Kỷ yếu Hội nghị Khoa học Quốc gia lần thứ IX "Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR'9)"; Cần Thơ, ngày 4-5/8/2016
DOI: 10.15625/vap.2016.00030

KHAI THÁC LUẬT PHÂN LỚP KẾT HỢP TRÊN CƠ SỞ DỮ LIỆU MẤT CÂN BẰNG VỀ LỚP

Nguyễn Thị Thúy Loan[1], Trần Thị Minh Thúy[2], Giang Hào Côn[1]

[1]Khoa Công nghệ thông tin, Đại học Nguyễn Tất Thành, Tp.HCM
[2]Khoa Công nghệ thông tin, Trung cấp Kinh tế kỹ thuật Quận 12, Tp.HCM

nttloan@ntt.edu.vn; ttmthuy@dttec.edu.vn; ghcon@ntt.edu.vn

TÓM TẮT: *Phân lớp dựa vào luật phân lớp kết hợp đã được chứng minh là tốt hơn các phương pháp phân lớp dựa vào luật hiện có như cây quyết định, ILA, v.v. Tuy nhiên, do dựa vào khai thác luật kết hợp nên chỉ những luật phổ biến (có độ hỗ trợ cao) được khai thác. Trong các cơ sở dữ liệu (CSDL) mất cân bằng về lớp, mặc dù các lớp thiểu số cũng đóng vai trò quan trọng nhưng chúng sẽ không được khai thác khi dựa vào luật phân lớp kết hợp. Trong bài báo này, chúng tôi đề xuất một phương pháp biến đổi CSDL sao cho sự phân bố các lớp được cân bằng, sau đó khai thác luật phân lớp kết hợp dựa trên tập dữ liệu đã biến đổi. Để biến đổi dữ liệu, chúng tôi chia tập dữ liệu thành m tập con, mỗi tập con tương ứng với một giá trị của thuộc tính lớp. Với mỗi tập dữ liệu, chúng tôi sử dụng K-means để gom chúng thành k nhóm (k chính là số dòng dữ liệu của tập dữ liệu có ít dòng nhất). Với mỗi nhóm, chúng tôi chọn dòng đại diện chính là dòng có khoảng cách gần với trọng tâm nhất. Sau khi gom nhóm, chúng tôi tập hợp dữ liệu lại và sử dụng CAR-Miner để khai thác luật phân lớp. Kết quả thực nghiệm cho thấy phương pháp của chúng tôi thường có độ chính xác cao hơn so với phương pháp khai thác luật phân lớp từ toàn bộ cơ sở dữ liệu.*

Từ khoá*: Khai thác luật phân lớp kết hợp, gom nhóm, cơ sở dữ liệu mất cân bằng về lớp, độ chính xác.*

I. GIỚI THIỆU

Khai thác luật phân lớp kết hợp được đề xuất bởi Liu và các đồng sự vào năm 1998 [2]. Thuật toán CBA cũng đã được đề xuất trong công trình này. Phương pháp này thường cho độ chính xác cao hơn so với các phương pháp phân lớp dựa trên luật khác như cây quyết định [8], ILA [15], v.v. Từ đó đến nay, đã có nhiều thuật toán được phát triển nhằm làm tăng độ chính xác, giảm thời gian khai thác như CMAR [3], MMAC [10], MCAR [11], ECR-CARM [12], CAR-Miner [6], CAR-Miner-Diff [7]. Trong số các thuật toán kể trên, CMAR và MMAC đề xuất phương pháp dự đoán lớp của mẫu mới dựa vào đa luật nên thường có độ chính xác cao hơn so với CBA. ECR-CARM, CAR-Miner và CAR-Miner-Diff tập trung giải quyết vấn đề thời gian khai thác sao cho tập luật khai thác được vẫn bảo đảm như CBA/CMAR nhưng thời gian khai thác nhanh hơn. Một trong các điểm yếu của phân lớp dựa vào luật phân lớp kết hợp là chọn ngưỡng độ hỗ trợ tối thiểu. Một ngưỡng quá cao dẫn đến các lớp chứa ít mẫu sẽ không phổ biến và vì vậy, không luật nào chứa lớp này sẽ ảnh hưởng đến giai đoạn dự đoán lớp. Trong khi đó nếu chọn ngưỡng độ hỗ trợ tối thiểu thấp để khai thác được các luật chứa lớp thiểu số thì số lượng luật của lớp đa số vẫn áp đảo nên cũng ảnh hưởng đến giai đoạn dự đoán lớp.

