Mise en œuvre des méthodes de fouille de données spatiales Alternatives et performances

Nadjim Chelghoum, Karine Zeitouni

Laboratoire PRISM, Université de Versailles
45, avenue des Etats-Unis, 78035 Versailles Cedex, France
Nadjim.Chelghoum@prism.uvsq.fr, Karine.Zeitouni@prism.uvsq.fr

Résumé. La fouille de données spatiales nécessite l'analyse des interactions dans l'espace. Ces interactions peuvent être matérialisées dans des tables de distances, ramenant ainsi la fouille de données spatiales à l'analyse multi-tables. Or, les méthodes de fouilles de données traditionnelles considèrent une seule table en entrée où chaque tuple est une observation à analyser. De simples jointures entre ces tables ne résoud pas le problème et fausse les résultats en raison du comptage multiple des observations. Nous proposons trois alternatives de fouille de données multi-tables dans le cadre de la fouille des données spatiales. La première consiste à interroger à la volée les différentes tables et modifie en dur les algorithmes existants. La seconde est une optimisation de la première qui pré-calcule les jointures et adapte les algorithmes existants. La troisième réorganise les données dans une table unique en complétant - et non en joignant- la table d'analyse par les données présentes dans les autres tables, ensuite applique un algorithme standard sans modification. Cet article présente ces trois alternatives. Il décrit leur implémentation pour la classification supervisée et compare leur performance.

1. Positionnement du problème

La fouille de données spatiales (FDS) est aujourd'hui un domaine bien identifié de la fouille de données. Elle est née du besoin d'exploitation dans un but décisionnel de données à caractère spatial produites, importées ou accumulées, susceptibles de délivrer des informations ou des connaissances par le moyen d'outils exploratoires (Zeitouni 2000). Sa principale caractéristique est qu'elle considère les relations spatiales – qu'on appellera de voisinage – (Egenhofer et al. 1993). Ces relations sont à l'origine implicites et nécessitent des jointures coûteuses sur des critères spatiaux pour être exhibées. Nous avons proposé dans nos travaux antérieurs de les matérialiser en utilisant une structure secondaire appelée index de jointure spatial (Zeitouni et al. 2000). L'idée est de pré-calculer la relation spatiale exacte entre les localisations de deux collections d'objets spatiaux et de la stocker dans une table de type (objet1, relation-spatiale, objet2). Ceci nous permet de pallier le problème du coût des jointures spatiales au moment de l'analyse. Néanmoins, cette organisation ne peut pas être directement analysée par les méthodes de fouilles de données car celles-ci considèrent que les données en entrée sont dans une table unique et que chaque tuple de cette table constitue une observation ou un individu à analyser. On se trouve alors confronté au problème qu'on ne peut exploiter telles quelles les données organisées en plusieurs tables. Il est possible de se ramener à une seule table en joignant les différentes tables initiales. Or, cette jointure peut dupliquer des tuples car les observations à analyser sont en liaison N-M avec les objets

voisins. Ceci fausse les résultats des méthodes de fouille de données en raison du multiple comptage de ces observations.

Cet article propose trois autres alternatives de fouille de données multi-tables dans le cadre de la fouille de données spatiales. La première consiste à interroger à la volée les trois tables. Elle fait appel à des requêtes comprenant des jointures et des agrégats chaque fois que nécessaire. La seconde effectue, une fois pour toutes, les jointures sur clés entre les différentes tables et modifie les algorithmes classiques en évitant les comptages multiples des tuples. Ce qui revient à optimiser la première méthode. La troisième transforme la structure multi-tables en une table unique de manière à ne pas dupliquer les observations et applique ensuite une méthode de fouille de données classique. Nous décrivons, dans la section 2, ces trois alternatives. Dans la section 3, des expérimentations et des tests de performances seront présentés, suivis par une discussion et une conclusion.

