Hướng tiếp cận hiệu quả cho việc phân lớp dựa trên luật kết hợp

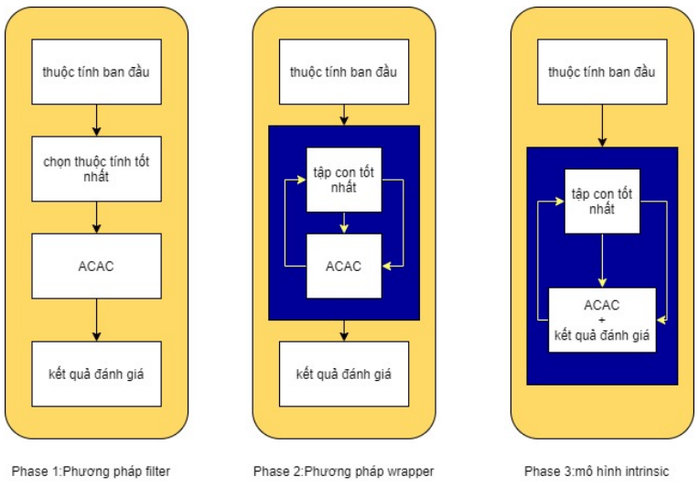
Nguyễn Quốc Huy [1], Trần Anh Tuấn[1], Đỗ Như Tài[2], Nguyễn Thị Ngọc Thanh[3]
[1] Trường Đại học Sài Gòn, [2]Đại học Kinh Tế Tp.HCM, [3]Trường Đại học Mở Thành phố Hồ Chí Minh
nqhuy@sgu.edu.vn, dasanbob22122002@gmail.com, taidn@ueh.edu.vn, thanh.ntn@ou.edu.vn

TÓM TẮT—*Phân lớp dựa trên luật kết hợp là một cách tiếp cận thú vị trong khai thác dữ liệu để tạo ra các hệ thống dự đoán chính xác và dễ dàng, dễ giải thích hơn. Cách tiếp cận này thường được xây dựng dựa trên cả kỹ thuật khai thác luật kết hợp và phân lớp, để tìm ra tập luật gọi là luật kết hợp dùng cho phân lớp (CAR) thuộc tính nhãn. Thuật toán ACAC thuộc họ bài toán CPAR nhưng độ chính xác phân loại còn thấp trên dữ liệu lớn. Bài báo này đề xuất ba bước tìm tập thuộc tính tối ưu để nâng cao đáng kể hiệu suất của thuật toán ACAC về mặt thời gian cũng như độ chính xác trên dữ liệu lớn. Kết quả thực nghiệm cho thấy việc chọn tập dữ liệu tối ưu giúp cho thuật toán ACAC trở nên hữu ích hơn trên dữ liệu lớn, đặc biệt chi phí tối ưu tập dữ liệu theo ba bước này hoàn toàn hợp lý trong thực tế.*

Từ khóa— *CAR; CPAR; lựa chọn đặc trưng; entropy; leo đồi, dữ liệu lớn.*

I. GIỚI THIỆU

Bộ phân loại kết hợp là một mô hình học có giám sát dùng luật kết hợp để gán nhãn. Mô hình có được sau khi huấn luyện bao gồm các luật kết hợp được tạo ra dùng để gán nhãn dữ liệu mới. Vì vậy mô hình cũng có thể được xem là một danh sách các mệnh đề "if-then": nếu dòng dữ liệu mới đáp ứng các tiêu chí nhất định (chứa đủ các thuộc tính phân lớp bên trái), thì dòng dữ liệu này sẽ được phân lớp theo đúng giá trị bên phải của luật [7, 8]. Bộ phân lớp kết hợp kế thừa độ Hỗ trợ và độ Tin cậy để lọc bớt các luật không cần thiết từ dữ liệu huấn luyện. CPAR là một loại phân lớp kết hợp dựa trên các luật kết hợp dự đoán, phương pháp này kết hợp các ưu điểm của phân lớp kết hợp và phân lớp dựa trên quy tắc truyền thống khi vừa sinh luật trực tiếp trong giai đoạn huấn luyện vừa kiểm tra để tránh bỏ sót các luật quan trọng. Vào năm 2003, Omiecinski đưa ra cách tính All-confidence rất hiệu quả trong khai thác các mẫu kết hợp. Dựa vào độ đo này, nhóm nghiên cứu của Zaixiang Huang đề xuất thuật toán ACAC[1] sử dụng cả độ đo All-confidence và độ hỗ trợ để khai thác những tập không những phổ biến mà các tập mục con còn có mối quan hệ hai chiều, tính chất này rất cần thiết cho việc phân lớp dựa trên luật kết hợp, và đây cũng là một thuật toán tiêu biểu theo phương pháp CPAR[4, 5, 6, 9]. Tuy nhiên thuật toán này vẫn gặp nhiều hạn chế về mặt thời gian và độ chính xác phân lớp khi số thuộc tính dữ liệu lớn. Bài báo này đề xuất phương pháp loại bỏ những thuộc tính không quan trọng giúp cải tiến phương pháp ACAC theo ba giai đoạn như trong hình 1. Việc cải tiến này giúp tăng tốc độ xử lý đồng thời tăng độ chính xác lên đáng kể.