Để cải thiện khuyết điểm này, chúng tôi đề xuất một phương pháp cân bằng lại dữ liệu trên các cơ sở dữ liệu mất cân bằng về lớp, nhằm cân bằng tỉ lệ luật được sinh ra giữa các lớp góp phần tăng độ chính xác cho giai đoạn dự đoán lớp. Đầu tiên, chúng tôi chia tập dữ liệu có n mẫu chứa m lớp thành m tập dữ liệu con, tập dữ liệu con thứ i sẽ chứa các mẫu dữ liệu có giá trị lớp thứ i. Sau đó, với những tập dữ liệu có số dòng lớn hơn k (k là số dòng dữ liệu của tập dữ liệu có ít dòng nhất), chúng tôi sử dụng K-means (thường được ứng dụng bởi cộng đồng khai thác dữ liệu [14]) để gom tập dữ liệu này thành k nhóm và chọn mỗi nhóm một mẫu đại diện. Như vậy, số dòng của mỗi tập dữ liệu chỉ còn k dòng. Cuối cùng, chúng tôi tập trung m tập dữ liệu lại (mỗi tập có k dòng nên CSDL tổng hợp đã cân bằng) và sử dụng thuật toán CAR-Miner để khai thác luật.

Phần còn lại của bài báo được tổ chức như sau: Phần 2 trình bày các định nghĩa và khái niệm liên quan đến bài toán khai thác luật phân lớp kết hợp và gom nhóm dữ liệu. Phần 3 trình bày một số nghiên cứu liên quan đến bài toán khai thác luật phân lớp kết hợp và gom nhóm dữ liệu. Phần 4 trình bày về phương pháp đề xuất bao gồm các bước thực hiện, các thuật toán sẽ được áp dụng tại mỗi bước và nhận xét đánh giá về phương pháp đề xuất. Phần 5 trình bày kết quả so sánh về độ chính xác giữa phương pháp đề xuất và phương pháp khai thác dựa trên toàn bộ tập dữ liệu. Kết luận và hướng phát triển được trình bày ở phần 6.

II. MỘT SỐ ĐỊNH NGHĨA VÀ KHÁI NIỆM

Khai thác luật phân lớp dựa vào khai thác luật kết hợp (**C**lass **A**ssociaton **R**ule**s** – CARs) là tìm một tập con của các luật kết hợp có trong cơ sở dữ liệu. Mỗi luật trong tập con này chứa vế phải là giá trị của thuộc tính lớp. Bài toán được phát biểu như sau:

Cho cơ sở dữ liệu D, I là tập tất cả các item trong D và Y là tập các nhãn lớp. Luật phân lớp kết hợp là một biểu thức có dạng $X \rightarrow y$ trong đó $X \subseteq I$ và $y \in Y$. Độ tin cậy của luật là c nếu c% mẫu trong D chứa X được gán nhãn là lớp y. Độ phổ biến của luật là s nếu có s% mẫu trong D chứa X được gán nhãn là lớp y.

Mục tiêu của khai thác luật phân lớp dựa vào khai thác luật kết hợp là:

(1) Khai thác tập CARs thỏa ngưỡng độ hỗ trợ tối thiểu (*MinSup*) và ngưỡng độ tin cậy tối thiểu (*MinConf*).

(2) Xây dựng bộ phân lớp từ CARs.

Một cách hình thức, bài toán khai thác CARs được phát biểu như sau: Cho D là một CSDL huấn luyện với n thuộc tính $A_1, A_2, \ldots, A_n$, mỗi thuộc tính có một tập các giá trị tương ứng. C là thuộc tính lớp chứa k giá trị khác nhau $c_1, c_2, \ldots, c_k$ đại diện các lớp trong D.

Chẳng hạn, cho D là CSDL huấn luyện được cho trong bảng 1, với 8 dòng dữ liệu ($|D| = 8$), trong đó A = {A, B, C}, lớp là thuộc tính quyết định, chẳng hạn Class = {*y*, *n*}, là hai lớp.

Bảng 1. Một ví dụ về cơ sở dữ liệu huấn luyện mẫu

OID	A	B	C	Class
1	*a1*	*b1*	*c1*	*y*
2	*a1*	*b2*	*c1*	*n*
3	*a2*	*b2*	*c1*	*n*
4	*a3*	*b3*	*c1*	*n*
5	*a3*	*b1*	*c2*	*n*
6	*a3*	*b3*	*c1*	*y*
7	*a1*	*b3*	*c2*	*y*
8	*a2*	*b2*	*c2*	*n*

Định nghĩa 1: Itemset là một tập các thuộc tính cùng với giá trị xác định của nó trong tập đó, kí hiệu $<(A_{i1}, a_{i1}), \ldots, (A_{im}, a_{im})>$.

Định nghĩa 2: Luật phân lớp r là phép kéo theo có dạng $<(A_{i1}, a_{i1}),\ldots, (A_{im}, a_{im})> \rightarrow c_j$. Trong đó $<(A_{i1}, a_{i1}),\ldots, (A_{im}, a_{im})>$ là một Itemset còn $c \in C$ là một nhãn lớp.

