présentent un certain nombre de caractéristiques :

1. elles sont engendrées automatiquement par un système,
2. elles ont pour but l'évaluation de connaissances engendrées lors d'un processus de FD,
3. ce sont des requêtes de dénombrement,
4. elles sont composées :
   - d'une seule occurrence de $label$, relation sur laquelle s'effectue le dénombrement,
   - de jointures spécifiant des égalités entre attributs (dans la suite chaque fois qu'on parlera de jointure, il s'agira de l'équi-jointure),
   - de sélections traduisant des conditions sur les valeurs des attributs,
5. les contraintes liées au schéma de la base et les biais d'apprentissage utilisés par les systèmes font qu'il existe une combinaison de jointures qui inclut toutes les combinaisons apparaissant dans les requêtes.

A partir de ces caractéristiques nous pouvons faire plusieurs observations qui donnent des pistes pour accélérer leur exécution. Une première est que les mécanismes mis en œuvre pour engendrer automatiquement de nouvelles règles peuvent conduire à l'ajout de prédicats redondants ou non pertinents dans ces règles. Ceci justifie l'intérêt des techniques de réduction de requêtes. Une seconde observation découle de l'utilisation de ces requêtes en FD. En effet, on tolère une marge d'imprécision, notamment au début du processus qui correspond plus à une phase exploratoire durant laquelle on cherche simplement à identifier des zones intéressantes de l'espace de recherche. Ceci permet d'utiliser des techniques de réduction des données.

3 Approche étudiée

L'idée de la réduction de données est de créer un résumé des données et de travailler sur cette forme réduite. Contrairement à la réduction de requête, ces optimisations ne conduisent pas à des résultats exacts et la précision peut alors être affectée. Ce sacrifice vaut cependant la peine s'il n'est pas exagérément important et si le gain de temps en retour est significatif.

Plusieurs techniques sont décrites dans [Barbará *et al.*, 1997] : ondelettes (*wavelets*), histogrammes, échantillonnage, Décomposition en Valeurs Singulières (DVS), régression, modèles log-linéaires, regroupement et arbres d'indexage. Nous avons considéré les trois premières.

Les histogrammes et les ondelettes rencontrant des difficultés dans le cas du traitement d'attributs non ordonnés, nous nous sommes concentrés sur l'échantillonnage. C'est également une technique d'une utilisation plus souple, qui peut donc être plus facilement adaptée pour pallier ses difficultés, et qui a été étudiée dans de nombreux travaux prouvant son intérêt et la dotant de bases théoriques.

L'échantillonnage d'une relation R (resp. requête Q) retourne un sous-ensemble des tuples de cette relation (du résultat de cette requête).

On essaiera souvent de faire un *échantillonnage aléatoire uniforme* ou *aléatoire simple* pour lequel chaque élément de l'ensemble de base a même probabilité d'inclusion dans l'échantillon. Il ne requiert aucune information sur les données et présente également l'avantage d'être l'objet de nombreux résultats théoriques.

Nous détaillons ci-dessous les choix faits dans notre approche, qui sont basés sur les spécificités de notre cadre applicatif et des requêtes engendrées.

L'évaluation du résultat d'une requête de dénombrement par échantillonnage comprend trois étapes :

1. engendrer un échantillon des tuples vérifiant les critères de la requête,

2. dénombrer les entités d'intérêt (dans notre cas, le nombre d'entités distinctes n de $label$),

3. extrapoler le résultat de la requête non échantillonnée à partir de n.

Les deux étapes délicates sont la première et la dernière. Pour la première il faut parvenir à engendrer un échantillon représentatif. On peut procéder de deux manières :

- échantillonner la relation $label$, puis exécuter les requêtes sur l'échantillon,

- échantillonner le résultat de la requête.

Pour l'extrapolation, on utilise des fonctions appelées *estimateurs* (voir [Haas *et al.*, 1995] pour une description d'estimateurs courants).

Nous allons maintenant étudier ces deux étapes.