Hình 1. Ba giai đoạn chọn tập thuộc tính tối ưu

Giai đoạn 1 chọn tập thuộc tính ban đầu theo độ đo Entropy. Ở giai đoạn này, các thuộc tính bị loại bỏ dựa trên mối quan hệ Entropy của thuộc tính đó với thuộc tính nhãn. độ đo Entropy thể hiện mối tương quan để kiểm tra xem các thuộc tính có đóng vai trò phân lớp cao hay thấp với thuộc tính nhãn và quyết định loại bỏ hay không các thuộc tính tương ứng.

Giai đoạn 2 tìm tập thuộc tính tối ưu xuất phát từ tập thuộc tính trong giai đoạn 1. Tập thuộc tính được chia thành các tập hợp con và huấn luyện mô hình dựa trên những tập con dữ liệu này. Dựa trên độ chính xác của mô hình, ta có thể thêm và bớt các thuộc tính trên tập thuộc tính có được trong giai đoạn 1 và huấn luyện lại mô hình. Phương pháp này hình thành các tập hợp con bằng cách sử dụng cách tiếp cận tham lam và đánh giá tính chính xác của tất cả các tổ hợp xuất phát từ tập thuộc tính trong giai đoạn 1 thay vì thử hết trên tất cả các tổ hợp thuộc tính có thể có.

Giai đoạn 3 mở rộng tập thuộc tính tối ưu để tăng độ chính xác phân lớp. Mở rộng bằng tập thuộc tính tối ưu trong giai đoạn 2 kết hợp với từng thuộc tính còn lại để tìm tập tối ưu mới có độ chính xác cao hơn.

Đóng góp chính của bài báo tập trung vào phần 2.B và phần 3. Nội dung bài báo được sắp xếp như sau: phần 2.A sẽ đề cập lại việc phân lớp dựa trên luật kết hợp và tính chất độ đo Entropy, phần 2.B trình bày phương pháp ba giai đoạn, kết quả việc thực nghiệm và minh chứng xác định tính hiệu quả được mô tả trong phần 3, và kết luận việc nghiên cứu được trình bày trong phần 4.

II. PHÂN LỚP THEO CÁC THUỘC TÍNH CÓ LIÊN QUAN

Theo kiểu phân lớp này, dữ liệu huấn luyện T có m thuộc tính A1, A2, …, Am, và thuộc tính phân lớp. Các giá trị thuộc tính có thể liên tục hay rời rạc. Đối với thuộc tính có giá trị rời rạc, ta có thể dùng các số nguyên dương liên tiếp để biểu diễn. Đối với thuộc tính có giá trị liên tục, ta rời rạc hóa các giá trị này trước và dùng các số nguyên dương liên tiếp để biểu diễn các giá trị rời rạc này. Một tập mục $X = a_{i_1},...,a_{i_j},...,a_{i_k}$ là tập các giá trị của các thuộc tính khác nhau. Như vậy tập mục 1-item được xác định bằng một giá trị của một thuộc tính, và chính là a_{i_j} . Số dòng dự liệu trong T khớp với tập mục X được gọi là độ hỗ trợ của X, biểu diễn là sup(X). All-confidence[10] của tập mục $X = a_{i_1},...,a_{i_j},...,a_{i_k}$ được định nghĩa như sau: $allconf(X) = \frac{\sup(X)}{\max(\sup(a_{i_1}),...,\sup(a_{i_k}))}$

Công thức này xác định độ tin cậy tối thiểu của tất cả những luật được sinh ra từ một tập mục X.

Cho tập dữ liệu huấn luyện T, gọi c là nhãn lớp. Một ruleitem có dạng {condset, c}, với condset là một tập mục. Mỗi ruleitem biểu diễn cơ bản một luật: $condset \rightarrow c$. Ruleitem mà condset có k item thì gọi là k-ruleitem. Độ hỗ trợ của condset (gọi là consup) là số lần xuất hiện của condset trong T. Độ hỗ trợ của một luật (gọi là rulesup) là số lần xuất hiện của cả condset và c trong T.