Định nghĩa 3: Độ phổ biến r, ký hiệu là Sup(r), là số dòng của r chứa cả vế trái lẫn vế phải.

Định nghĩa 4: Độ tin cậy của r, ký hiệu là *Conf*(r), $Conf(r) = Sup\{(A_{i1}, a_{i1}),\ldots,(A_{im}, a_{im}), c\}/ Sup\{(A_{i1}, a_{i1}), \ldots, (A_{im}, a_{im})\}$.

Ví dụ: Xét luật r: $(A, a_1) \rightarrow y$ với $X = (A, a_1)$ và $c_i = y$ từ bảng 1, chúng ta có $Supp(X) = 3$, $Supp(r) = 2$, và $Conf(r) = \dfrac{Supp(r)}{Supp(X)} = \dfrac{2}{3}$.

III. CÁC NGHIÊN CỨU LIÊN QUAN

A. Khai thác luật phân lớp kết hợp

Năm 1998, Liu và các đồng sự đề xuất phương pháp CBA [2] (Classification based on associations) để khai thác luật phân lớp kết hợp. CBA bao gồm hai giai đoạn chính:

- Giai đoạn sinh luật – thuật toán CBA-RG.
- Giai đoạn xây dựng bộ phân lớp.

Năm 2001, Li và các đồng sự đã đề xuất thuật toán CMAR (classification based on multiple association rules) [3]. Phương pháp này dựa vào FP-tree để nén dữ liệu và dùng phép chiếu trên cây để tìm luật phân lớp. Vào năm 2004, Thabtah và các đồng sự đã đề xuất thuật toán MMAC (multi-class, multi-label associative classification) [9]. Năm 2008, Vo và Le đề xuất thuật toán ECR-CARM (Equivalence class rule – class association rule mining) [12]. Đầu tiên, các tác giả đề xuất cấu trúc cây ECR, dựa trên cây này, các tác giả đề xuất thuật toán ECR-CARM để khai thác CARs với chỉ một lần quét CSDL và dựa vào phần giao giữa các tập định danh các đối tượng để tính nhanh độ hỗ trợ của các itemset và luật. Mặc dù ECR-CARM có một số ưu điểm như chỉ quét CSDL một lần và khai thác luật nhanh nhưng vẫn còn một số hạn chế như sau: Do nhóm tất cả các giá trị của cùng một tập thuộc tính thành một nút nên ECR-CARM tốn thời gian kiểm tra tiền tố để kết hợp các phần tử từ các nút trên cây. Chính vì vậy, năm 2013, Nguyen và các đồng sự đề xuất thuật toán CAR-Miner để khai thác nhanh CARs [6]. CAR-Miner đã cải tiến ECR-CARM bằng cách mỗi nút chứa một giá trị (thay vì tập các giá trị như ECR-CARM) và vì vậy, nó không cần kiểm tra tiền tố. Dựa trên cấu trúc mới (MECR-tree), CAR-Miner phát triển hai định lý nhằm tỉa sớm các ứng viên không phổ biến và xác định nhanh các thông tin của nút con dựa trên thông tin của nút cha. Ngoài ra, Nguyen và các đồng sự cũng sử dụng kỹ thuật diffset để khai thác nhanh CARs [7]. Ngoài ra, việc khai thác CARs với ràng buộc [5] và khai thác CARs sử dụng trong dữ liệu chứng khoáng [1] cũng đã được đề xuất.

B. Thuật toán K-means

Thuật toán K-means do MacQueen đề xuất vào năm 1967 [4]. Thuật toán dựa trên độ đo khoảng cách của các đối tượng dữ liệu trong nhóm. Trong thực tế, nó đo khoảng cách đến giá trị trung bình của các dữ liệu trong nhóm, đó

chính là trọng tâm nhóm. Do đó, cần khởi tạo một tập trọng tâm các nhóm ban đầu, thông qua đó lặp lại các bước: gán nhãn mỗi đối tượng tới trọng tâm gần nhất và tính lại trọng tâm của mỗi nhóm trên cơ sở gán mới cho các đối tượng. Quá trình này dừng lại khi các trọng tâm nhóm hội tụ. K-means là một thuật toán gom nhóm dữ liệu được ứng dụng rộng rãi trong cộng đồng khai thác dữ liệu [13]. Đây là một thuật toán được xếp thứ hai trong top 10 thuật toán khai thác dữ liệu (được đề cử bởi các nhà khoa học uy tín về khai thác dữ liệu tại hội nghị IEEE ICDM 2006) [14].

IV. PHƯƠNG PHÁP ĐỀ XUẤT

Cách tiếp cận của bài báo được chia làm 2 giai đoạn.