2. Solutions proposées

Comme souligné précédemment, l'introduction de l'index de jointure spatial dans la fouille de données spatiales a le grand avantage de ramener cette dernière à la fouille de données multi-tables. Or, cette nouvelle organisation de données n'est pas directement exploitable par les méthodes de fouilles de données car ces dernières considèrent que les données en entrée sont dans une table unique. Récemment, des travaux dits de fouille de données relationnelles ont abordé ce problème de fouille de données multi-tables (Dzeroski et al. 2001) en s'appuyant sur la programmation logique inductive (PLI). Leur inconvénient est que leur utilisation nécessite la transformation coûteuse des données relationnelles en un ensemble de faits exprimés en logique du 1er ordre. A la place, nous proposons trois alternatives de fouille de données multi-tables dans le cadre de la fouille de données spatiales. Elles sont détaillées ci-dessous.

2.1. Alternative 1: Interroger à la volée les différentes tables

Contrairement aux méthodes classiques, la méthode proposée prend en entrée trois tables : table des objets à analyser, tables des objets du voisinage et les index de jointures spatiaux. Chaque fois que l'attribut à analyser est un attribut du voisinage, l'algorithme fait appel à une double jointure entre la table de faits, l'index et la table de voisinage (c'est là que réside la modification à apporter aux algorithmes existants), sinon on applique l'algorithme classique sans modification. Un exemple d'algorithme en utilisant cette alternative est donné dans (Chelghoum et al. 2002).

Le défaut de cette solution est que le temps d'exécution se dégrade très vite avec l'augmentation du volume de données (cf. FIG 1) car des requêtes de jointures coûteuses sont invoquées dans une boucle, multipliant les accès aux tables. Ceci nous amené à proposer une autre alternative que nous présentons ci-dessous.

2.2. Alternative 2: Matérialiser les jointures

Cette alternative consiste à effectuer, une fois pour toutes, les jointures sur clés entre les trois tables. Ce qui évite la multiplication de requêtes de jointures de la première alternative. Cependant, le résultat de ces jointures mène à la duplication des objets de l'analyse. Ce qui oblige à modifier les algorithmes de fouille de données afin de prendre en compte cette duplication dans les calculs (on ne compte l'observation qu'une seule fois). Cette alternative a l'avantage d'être plus rapide que la précédente grâce à la matérialisation des jointures.

Faisons remarquer que ces deux alternatives (1 et 2) ne sont pas spécifiques à la fouille de données spatiales, mais s'appliquent dans un cas multi-tables en général. Elles présentent néanmoins l'inconvénient de ne pouvoir utiliser les outils de fouille de données existants car l'adaptation doit se faire en dur pour chaque algorithme.

2.3. Alternative 3: Réorganiser les données en une seule table

Il s'agit de se ramener à une table unique en effectuant des jointures sur les trois tables sans dupliquer les objets. L'idée ici est de compléter, et non pas de joindre, la table d'analyse par des données présentes dans les autres tables. On propose un opérateur dit CROISEMENT défini ci-dessous et dont le principe est de générer pour chaque valeur d'attribut de la table liée un attribut dans la table résultat. L'application préalable de cet opérateur a l'avantage d'éviter la duplication des objets et de permettre ensuite l'utilisation de n'importe quelle méthode de fouille de données, sans la moindre modification.

Définition de l'opérateur CROISEMENT

Soient R (ID1, A_1, …, A_n), V (ID2, B_1, …, B_m) et I (ID1, ID2, P) trois tables dont les clés sont celles soulignées et telles que les attributs B_i sont qualitatifs et b_{ij} ($j = 1, \ldots K_i$) leurs valeurs distinctes. Soit $F = \{F_1, F_2, \ldots, F_m\}$ un ensemble de fonctions agrégats.

CROISEMENT (R, V, I, F) est une table T ayant le schéma suivant :
T (ID1, A_1, …, A_n, P_b_{11}, …, P_b_{1K1}, **……,** P_b_{m1}, …, P_b_{mKm}) de clé ID1 et où :
$\forall$ t = (id1, a_1, a_2, …, a_n, P_b_{11}, …, P_b_{1K1}, **……** , P_b_{m1}, P_b_{m2}, …, P_b_{mKm}) $\in$ T,

- (id1, a_1, a_2, …, a_n) = $\sigma_{(ID1 = Id1)}$ (R)
- P_b_{ij} = F_i ($\sigma_{(ID1 = Id1)}$ (I) |X| $\sigma_{(Bi = bij)}$ (V) ; P) si $\sigma_{(ID1 = Id1)}$ (I) est non vide, la valeur nulle sinon.