A. Phương pháp truyền thống

Hầu hết các thuật toán phân lớp dựa trên luật kết hợp có ba giai đoạn:

- Sinh luật từ tập huấn luyện, luật kết hợp có dạng $condset \rightarrow c$. Để bớt tốn chi phí, ta dùng cả độ hỗ trợ và độ all-confidence để khai thác những tập không những phổ biến mà các tập mụzc con còn có mối quan hệ hai chiều.
- Bỏ đi những luật có thể gây ra quá khớp hoặc dư thừa.
- Phân lớp dữ liệu mới.

Để cho dễ hình dung, ta xem ví dụ trên tập T cho trước như bảng 1. Trong ví dụ này, ngưỡng của độ hỗ trợ là 2, ngưỡng all-confidence là 50%, và ngưỡng độ tin cậy là 100%.

Bảng 1. Dữ liệu huấn luyện T

A	B	C	D	class
32	55	80	83	90
33	52	80	85	89
33	52	80	85	89
33	55	79	82	90
34	55	79	82	89
34	55	77	82	89
32	55	80	88	90
33	55	79	82	90

Trong bảng này, ta chọn các ruleitem thỏa ngưỡng condsup và all-confidence và đưa vào trong tập ứng viên 1-ruleitem F1. Từ tập F1, ta chọn những ruleitem nào thỏa ngưỡng confidence và đưa vào tập R1, rồi xóa tập F1 đi.

Bảng 2. Tập ứng viên 1-ruleitem F1

itemset	class	allconf	conf	condsup	rulesup
33	89	100	50	4	2
33	90	100	50	4	2
55	89	100	67	6	2
55	90	100	33	6	4
79	89	100	33	3	1
79	90	100	67	3	2
80	89	100	50	4	2
80	90	100	50	4	2
82	89	100	50	4	2

82	90	100	50	4	2

Bảng 3. Tập luật R1

itemset	class	allconf	conf	condsup	rulesup
32	90	100	100	2	2
34	89	100	100	2	2
52	89	100	100	2	2
85	89	100	100	2	2

Các ứng viên 2-ruleitem được kết hợp từ tập F1, giả sử trong F 1 có 2 luật:

Ruleitem{(55), 90} có rulesup là 4

Ruleitem{(79), 90} có rulesup là 2

Thì luật sinh ra sẽ là {(55, 79), 90} có rulesup là 2, all-conf = 2/max(4,2) = 50%. Kết quả các 2-ruleitem được đưa vào bảng F2, và bảng R2 là bảng chứa những 2-ruleitem nào thỏa ngưỡng confidence.

Bảng 4. Tập ứng viên 2-ruleitem F2

itemset	class	allconf	conf	condsup	rulesup
55 79	90	50	67	3	2
55 82	89	50	50	4	2
55 82	90	100	50	4	2
79 82	90	100	67	3	2

Bảng 5. Tập luật R2

itemset	class	allconf	conf	condsup	rulesup
33 55	90	100	100	2	2
33 79	90	100	100	2	2
33 80	89	100	100	2	2
33 82	90	100	100	2	2
55 80	90	100	100	2	2

Quá trình này được lặp đi lặp lại cho đến khi tập ứng viên k-ruleitem rỗng. Thuật toán 1 có tên là ACAC-RG, trong đó tập ứng viên k-itemset Ck, tập k-itemset phổ biến Fk. R là tập các luật. Dòng 1-2 tính condsup và rulesup của từng item đồng thời, rồi hàm ruleSelection được gọi tại mỗi lần lặp (dòng 6). Trong mỗi lần lặp, thuật toán thực hiện 3 thao tác chính. Đầu tiên, các itemset phổ biến Fk-1 được tìm thấy trong lần lặp (k-1) dùng để sinh các itemset ứng viên Ck theo như hàm candidateGen (dòng 4), hàm này tương tự với hàm Apriori-Gen. Tiếp theo, hàm supportCount quét tập dữ liệu và tính condsup và rulesup của các ứng viên trong Ck (dòng 5). Cuối cùng thì thực hiện hàm ruleSelection.

Hàm ruleSelection tính all-confidence của mỗi itemset ứng viên trong mỗi lớp và độ tin cậy của mỗi luật ứng viên (dòng 1 đến 3). Nếu các luật ứng viên thỏa ngưỡng độ hỗ trợ, ngưỡng all-confidence, và ngưỡng độ tin cậy, thì luật này sẽ đưa vào tập R. Nếu luật ứng viên chỉ thỏa ngưỡng độ hỗ trợ, ngưỡng all-confidence, thì đưa nó vào tập Fk (dòng 4 đến 10). Khi một luật ứng viên được chọn, condset của nó sẽ không được mở rộng trong vòng lặp con tiếp theo.