A. Giai đoạn tạo luật phân lớp dựa trên gom nhóm

Trong giai đoạn này, tập dữ liệu huấn luyện được chia thành m tập con tương ứng với m lớp có trong CSDL huấn luyện. Với mỗi tập con có số mẫu lớn hơn *k* (với *k* là số mẫu của tập con chứa ít mẫu nhất), sử dụng thuật toán K-means để gom các mẫu này thành *k* nhóm, mỗi nhóm chỉ giữ lại một mẫu đại diện (là phần tử gần trọng tâm nhóm nhất chẳng hạn). Như vậy, cuối cùng mỗi tập con sẽ giữ lại *k* mẫu. Cách tiếp cận này giúp CSDL tổng hợp có số mẫu thuộc mỗi lớp là cân bằng và vì vậy, phát huy tốt ưu điểm của các phương pháp khai thác luật phân lớp kết hợp. Các bước thực hiện của giai đoạn này được trình bày trong Hình 1.

a) Các bước thực hiện

Bước 1: Chia CSDL thành *m* bảng con tương ứng với *m* giá trị của thuộc tính lớp. Gọi *k* là số dòng dữ liệu của bảng con có số dòng ít nhất.

Bước 2: Với mỗi bảng con có số dòng dữ liệu lớn hơn *k*, tiến hành gom nhóm các dòng dữ liệu trong bảng con đó thành *k* nhóm. Mỗi nhóm chỉ chọn một mẫu đại diện.

Bước 3: Thực hiện khai thác luật phân lớp kết hợp trên tập dữ liệu tổng hợp từ *m* nhóm.

Hình 1. Các bước thực hiện của phương pháp tạo luật phân lớp dựa vào gom nhóm

Ở bước 2, chúng ta có thể chọn K-means, K-medoids hoặc phương pháp phân cấp để thực hiện. Do K-means là thuật toán lặp đơn giản và cũng gom nhóm khá hiệu quả nên trong phạm vi bài báo, chúng tôi sử dụng K-means cho bước này. Để chọn mẫu đại diện cho mỗi nhóm, cách đơn giản nhất là chọn phần tử gần trọng tâm của nhóm nhất.

Ở bước 3, chúng ta có thể sử dụng bất kỳ thuật toán phân lớp kết hợp nào để khai thác luật. Kết quả thực nghiệm từ [6] cho thấy CAR-Miner thường hiệu quả hơn so với các thuật toán sinh luật trước đó nên chúng tôi sử dụng CAR-Miner cho bước này. Ở bước này, chúng ta có thể sử dụng phương pháp tỉa luật thừa như được sử dụng trong [2] hay [13] nhằm giảm thiểu số lượng luật cần xét trong giai đoạn 2.

b) Ví dụ minh họa

Bảng 2. Dữ liệu mất cân bằng về lớp (5 mẫu thuộc lớp 0 và 2 mẫu thuộc lớp 1)

OID	A	B	C	D	CLASS
1	5	3	5	4	0
2	4	1	4	3	0
3	5	2	4	4	0
4	6	6	5	4	0
5	6	3	5	3	0
6	2	1	2	2	1
7	4	2	4	3	1

Bước 1**:** Đầu tiên chia cơ sở dữ liệu thành 2 bảng con tương ứng với 2 lớp 0 và 1

Do số dòng dữ liệu chứa lớp 1 là ít nhất (2 dòng) nên $k = 2$.

Bước 2: Dùng K-means gom các dòng có lớp là 0 thành 2 cụm. Mỗi cụm rút ra 1 dòng đại diện, ta có kết quả như bảng 3 (bên dưới sau khi đã được đánh lại OID).

Bước 3: Thực hiện khai thác luật phân lớp kết hợp bằng thuật toán CAR-Miner với dữ liệu trong bảng 3.

Bảng 3. Bảng CSDL phân lớp

OID’	A	B	C	D	CLASS
1	2	1	2	2	1
2	4	2	4	3	1
3	6	6	5	4	0
4	6	3	5	3	0

Cây MECR được xây dựng từ CSDL trong bảng 3 như sau: Đầu tiên, nút gốc L_r của cây chứa các nút 1-itemset phổ biến như sau:

$$\left\{\begin{pmatrix}1\times2\\1(0,\underline{1})\end{pmatrix},\begin{pmatrix}1\times4\\2(0,\underline{1})\end{pmatrix},\begin{pmatrix}1\times6\\34(\underline{2},0)\end{pmatrix},\begin{pmatrix}2\times1\\1(0,\underline{1})\end{pmatrix},\begin{pmatrix}2\times2\\2(0,\underline{1})\end{pmatrix},\begin{pmatrix}2\times6\\3(\underline{1},0)\end{pmatrix},\begin{pmatrix}2\times3\\4(\underline{1},0)\end{pmatrix},\begin{pmatrix}4\times2\\1(0,\underline{1})\end{pmatrix},\begin{pmatrix}4\times4\\2(0,\underline{1})\end{pmatrix},\begin{pmatrix}4\times5\\34(\underline{2},0)\end{pmatrix},\begin{pmatrix}8\times2\\1(0,\underline{1})\end{pmatrix},\begin{pmatrix}8\times3\\24(\underline{1},1)\end{pmatrix},\begin{pmatrix}8\times4\\3(\underline{1},0)\end{pmatrix}\right\}$$