En fait, I est une table de correspondances pondérées qui relie la table R de « faits » à analyser avec une table comprenant des « dimensions » à considérer. Cet opérateur n'est recommandé que lorsque les attributs B_i de V comportent assez peu de valeurs distinctes. La propriété de l'opérateur CROISEMENT est que le résultat comporte toujours comme partie gauche la table R dans son intégralité sans duplication de ses tuples et que celle-ci est complétée en partie droite par les poids des « dimensions » provenant de V et de I. Ce cas de figure arrive couramment dans les données relationnelles où la table de correspondance traduit les liens de cardinalité N-M. Cet opérateur peut être vu comme un moyen de préparation des données relationnelles pour la fouille de données.

Faisons remarquer que pour un même tuple de R et une même valeur d'attribut b_{ij} de V, il peut y avoir plusieurs liens dans I avec des poids éventuellement différents. La valeur P_b_{ij} du résultat T devant être unique, nous avons introduit des fonctions agrégats afin de calculer une seule valeur pour P_b_{ij}. En cas d'inexistence de ce lien, la valeur nulle remplace cette fonction. Les tuples n'ayant pas de correspondant dans I sont complétés par des valeurs nulles tout comme une jointure externe gauche.

Récemment un opérateur UNPIVOT a été implémenté dans les modules d'extraction transformation chargement ou ETL d'un entrepôt de données (Oracle9i 2003). Une autre forme d'opérateur UNPIVOT a été proposée dans (Graefe et al 1998). La différence entre l'opérateur CROISEMENT que nous avons proposé et l'opérateur UNPIVOT est que le premier, contrairement au second, prend en entrée trois tables et intègre les fonctions

agrégats. Ce qui répond à notre problématique. L'expression de l'opérateur CROISEMENT par UNPIVOT d'Oracle est possible et donne la formule suivante :

CROISEMENT (R,I,V,F) = R |X| UNPIVOT (F_1 (I |X| V, ID_1 , B_1; P)) |X| UNPIVOT (F_2 (I |X| V, ID_1 , B_2 ; P)) |X|…|X| UNPIVOT (F_m (I |X| V, ID_1 , B_m; P))

Cependant, une implémentation directe de cet opérateur évite les multiples jointures dans le cas de plusieurs attributs ainsi que le stockage des résultats intermédiaires de UNPIVOT.

3. Expérimentations et performances

C'est dans le cas d'analyse de la sécurité routière que nous avons testé les trois alternatives précédentes. L'objectif est de construire un modèle prédictif en recherchant des correspondances entre les accidents et les autres couches thématiques comme le réseau ou le tissu urbain. Les expérimentations que nous présentons ci-dessous porteront sur l'application des alternatives décrites dans cet article dans le cas spécifique d'un arbre de décision spatial. Plus précisément, nous avons appliqué ces alternatives avec la méthode CART (Breiman et al. 1984) générant ainsi un arbre de décision spatial. Le tableau ci-dessous (

TAB 1) résume les coûts d'exécution en seconde de chacune des alternatives. Ces expérimentations ont été effectuées sur un PC équipée d'un processeur Pentium IV de 2.5 Ghz. L'implémentation a été réalisée dans l'environnement Oracle 9i et JAVA. Les algorithmes 1, 2 et 3 correspondent respectivement aux alternatives 1, 2 et 3. La phase 1 correspond à l'étape de transformation préalable des données, autrement dit, à la matérialisation de la jointure pour l'alternative 2 ou à l'application de l'opérateur CROISEMENT pour l'alternative 3. La phase 2 représente l'étape de construction de l'arbre de décision.

Les tests visent à comparer les performances de chacune des alternatives. Nous avons retenu trois critères : la taille de la table cible, la taille de la table liée (voisin dans le cas de la FDS) et la taille de la table de correspondance (index de jointures spatial en FDS). La FIG 1 donne le temps d'exécution des trois algorithmes en fonction de la taille de la table cible.