Algorithm ACAC-RG(T)

Input: tập dữ liệu huấn luyện, minSup, minConf, minAllConf
Output: R: tập luật cho mô hình dự đoán nhãn
1: C_1 ← init − pass(T);
2: ruleSelection(C_1, F_1, R);
3: ***for***(k=2; $F_{k-1} \neq \emptyset$; k++)
4: Ck←candidateGen(F_{k-1});
5: supportCount(C_k);
6: ruleSelection(C_k,F_k,R);
7: ***end for***
8: ***return*** R;

Algorithm ACPredict(R,D)

Input:
- R là tập luật được khai thác,
- Dnew (dữ liệu mới chưa có nhãn),
Output: D' (dữ liệu được gán nhãn)
1:D' = empty
2:***foreach*** t_i in D:
3: v_1= S(R,t_i,Yes),
4: v_2= S(R,t_i,No),
5: ***if***(v1>v2) ***then***:
6: t_i [Label] ← Yes;//gán nhãn là Yes
7: ***else***:
8: t_i [Label] ← No;//gán nhãn là No
9: D' = D' + t_i;
10:***return*** D';

Sau khi thực hiện thuật toán ACAC-RG, ta thu được tập luật R. Dựa vào tập luật R, các bước phân lớp như sau:

Tính giá trị Info(entropy) cho từng thuộc tính theo công thức như sau:

$$Info(X)= \frac{1}{\log 2k} \sum_{i=k}^{n} P(C_i|X)\log_2 P(C_i|X)$$

trong đó k là số lượng lớp, $P(C_i|X)$ là khả năng phù hợp giữa C_i và X.

Đại lượng entropy ở trên sẽ được sử dụng để tính S(r), 1 đại lượng đo lường độ phù hợp của các luật theo công thức sau:

$$S(r)=0.9*(1-\frac{\Sigma sup(Xi)*Info(Xi)}{\Sigma sup(Xi)})+0.1*\frac{n}{ntot}$$

Trong đó Xi là tập luật con của r_i ($r_i \in R$), n_{tot} là tổng số luật phù hợp với đối tượng mới, n là số lượng luật trong tập R.

Sau khi tính toán được giá trị S(r), ACAC dùng nhóm có S(r) cao nhất thể chọn nhãn cho dữ liệu mới.

Quá trình phân lớp theo công thức S(r) được mô tả trong thuật toán ACPredict.

B. Phương pháp cải tiến

Với tập dữ liệu lớn và nhiều thuộc tính, chi phí để chọn ra một tập thuộc tính tối ưu là rất lớn. Phương pháp bài báo đề xuất kết hợp tính chất của độ đo Entropy và phương pháp Leo đồi để lựa chọn đặc trưng có ba giai đoạn. Giai đoạn 1 chủ yếu là lọc thuộc tính không quan trọng theo độ đo Entropy (giá trị Entropy giữa một thuộc tính và thuộc tính phân lớp) và phương pháp lựa chọn đặc trưng theo hình thức Leo đồi. Độ chính xác của mô hình sau khi loại bỏ thuộc tính được kiểm nghiệm lại theo phương pháp kiểm tra chéo. Tập thuộc tính được chọn sẽ là tập thuộc tính tối ưu cho sự phân lớp. Giá trị Entropy nằm trong khoảng [0,1], những thuộc tính có giá trị Entropy lớn thì sẽ không có đóng góp nhiều thông tin cho việc phân lớp thì nên được loại bỏ. Trong những tập dữ liệu có ít thông tin, ta có thể tính tất cả các entropy của các thuộc tính và sắp xếp thứ tự từ cao đến thấp. Sau đó, tuần tự kiểm tra việc loại bỏ từng thuộc tính theo độ ưu tiên từ cao đến thấp và đánh giá độ chính xác của tập thuộc tính còn lại. Nếu độ chính xác của tập thuộc tính còn lại bằng hoặc lớn hơn giá trị cũ (khi chưa bỏ thuộc tính) thì thuộc tính được bỏ thật sự là nên bỏ. Vì đây là hướng tiếp cận từ dưới lên (bottom up) nên chi phí lại bỏ trong tập dữ liệu lớn ban đầu là không khả thi. Bài báo đề xuất ngưỡng Entropy ban đầu, thay vì phải thử Entropy theo thứ tự từ cao xuống thấp như trong thuật toán Phase 1.