Áp dụng thuật toán CAR-Miner với *MinSup* = 10% và *MinConf* = 60% để tính toán cho các itemset. Thủ tục CAR-Miner được gọi với tham số L_r

Áp dụng thủ tục CAR-Miner được gọi với tham số L_r. Nút $l_i = \begin{pmatrix}1\times2\\1(0,\underline{1})\end{pmatrix}$

- Xét nút $l_j = \begin{pmatrix}1\times4\\2(0,\underline{1})\end{pmatrix}$: hai nút l_i và l_j có cùng thuộc tính và khác giá trị nên không kết với nhau. Tương tự các nút $\begin{pmatrix}1\times6\\34(\underline{2},0)\end{pmatrix}$ cũng không kết với nhau.

- Xét nút $l_j = \begin{pmatrix}2\times1\\1(0,\underline{1})\end{pmatrix}$: Vì hai nút này khác thuộc tính nhau, nên ba yếu tố được tính lại như sau O.att = l_i.att ∪ l_j.att = 1 | 2 = 3 hoặc 11 theo biểu diễn bit; O.values = l_i.values ∪ l_j.values = *2 ∪ 1 = 21* và $O.Obidset = l_i.Obidset \cap l_j.Obidset$ = {1} ∩ {1} = {1}. Bởi vì $|l_i.Obidset| = |O.Obidset|$, thuật toán sẽ chép thông tin từ l_i xuống O vì nút O là con của nút l_i. Điều đó có nghĩa rằng O.count = l_i.count = (0,1) và O.pos = 2. Vì O.count[O.pos] = 1 > *MinSup*, O được thêm vào $P_i \Rightarrow P_i = \left\{\begin{pmatrix}3\times21\\1(0,\underline{1})\end{pmatrix}\right\}$

- Xét nút $l_j = \begin{pmatrix}2\times2\\2(0,\underline{1})\end{pmatrix}$: Vì hai nút này khác thuộc tính nhau, nên ba yếu tố được tính lại như sau O.att = l_i.att ∪ l_j.att = 1 | 2 = 3 hoặc 11 theo biểu diễn bit; O.values = l_i.values ∪ l_j.values = *2 ∪ 2 = 22*, và $O.Obidset = l_i.Obidset \cap l_j.Obidset$ = {1} ∩ {2} = {∅}. Vì O.count[O.pos] = 0 < *MinSup*, O không được thêm vào P_i.

- Tương tự $\begin{pmatrix}2\times6\\3(\underline{1},0)\end{pmatrix}, \begin{pmatrix}2\times3\\4(\underline{1},0)\end{pmatrix}$ *Obidset* giao nhau ={∅} do đó không thêm vào P_i.

- Xét nút $l_j = \begin{pmatrix}4\times2\\1(0,\underline{1})\end{pmatrix}$: Vì hai nút này khác thuộc tính nhau, nên ba yếu tố được tính lại như sau O.att = l_i.att ∪ l_j.att = 1 | 4 = 5 hoặc 101 theo biểu diễn bit; O.values = l_i.values ∪ l_j.values = *2 ∪ 2 = 22*, và $O.Obidset = l_i.Obidset \cap l_j.Obidset$ = {1} ∩ {1} = {1}. Bởi vì $|l_i.Obidset| = |O.Obidset|$, thuật toán sẽ chép thông tin từ l_i xuống *O(theo định lý 2.2)*. Điều đó có nghĩa rằng O.count = l_i.count = (0,1) và O.pos =2. Vì O.count[O.pos] = 1 ≥ *MinSup*, O được thêm vào $P_i \Rightarrow P_i = \left\{\begin{pmatrix}3\times21\\1(0,\underline{1})\end{pmatrix},\begin{pmatrix}5\times22\\1(0,\underline{1})\end{pmatrix}\right\}$

- Xét $\begin{pmatrix}4\times4\\2(0,\underline{1})\end{pmatrix}$ $\begin{pmatrix}4\times5\\34(2,0)\end{pmatrix}$ *Obidset* giao nhau ={∅} do đó không thêm vào P_i