Echantillonnage d'une requête. La mise en place du premier type d'échantillonnage est assez simple mais n'est pas satisfaisante dans le cas où la requête comprend des jointures. En effet, on ne contrôle pas la taille de l'échantillon résultant car le processus d'échantillonnage est réalisé tout au début et l'on peut être confronté à deux configurations défavorables :

- peu de valeurs de $label$ s'apparient avec beaucoup d'éléments de la relation avec laquelle il y a jointure, on peut alors obtenir un résultat de taille trop faible si les valeurs qui s'apparient ne sont pas dans l'échantillon ;

- la taille du résultat reste trop importante pour être traitée.

Nous avons donc opté pour la deuxième solution, l'échantillonnage du résultat de la requête. Pour obtenir un échantillonnage aléatoire uniforme d'une jointure, il faut donner à chaque tuple de la jointure de base une même probabilité d'inclusion dans l'échantillon. Malheureusement, le type de requêtes que nous souhaitons traiter combine deux difficultés majeures pour réaliser l'échantillonnage de leur résultat [Chaudhuri et Motwani, 1999] :

- la jointure de deux échantillons uniformes n'est pas un échantillon uniforme de la jointure [Chaudhuri *et al.*, 1999] ;

- il n'existe pas de bon estimateur (i.e. avec erreur garantie) du nombre de valeurs distinctes d'un attribut d'une relation à partir d'un échantillonnage aléatoire uniforme, à moins d'examiner une large portion des données [Chaudhuri *et al.*, 1998][Charikar *et al.*, 2000].

Pour pallier ces problèmes, nous proposons d'adopter l'approche suivante, en deux étapes :

- échantillonnage (non obligatoirement uniforme) de la partie jointures de la requête et stockage de l'échantillon obtenu,

- application de la partie sélections de la requête sur l'échantillon obtenu à l'étape 1.

Nous exploitons alors le fait qu'il existe un nombre limité de combinaisons de jointures pour nos requêtes, et calculons un échantillon pour chacune de ces configurations. Ainsi lors de l'évaluation d'une requête, on considère l'échantillon adéquat pour la partie jointures de la requête. Ceci permet d'éviter le coût des entrées-sorties liées à la création de l'échantillon à chaque exécution de requête. Le nombre limité de combinaisons de jointures rend cette hypothèse réaliste.

Il est inutile et coûteux de calculer complètement l'équi-jointure pour en obtenir un échantillon. Nous utilisons notre propre variante de l'algorithme de [Olken, 1993] pour obtenir un

échantillonnage uniforme ou biaisé d'une équi-jointure sans la calculer entiérement.

Estimation du résultat. Nous présentons ici deux estimateurs courants, l'un n'utilise que les informations présentes dans l'échantillon alors que le second requiert une information sur la requête non échantillonnée. L'utilisation d'informations supplémentaires permet d'améliorer l'estimation, mais complique souvent le calcul.

Notons f (resp. f_i) le nombre d'éléments de $label$ présents (resp. apparaissant i fois) dans l'échantillon, t le nombre de $label$ présents dans la partie équi-jointure de la requête, f' le nombre de $label$ de l'échantillon qui vérifient les prédicats *ad hoc*.

L'estimateur de Chao n'utilise que les informations données par l'échantillon

$$\hat{D}_{Chao} = f + \frac{f_1^2}{2(f_2+1)} - \frac{f_1 f_2}{2(f_2+1)^2}.$$

L'estimateur Ratio requiert plus d'informations : $t' = t\frac{f'}{f}$.

4 Résultats et discussion

Nous avons appliqué notre technique d'échantillonnage sur des requêtes lançées sur la base Mutagénèse [King et Srinivasan, 1995] qui est classique en PLI et stocke des informations relatives à des objets de type molécule. On fait varier la taille de l'échantillon (1%,2%,3%,5%,10% et 20%). Nous obtenons évidemment des gains en temps très significatifs.

Nous réalisons des tests sur des échantillons aléatoires uniformes, mais nous proposons de plus des échantillons que nous appelons biaisés sur la relation $label$. Il s'agit pour ces échantillons d'éviter une sous-représentation de certains labels qui sont moins présents dans la jointure. Pour cela nous choisissons un label aléatoirement avant de choisir aléatoirement un tuple comprenant ce label.