Trong thuật toán Phase 1, dòng 13 chính là thuật toán ACAC chưa cải tiến, dòng 14 tính độ chính xác của mô hình có được từ dòng 13. Đoạn mã từ dòng 8 đến dòng 18 lặp đi lặp lại công việc cải tiến cho đến khi độ chính xác của mô hình đạt đỉnh. Trong mỗi lần lặp, dựa trên độ entropy lân cận entropy hiện hành (dòng 11, 12). Dựa vào độ entropy e', thuật toán xác định tập thuộc tính De bao gồm những thuộc tính có độ entropy so với lớp nhãn nhỏ hơn e'. Nói theo cách khác, trong lần lặp này thuật toán đã thử loại bỏ thuộc tính được cho là thừa ra khỏi tập huấn luyện. Dòng 15 kiểm tra xem việc loại bỏ liệu có đúng đắn hay không, nếu đúng thì cập nhật lại giá trị tốt hơn. Sau khi thoát khỏi vòng lặp, thuật toán tìm được tập thuộc tính tương đối tốt như dòng số 18. Trong dòng 2, D là tập dữ liệu được chia ra thành 2 nhóm Dtrain và Dtest theo tỷ lệ 80/20. Dựa vào Dtrain để tìm tập luật phân lớp, và Dtest dùng để kiểm thử độ chính xác của tập luật phân lớp.

Algorithm Phase1(FSAC)

1: minSup, minConf, minAllConf,
2: D: dataset (D_{train}, D_{test})
3: e := entropy threshold
4: D_e := {các thuộc tính D_{train} a_i} với entropy(a_i, label) ≤ e
5: $neighbor = [e, e \pm \Delta]$
6: e' := random(neighbor)
7: $D_{e'}$:= {các thuộc tính D_{train} a_i} với entropy(a_i, label) ≤ e'
8: **do**
9: $model_e$:= ACAC-RG(D_e);
10: $accuracy_e$:= $model_e$.predict(D_{test});
11: $neighbor = [e, e \pm \Delta]$
12: e' := random(neighbor);
13: $model_{e'}$:= ACAC-RG(D_e);
14: $accuracy_e$':= modele'.predict(D_{test});
15: ***if*** ($accuracy_e \leq accuracy_{e'}$) ***then*:**
16: e:=e';
17: ***end if***
18: ***while*** ($accuracy_{e'} \leq accuracy_e$);
19: ***return*** $model_e$;

Algorithm Phase 2 (Hill Climbing)

Input:
- T là tập dataset
- best_Attributes (tập thuộc tính ban đầu)
- sub_Attributes (tập thuộc tính dùng để thay vào best_Attributes)
- all_Attributes (tất cả thuộc tính của dataset)
- initial_Validation (chỉ số đánh giá ban đầu)

Output: tập thuộc tính có chỉ số đánh giá tốt nhất

1: delta := Δ;
2: ***for***(i=0;i ≤ n; i++)
3: new_Attributes:=GenerateAttributes(best_Attributes, sub_Attributes, delta);
4: validation:=ACAC(new_Attributes);
5: ***if*** (validation > initial_Validation) *then*:
6: initial_Validation := validation;
7: i:=0;
8: best_Attributes := new _Attributes
9: sub_Attributes:= all_Attributes- best_Attributes
10: ***end if***
11: ***end for***
12: ***return*** best_Attributes;

Kết quả của việc thực hiện Phase 1, ta thu được tập thuộc tính ban đầu tương đối tốt nhưng chưa thực sự tối ưu vì hạn chế của phương pháp chọn dữ liệu theo hướng lọc (Filter method) từng thuộc tính. Giai đoạn 2 mô phỏng phương pháp leo đồi với nghiệm đầu tiên là tập thuộc tính ban đầu thu được từ giai đoạn 1. Giai đoạn này, thuộc tính đang được