- Xét nút $l_j = \begin{pmatrix}8\times2\\1(0,\underline{1})\end{pmatrix}$: Vì hai nút này khác thuộc tính nhau, nên ba yếu tố được tính lại như sau O.att = l_i.att ∪ l_j.att = 1 | 8 = 9 hoặc 1001 theo biểu diễn bit; O.values = l_i.values ∪ l_j.values = *2 ∪ 2 = 22*, và $O.Obidset = l_i.Obidset \cap l_j.Obidset$ = {1} ∩ {1} = {1 Bởi vì $|l_i.Obidset| = |O.Obidset|$, thuật toán sẽ chép thông tin từ l_i xuống O. Điều đó có nghĩa rằng O.count = l_i.count = (0,1) và O.pos = 2. Vì O.count[O.pos] = 1 ≥ *MinSup*, O được thêm vào $P_i \Rightarrow P_i = \left\{\begin{pmatrix}3\times21\\1(0,\underline{1})\end{pmatrix},\begin{pmatrix}5\times22\\1(0,\underline{1})\end{pmatrix},\begin{pmatrix}9\times22\\1(0,\underline{1})\end{pmatrix}\right\}$

- Xét $\begin{pmatrix}8\times3\\24(\underline{1},1)\end{pmatrix}$ $\begin{pmatrix}8\times4\\3(\underline{1},0)\end{pmatrix}$ *Obidset* giao nhau ={∅} do đó không thêm vào P_i

- Sau khi P_i được tạo ra, thuật toán CAR-Miner được gọi đệ quy với tham số P_i, *MinSup*, và *MinConf* để tạo ra các nút con của P_i. Xét việc xử lý để tạo ra các nút con của nút $l_i = \begin{pmatrix}3\times21\\1(0,\underline{1})\end{pmatrix}$:

- Xét nút $l_j = \begin{pmatrix} 5\times 22 \\ 1(0,\underline{1}) \end{pmatrix}$: Vì hai nút này khác thuộc tính nhau, nên ba yếu tố được tính lại như sau O.att = l_i.att ∪ l_j.att = 3 | 5 = 7 hoặc 11 theo biểu diễn bit; O.values = l_i.values ∪ l_j.values = *21* ∪ 22 = *212*, và O.*Obidset* = l_i.*Obidset* ∩ l_j.*Obidset* = {1} ∩ {1} = {1} = l_j.*Obidset*. Thuật toán sẽ chép thông tin từ l_j xuống O, điều đó có nghĩa rằng O.count = l_j.count = (0,1) và O.pos = 2. Vì O.count[O.pos] = 1 > *MinSup*, O được thêm vào $P_{i'}$ $\Rightarrow P_{i'} = \left\{\begin{pmatrix} 7\times 212 \\ 1(0,\underline{1}) \end{pmatrix}\right\}$
- Tương tự cho nút $l_j = \begin{pmatrix} 9\times 22 \\ 1(0,\underline{1}) \end{pmatrix}$, ta có kết quả $P_{i'} = \left\{\begin{pmatrix} 7\times 212 \\ 1(0,\underline{1}) \end{pmatrix}, \begin{pmatrix} 11\times 212 \\ 1(0,\underline{1}) \end{pmatrix}\right\}$
- Sau khi P_i được tạo ra, thuật toán CAR-Miner được gọi đệ quy với tham số P_i, *MinSup*, và *MinConf* để tạo ra các nút con của P_i. Xét việc xử lý để tạo ra các nút con của nút $l_{i=} \begin{pmatrix} 7\times 212 \\ 1(0,\underline{1}) \end{pmatrix}$. Ta tính được $P_{i''} = \left\{\begin{pmatrix} 15\times 2122 \\ 1(0,\underline{1}) \end{pmatrix}\right\}$
- Tương tự như trên để xét tiếp việc xử lý để tạo ra các nút con của nút $l_i = \begin{pmatrix} 5\times 22 \\ 1(0,\underline{1}) \end{pmatrix}$ và $\begin{pmatrix} 9\times 22 \\ 1(0,\underline{1}) \end{pmatrix}$

Tương tự cho đến khi thuật toán dừng (không còn nút nào được sinh ra).

B. Giai đoạn kiểm tra

Dựa vào các luật đã khai thác được trong giai đoạn 1, chúng tôi tiến hành thử nghiệm trên dữ liệu kiểm tra. Các bước cụ thể được trình bày trong Hình 2.

Định nghĩa 5: Cho hai luật r_i và r_j, (kí hiệu $r_i \succ r_j$) r_i có thứ bậc lớn hơn r_j nếu:

1. Độ tin cậy của r_i lớn hơn r_j, hoặc
2. Độ tin cậy của chúng bằng nhau nhưng độ phổ biến của r_i lớn hơn r_j, hoặc:
3. Cả độ tin cậy và độ phổ biến như nhau nhưng r_i được tạo ra trước r_j.