L'analyse de ces résultats montre que le temps d'exécution de l'algorithme 1 est nettement plus important que dans les algorithmes 2 et 3. En fait, les jointures répétitives et coûteuses appelées dans les calculs pénalisent l'algorithme 1. Par exemple, pour chaque combinaison de classe, de valeur d'attribut et de lien dans l'algorithme 1, plusieurs jointures sont invoquées. On constate également que l'algorithme 3 est un peu moins performant que l'algorithme 2. Ceci est dû à une augmentation du volume des données, comparativement aux résultats de jointures, suite à l'opérateur de CROISEMENT. Ce qui peut arriver lorsque les valeurs distinctes sont nombreuses ou que peu d'entre elles sont liées avec la table d'analyse. Cette baisse est constatée dans la phase de préparation des données (phase1). La phase 2 est pratiquement équivalente entre les algorithmes 2 et 3.

Conclusion

Cet article montre comment traduire tout problème de fouille de données spatiales en un problème de fouille de données multi-tables et ensuite, ramener cette fouille multi-table à son tour à un problème de fouille de données mono-table et ce grâce à l'opérateur CROISEMENT défini ici. Le grand avantage est de permettre l'utilisation de n'importe quel

algorithme ou outil de fouille de données (clustering, règles d'associations, etc.). Une version détaillée de cet article est donnée sur le site http://www.prism.uvsq/fr/recherche/rapports.

Nous avons proposé et analysé trois alternatives de fouilles de données multi-tables. Leur application à la méthode d'arbre de décision spatial a été décrite. Leurs performances ont été exposées. Les résultats obtenus avec le prototype implémenté confirment l'efficacité de notre approche. Néanmoins, les tests effectués peuvent guider la recommandation de l'une ou l'autre de ces alternatives. Ainsi, lorsque l'on veut avoir la flexibilité d'appliquer n'importe quel outil et algorithme de fouille de donnée, l'utilisation préalable de l'opérateur CROISEMENT est préconisée. L'algorithme 1 constitue une méthode naïve beaucoup trop coûteuse en temps d'exécution qu'il faut évidemment éviter. La matérialisation de la jointure s'accompagne de la modification en dur des algorithmes. Hormis cet inconvénient, elle apparaît la plus rapide.

Des pistes pour faire évoluer ces méthodes sont envisagées. L'optimisation de CART classique et spatial pourrait utiliser les préparations aux calculs de (Graefe et al 1998). Il faudra mesurer précisément le gain en considérant le coût des prétraitements. Au-delà, ces comptages préalables pourraient inspirer l'optimisation d'autres méthodes de fouille de données complexes comme les règles d'association. Egalement en perspectives, l'extension de l'opérateur dans le cadre d'un schéma d'entrepôt de données ou plusieurs tables sont liées en étoile. Ce qui amènerait à considérer des index multidimensionnels à la place de l'index de jointure. Ce qui a guidé ce travail au départ est un problème de fouille de données spatiales, mais cette recherche va au-delà pour couvrir des problèmes communs de fouille de données complexes de part leur organisation.

			Algo 1	Algo 2			Algo 3		
Taille de R (objets)	Taille de I (objets)	Taille de V (objets)	Durée totale (s)	Durée totale (s)	Durée de la phase 1 (s)	Durée de la phase 2 (s)	Durée totale (s)	Durée de la phase1 (s)	Durée de la phase2 (s)
122	147	37	240	3	1	2	3	1	2
204	148	52	300	4	1	3	3	1	2
1594	6330	869	16860	7	1	6	9	5	4
3437	20180	869	24799	8	1	7	76	71	5
8668	20180	869	35205	18	2	16	157	150	7
15574	27054	869	48403	19	2	17	640	626	14
21892	53631	869	70020	56	2	54	925	882	43
29810	74302	869	88320	32	2	30	1372	1330	42