chia thành 2 nhóm thuộc tính (best_Attributes và Sub_attributes). Tập thuộc tính best_Attributes chứa các thuộc tính thu được từ giai đoạn 1, tạm cho là tập thuộc tính tốt. Tập thuộc tính sub_Attributes chứa các thuộc tính còn lại. Thuật toán Phase 2 thực hiện công việc tìm xem có tập nào tốt hơn tập best_Attributes hiện hành hay không, nếu có thì cập nhật lại tập best_Attributes, đây là công việc từ dòng 3 đến dòng 10 trong thuật toán Phase 2. Công việc này lặp đi lặp lại cho đến khi tìm được tập tốt nhất với số lượng thuộc tính vẫn không thay đổi so với tập thuộc tính thu được từ giai đoạn 1. Trong mỗi lần lặp như vậy là thử trên tập thuộc tính mới được sinh ra từ hàm *GenerateAttributes* trong dòng số 3, dựa vào giá trị *delta* để đưa ra số lượng *delta* thuộc tính ngẫu nhiên nào đó trong tập best_Attribute, và bổ sung số lượng *delta* thuộc tính ngẫu nhiên từ tập sub_Attributes vào tập best_Attributes. Nếu tập thuộc tính mới có kết quả tốt hơn tập thuộc tính best_Attributes hiện hành thì cập nhật lại (dòng 5 đến dòng 8 trong thuật toán Phase 2).

Algorithm Phase 3

Input:
- initial_Attributes (tập thuộc tính ban đầu của dataset)
- sub_Attributes (tập thuộc tính dùng để thay vào best_Attributes)
- T là tập dataset
- initial_Validation (chỉ số đánh giá ban đầu)

Output: tập thuộc tính có chỉ số đánh giá tốt nhất

```
1: best_Attributes :={};
2: for(i=0;i< sub_Attributes.length;i++) do:
3:          new_Attributes := initial _Attributes + sub_Attributes[i];
4:      validation := ACAC(new_Attributes);
5:      if (validation > initial_Validation) then:
6:          initial_Validation := validation
7:          best_ Attributes := new_Attributes;
8: return best_Attributes
```

Sau khi xong giai đoạn 2 (Phase 2), ta được tập thuộc tính tối ưu với số lượng thuộc tính bằng với tấp thuộc tính thu được trong giai đoạn 1. Giai đoạn 3 là giai đoạn mở rộng số lượng tập thuộc tính nhằm tìm giá trị tối ưu hơn xuất phát từ tập thuộc tính thu được trong giai đoạn 2. Dòng 2 đến dòng 7 trong thuật toán Phase 3 tiến hành việc thử tìm tập mở rộng tối ưu hơn bằng việc ghép tập thu được trong giai đoạn 3 ghép với từng thuộc tính còn lại trong tập sub_Attributes.

III. THỰC NGHIỆM

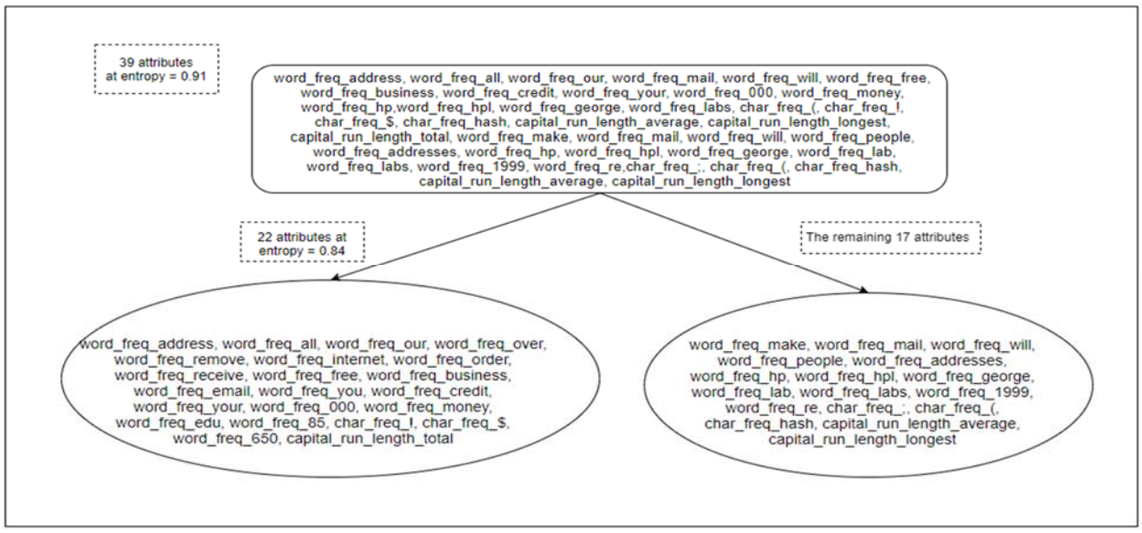
Môi trường thực nghiệm được xử lý tập trung trên máy tính có cấu hình Intel(R), Core(TM) i7-6820HQ CPU @ 2.70GHz (8 CPUs), ~2.7GHz và hệ điều hành Windows 10. Trong bài báo này thực nghiệm được thực hiện trên tập dữ liệu Spam Emails Dataset [3] một tập dự liệu khá phức tạp, dữ liệu có 4602 dòng và 58 cột. Trong đó, thuộc tính cuối là thuộc tính nhãn có tên là 'spam' để phân loại email có là spam hay ham (spam=1, ham=0). Dữ liệu này cũng được dùng trong thuật toán ACAC. Trong 57 thuộc tính còn lại có nhiều thuộc tính không có giá trị trong việc phân lớp cần phải loại bỏ. Để loại bỏ, bài báo dùng độ entropy để xác định khả năng đóng góp trong việc phân lớp của từng thuộc tính. Thuộc tính nào có giá trị entropy cao sẽ được loại bỏ khỏi tập dữ liệu, tập dữ liệu mới được thử lại theo phương pháp ACAC để tìm tập luật kết hợp 5 dùng cho phân lớp. Tập luật này sẽ được kiểm thử theo phương pháp kiểm tra chéo (80% - 5000 dòng dữ liệu dùng cho việc huấn luyện, 20% - 1001 dòng dữ liệu dùng cho việc kiểm thử). Nếu độ chính xác tăng lên thì việc loại bỏ thuộc tính là đúng, và việc này được thực hiện cho đến khi độ chính xác đạt giá trị cao nhất. Nói một cách khác, công việc sẽ dừng khi độ chính xác bị suy giảm.