Bước 1: Sắp xếp các luật theo chiều giảm dần của thứ bậc (Theo định nghĩa 5).

Bước 2: Với mỗi dòng dữ liệu trong tập kiểm tra, xét lần lượt với tập luật đã được sắp xếp từ trên xuống, tìm luật đầu tiên chứa vế trái thỏa mãn điều kiện của dòng dữ liệu và vế phải của luật chính là kết quả dự đoán lớp của mẫu này.

Hình 2. Các bước để dự đoán lớp của các mẫu thuộc dữ liệu kiểm tra

V. KẾT QUẢ THỰC NGHIỆM

A. Cơ sở dữ liệu và môi trường thực nghiệm

Các thuật toán được sử dụng trong phần thực nghiệm đã được cài đặt trên máy tính chạy trên C# 2012 với cấu hình máy tính cá nhân như sau: Intel Core i3-350 2.26GHz, 4GB RAM, 320GB. Các CSDL thực nghiệm trong được lấy từ website UCI http://mlearn.ics.uci.edu.

Bảng 4. CSDL Dữ liệu thực nghiệm

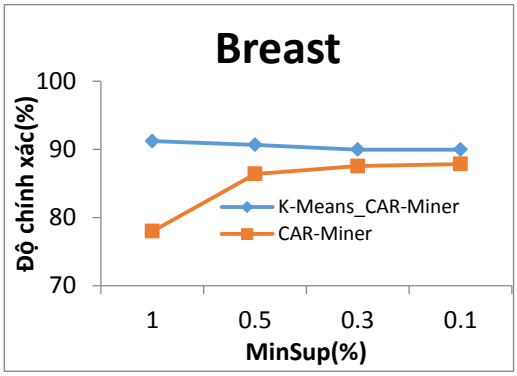
Tập dữ liệu	Số thuộc tính	Số lớp	Số mẫu	Mô tả
Breast	10	2	699	Lớp 0: 458 (65.5%) Lớp 1: 241 (34.5%)
Geman	21	2	1000	Lớp 0: 700 (70%) Lớp 1: 300 (30%)
Iris	4	3	150	Lớp 0: 50 (33.33%) Lớp 1: 50 (33.33%) Lớp 2: 50 (33.33%)

B. Kết quả thực nghiệm

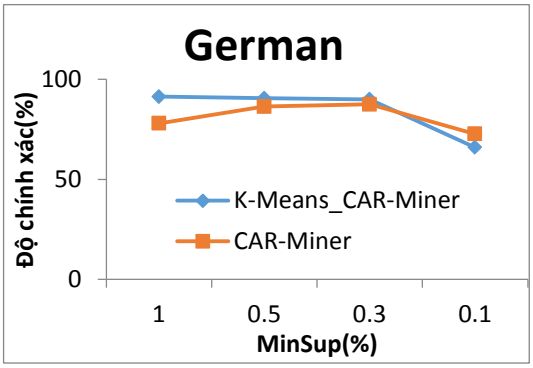
Kết quả thực nghiệm được đánh giá trên 3 tập dữ liệu từ bảng 4. Kết quả so sánh độ chính xác, giữa việc dùng CAR-Miner và K-means-CAR-Miner.

Chúng tôi so sánh dữ liệu thực nghiệm trên cả 2 thuật toán sử dụng độ tin cậy cố định *MinConf* = 60% cho tất cả các lần thực nghiệm và độ hỗ trợ thay đổi lần lượt 10%, 5%, 3%, 1%.

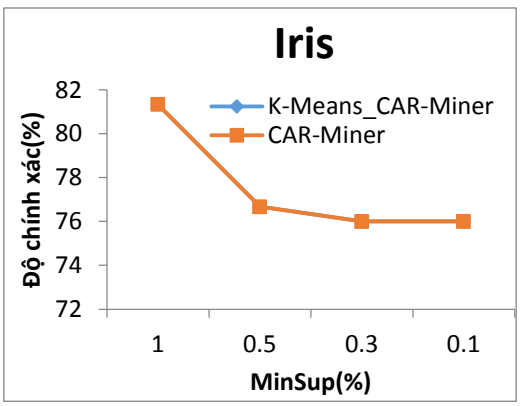
Độ chính xác phân lớp trên các CSDL ở bảng 4. Kết quả thực nghiệm so sánh độ chính xác giữa hai phương pháp K-means-CAR-Miner và CAR-Miner được trình bày trong các hình từ 3 đến 5.