Les tests nous permettent de vérifier le fait que la sélectivité de la requête influence beaucoup les résultats. En effet plus la requête est sélective, plus il est difficile d'obtenir une estimation correcte, notamment pour les petits échantillons. Nous constatons aussi que le fait de biaiser l'échantillonnage sur la relation $label$ améliore le résultat. Enfin on constate l'intérêt de l'échantillonnage, dans la mesure où l'on obtient des résultats d'une bonne précision pour des gains en temps importants.

Pour obtenir des résultats d'une grande précision avec une probabilité faible, on doit collecter plus d'informations. On peut par exemple calculer entièrement l'équi-jointure (sans la stocker) pour en extraire un échantillon non pas uniforme ou biaisé mais un échantillon distinct (*distinct sampling* [Gibbons, 2001]) plus représentatif qu'un échantillon uniforme. On adaptera la taille de l'échantillon à la précision voulue et à la probabilité souhaitée d'atteindre cette précision (dans certaines limites bien entendu).

Références

[Barbará *et al.*, 1997] Daniel Barbará, William DuMouchel, Christos Faloutsos, Peter J. Haas, Joseph M. Hellerstein, Yannis E. Ioannidis, H. V. Jagadish, Theodore Johnson, Raymond T. Ng, Viswanath Poosala, Kenneth A. Ross, et Kenneth C. Sevcik. The New Jersey data reduction report. *IEEE Data Engineering Bulletin*, 20(4):3–45, December 1997.

[Blockeel et Raedt, 1998] Hendrik Blockeel et Luc De Raedt. Top-down induction of first order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, June 1998.

[Chandra et Merlin, 1977] Ashok K. Chandra et Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *9th ACM Symposium on Theory of Computing*, pages 77–90, 1977.

[Charikar *et al.*, 2000] Moses Charikar, Surajit Chaudhuri, Rajeev Motwani, et Vivek R. Narasayya. Towards estimation error guarantees for distinct values. In *Proc. of PODS-00*, pages 268–279. ACM Press, May 15–17 2000.

[Chaudhuri *et al.*, 1998] Surajit Chaudhuri, Rajeev Motwani, et Vivek Narasayya. Random sampling for histogram construction: How much is enough? *SIGMOD Record*, 27(2):436–447, 1998.

[Chaudhuri *et al.*, 1999] Surajit Chaudhuri, Rajeev Motwani, et Vivek Narasayya. On random sampling over joins. In *Proc. of ACM SIGMOD*, pages 263–274. ACM Press, 1999.

[Chaudhuri et Motwani, 1999] Surajit Chaudhuri et Rajeev Motwani. On sampling and relational operators. *IEEE Data Engineering Bulletin*, 22(4):41–46, 1999.

[Dehaspe et Raedt, 1997] Luc Dehaspe et Luc De Raedt. Mining association rules in multiple relations. In *Proc. of ILP, LNAI 1297*, pages 125–132. Springer, 1997.

[Dehaspe, 1998] Luc Dehaspe. *Frequent Pattern Discovery in First-Order Logic*. PhD thesis, 1998.

[Gibbons, 2001] Phillip B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *VLDB 2001, Italy*, pages 541–550, 2001.

[Haas *et al.*, 1995] Peter J. Haas, Jeffrey F. Naughton, S. Seshadri, et Lynne Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *Proc. of VLDB'95*, pages 311–322, 11–15 September 1995.

[King et Srinivasan, 1995] R. D. King et A. Srinivasan. Relating chemical activity to structure: An examination of ILP successes. *New Generation Computing, Special issue on ILP*, 13(3-4):411–434, 1995.

[Kramer, 1996] S. Kramer. Structural regression trees. In *Proc. of AAAI-96*, pages 812–819, Cambridge/Menlo Park, 1996. AAAI Press/MIT Press.

[Muggleton et Raedt, 1994] Stephen Muggleton et Luc De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[Olken, 1993] Frank Olken. *Random Sampling from Databases*. PhD thesis, April 1993.

Summary

Many tasks in Data Mining aim at extracting knowledge expressed as a set of rules. The algorithms designed for those tasks generate rules for which one has to evaluate how much they fit the data. In our framework, this evaluation is realized by sending counting queries directly to the database and the database is relational. The queries count the data that match the rule, which can be computationaly very expensive. In this paper, we study the impact of an approach for sampling, aiming at decreasing the cost of the evaluation of relational rules, taking into account the structural specificities of the queries induced.