Với 57 thuộc tính phân lớp, việc phân lớp dựa trên thuật toán ACAC[1] là 1 công việc có chi phí thực hiện (thời gian và không gian lưu trữ) rất cao và các chỉ số đánh giá chưa chắc đã ở một mức cao. Bởi điểm hạn chế đó, ta có thể sử dụng độ đo entropy để loại bỏ bớt số luật rời rạc nhằm giảm thời gian thực hiện việc phân lớp như thuật toán Phase 1 thực hiện.

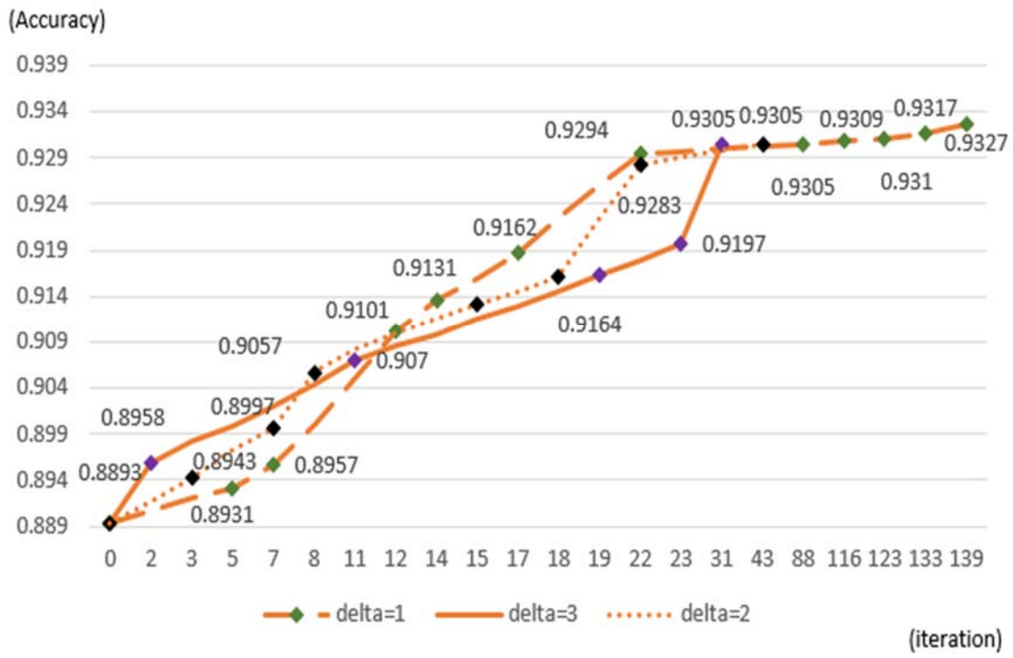
Bảng 6. Giá trị thực nghiệm của FSAC trên từng ngưỡng entropy

entropy	Attributes	Rules	Runtimes(ms)	Accuracy(%)
0.9	39	?	?	?
0.885	33	3108	7.78375E+13	94.2
0.88	31	2239	3.4375E+11	94.4
0.875	29	1612	11375000000	94.9
0.87	28	1254	261000000	94.06
0.865	27	989	51000000	93.04
0.86	26	692	10000000	92.4
0.85	24	338	2200000	90.99
0.84	22	109	35170	88.93
0.83	17	24	343	87.04
0.82	15	19	289	81.11
0.81	14	17	151	81.11
0.79	13	13	145	80.61