Hình 3. So sánh độ chính xác phân lớp giữa K-means-CAR-Miner và CAR-Miner cho CSDL Breast



Hình 4. So sánh độ chính xác phân lớp giữa K-means-CAR-Miner và CAR-Miner cho CSDL German



Hình 5. So sánh độ chính xác phân lớp giữa K-means-CAR-Miner và CAR-Miner cho CSDL Iris

Các kết quả từ hình 3 đến hình 5 cho thấy đối với các CSDL mất cân bằng về lớp như Breast và German, K-means-CAR-Miner có độ chính xác cao hơn CAR-Miner, đặc biệt là đối với các ngưỡng *MinSup* lớn. Đối với các CSDL không mất cân bằng về lớp chẳng hạn CSDL Iris thì hai phương pháp này có độ chính xác như nhau.

VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong bài báo này chúng tôi đã đề xuất một phương pháp tích hợp giữa gom nhóm và phân lớp để giải quyết bài toán phân lớp trên CSDL mất cân bằng về lớp. Đối với các lớp có số mẫu lớn, chúng tôi tiến hành gom chúng thành k nhóm (với k là số mẫu của nhóm ít nhất), mỗi nhóm chọn một mẫu đại diện (trong bài báo này, chúng tôi chọn mẫu gần trọng tâm nhất). Sau đó, CAR-Miner được sử dụng trên CSDL đã được lấy mẫu nhờ gom nhóm để khai thác luật phân lớp kết hợp dùng cho dự đoán lớp của mẫu mới. Kết quả thực nghiệm bước đầu cho thấy phương pháp của chúng tôi cho độ chính xác cao hơn so với phương pháp không gom nhóm (sử dụng CAR-Miner trên CSDL gốc).

Trong tương lai, chúng tôi sẽ tiếp tục thử nghiệm trên nhiều CSDL hơn để đánh giá khả năng ứng dụng của phương pháp đề xuất. Ngoài ra, phương pháp này cũng sẽ được áp dụng vào các loại phân lớp khác như cây quyết định, ILA, SVM, v.v.

Lời cảm ơn: Nghiên cứu này được tài trợ bởi Quỹ Phát triển khoa học và công nghệ NTTU trong đề tài mã số 2016.02.06.

TÀI LIỆU THAM KHẢO

[1] Y. W. C. Chien, Y. L. Chen (2010), Mining associative classification rules with stock trading data – A GA-based method. Knowledge-Based Systems, vol.23, no.6, pp. 605-614.

[2] B. Liu, W. Hsu, Y. Ma (1998). Integrating classification and association rule mining. In Proc. of the 4th International Conference on Knowledge Discovery and Data Mining, New York, USA, pp. 80-86.

[3] W. Li, J. Han, J. Pei (2001), CMAR: Accurate and efficient classification based on multiple class-association rules, 1st IEEE international conference on Data mining, pp. 369–376.

[4] J. B. MacQueen (1967). Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley, California, vol.1, pp. 281-297.

[5] D. Nguyen, L. T. T. Nguyen, B. Vo, T. P. Hong (2015). A novel method for constrained Class-association rule mining. Information Sciences, vol. 320, pp. 107-125.

[6] L. T. T. Nguyen, B. Vo, T. P. Hong, H. C. Thanh (2013). CAR-Miner: An efficient algorithm for mining class-association rules. Expert Systems with Applications, vol.40, no.6, pp. 2305-2311.

[7] L. T. T. Nguyen, N. T. Nguyen (2015). An improved algorithm for mining class association rules using the difference of Obidsets. Expert Systems with Applications, vol.42, no.9, pp. 4361-4369.

[8] R. Quinlan (1992), C4.5: programs for machine learning, Machine Learning, vol.16, pp. 235-240.

[9] F. A. Thabtah, P. Cowling, Y. Peng (2004), MMAC: A new multi-class, multi-label associative classification approach, the 4th IEEE International Conference on Data mining, pp. 217-224.

[10] F. Thabtah, P. Cowling, Y. Peng (2005), MCAR: Multi-class classification based on association rule, 3rd ACS/IEEE international conference on computer systems and applications, pp. 33–39.

[11] M. R. Tolun, S. M. Abu-Soud (1998), ILA: an inductive learning algorithm for rule extraction, Expert Systems with Applications, vol.14, no.3, pp. 361– 370.

[12] B. Vo, B. Le (2008), A novel classification algorithm based on association rule mining. In Proc. of the 2008 Pacific Rim Knowledge Acquisition Workshop (Held with PRICAI'08), LNAI 5465, vol. 5465, pp. 61-75.

[13] J. Wu (2012), Advances in K-means clustering: a data mining thinking. Springer Sceience & Business Media, pp. 17-35.

[14] X. Wu et al. (2008), Top 10 algorithms in data mining. Knowledge and Information Systems, vol.14, no.1, pp. 1-37.