TAB 1- *Les temps d'exécution*

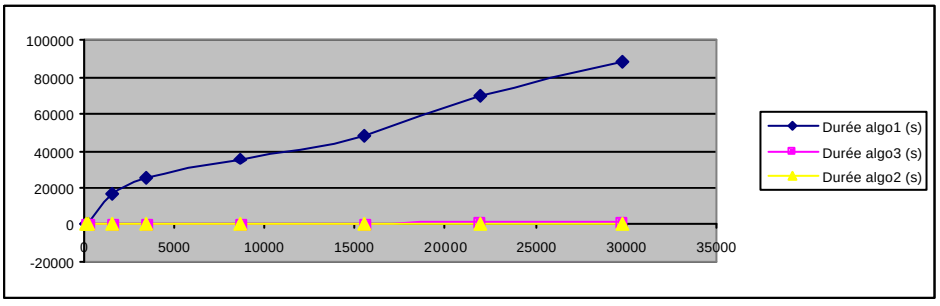

FIG 1- *temps d'exécution en fonction de la taille de la table cible*

Références

[Breiman et al. 1984] Breiman L., Friedman J.H., Olshen R.A., Stone C.J, Classification and Regression Trees. Ed: Wadsworth & Brooks. Montery, California, 1984.

[Chelghoum et al. 2002] Chelghoum N., Zeitouni K. (2002), A decision tree for multi-layered spatial data, In : Joint International Symposium on Geospatial Theory, Processing and Applications, Ottawa, Canada, July 8-12 2002.

[Dzeroski et al. 2001] Dzeroski S., Lavrac N., Relational Data Mining, Springer, 2001.

[Egenhofer et al. 1993] Egenhofer M.J., Sharma J., Topological Relations Between Regions in R2 and Z2, 5th International Symposium,SSD'93, Singapore, June1993, Springer-Verlag, pp. 316-331.

[Graefe et al. 1998] Graefe G., Fayyad U., Chaudhuri S., On the efficient gathering of sufficient statistics for classification of large SQL databases, In Proceedings of the Fourth International Conference on Knowledge Discovery and Data-Mining (KDD 98), AAAI Press, New York City, August 27-31, 1998.

[Knobbe et al. 1999] Knobbe. A.J., Siebes A., Wallen V., Daniel M.G., Multi-relational Decision Tree Induction, In Proceedings of PKDD' 99, Prague, Czech Republic, Septembre 1999.

[Lefébure et al. 1998] Lefébure R., Venturi G., Le Data Mining, Eyrolles, 1998.

[Oracle9i 2003] Oracle9i Warehouse Builder Transformation Guide, Release 2 (9.0.4) for Windows and UNIX, Part No. B10658-01, February 2003, Oracle Corporation

[Quinlan 1996] Quinlan J.R., Induction of Decision Trees, Machine Learning (1), pp 82-106,1986.

[Valduriez 19987] Valduriez P., Join indices, ACM Transactions on Database Systems, 12(2), pp 218-246.

[Zeitouni 2000] Zeitouni K., Fouille de données spatiales, Revue internationale de géomatique n° 4/99, Numéro spécial, Edition Hermès Sciences, Avril 2000.

[Zeitouni et al. 2000] Zeitouni K.,Yeh L., Aufaure M-A., Join indices as a tool for spatial data mining, Int. Workshop on Temporal, Spatial and Spatio-Temporal Data Mining, Lecture Notes in Artificial Intelligence n° 2007, Springer, pp 102-114, Lyon, France, September 12-16, 2000.

Summary

Spatial data mining requires the analysis of object interactions within the space. These interactions can be materialized using distance tables, reducing spatial data mining to multi-table analysis. However, conventional data mining algorithms consider only one input table with one observation by row. Classical joins between these tables doesn't solve the problem and mislead the results because of the multiple counting of duplicated observations. We propose three alternatives of multi-table data mining in the context of spatial data mining. The first makes a hard modification of the classification algorithm in order to consider those tables. The second is an optimization of the first approach that pre-computes all join operations and adapts the classification algorithm. The third re-organizes data into a unique table by complementing – not joining – the target table by relevant data from the other tables, then applies any standard data mining algorithm. This article presents these three alternatives. It describes their implementation for the supervised classification and compares their performances.