Để loại bỏ những thuộc tính không có vai trò phân lớp, thực nghiệm loại bỏ các thuộc tính có Entropy > 0.9. Qua đó số thuộc tính giảm từ 57 xuống còn 39 thuộc tính như trong bảng 6, thời gian xử lý và cải thiện các chỉ số đánh giá. Mặc dù đã loại bỏ 18 thuộc tính nhưng số thuộc tính vẫn là quá lớn cho việc tìm ra tập luật kết hợp phân lớp. Trong giai đoạn 1, thực nghiệm tìm ra tập thuộc tính ban đầu sao cho thời gian xử lý vẫn chấp nhận được và độ chính xác hợp lý. Và tập thuộc tính ban đầu tìm được từ thuật toán Phase 1 trong thực nghiệm này là 22 thuộc tính có độ Entropy ≤ 0.84, tập 17 thuộc tính còn lại được dùng trong phép thử tối ưu trong giai đoạn 2. Phase 1 dừng tại giá trị Entropy = 0.84 và đạt 22 thuộc tính vì cân bằng giữa hiệu suất và thời gian xử lý, khi giá trị Entropy = 0.85 lúc này số thuộc tính là 24 (xem bảng 6, cột 2) thì chi phí về thời gian thực hiện tăng lên đáng kể mặt dù độ chính xác vẫn còn tăng. Hình 2 mô tả kết quả sau khi thực hiện xong thuật toán Phase 1 là chia tập dữ liệu ban đầu thành 2 tập con. Tập best_Attributes (22 thuộc tính thu được từ thuật toán Phase 1) được cho là tập thuộc tính tương đối tốt nhưng chưa tối ưu và tập sub_Attributes (bao gồm các thuộc tính còn lại).



Hình 2. Tập thuộc tính dữ liệu sau khi thực hiện Phase 1.



Hình 3. Kết quả Phase 2 theo từng *delta*.

Sau khi chọn được tập thuộc tính ban đầu (best_Attributes) là tập 22 thuộc tính tại ngưỡng entropy = 0.84. bước tiếp theo đó chính là tìm ra tập 22 thuộc tính tối ưu hơn 22 thuộc tính ban đầu. Để thực hiện được điều đó, phần thực nghiệm sẽ được tiến hành chạy thuật toán Phase 2. Phần thực nghiệm này chọn giá trị delta lần lượt là 1, 2 và 3 với mỗi giá trị delta khác nhau sẽ cho những kết quả khác nhau như trong hình 3. Theo phương pháp leo đồi vẫn còn nhiều hạn

chế, chưa thể kết luận giá trị *delta* ảnh hưởng như thế nào đến việc tối ưu, nhưng trong thực nghiệm này giá trị *delta* = 1 mang lại kết quả tốt nhất với tập thuộc tính tối ưu có độ chính xác là 0.9327 so với các giá trị còn lại. Giá trị *delta* = 2 trong thực nghiệm này có kết quả kém nhất (tốn 88 lần lặp so với 31 lần lặp khi *delta* = 3) về mặt thời gian cũng như độ chính xác.

Qua giai đoạn 2, Ta đã chọn ra được tập luật gồm có 22 thuộc tính có độ chính xác = 0.9305 (tương đối tốt vì thời gian thực hiện của *delta* = 3 chỉ là 31 lần lặp so với thời gian *delta* = 1 mất 139 lần lặp) và 17 thuộc tính còn lại. Ở giai đoạn 3, thuật toán Phase 3 tìm ra tập 23 thuộc tính có độ chính xác cao hơn 0.9305. Để cụ thể hóa mục tiêu trên, phần thực nghiệm này sẽ kết hợp tập 22 thuộc tính và từng thuộc tính trong 17 thuộc tính còn lại. mỗi tập 23 thuộc tính sẽ cho những kết quả có thể khác nhau như trong bảng 7. Khi thực nghiệm Phase 3, ta có được 2 thuộc tính "word_freq_mail" và thuộc tính "char_freq_;" giúp cho tập thuộc tính tối ưu có kết quả tốt hơn.

Bảng 7. Kết quả sự kết hợp từng thuộc tính trong tập sub_Attributes với tập best_Attributes

Index	entropy	Tên thuộc tính	Time(ms)	rules	Accuracy(%)
1	0.30840	capital_run_length_average	3195068	640	0.9305
2	0.65802	capital_run_length_longest	2991905	640	0.9305
3	0.78659	word_freq_hp	26753570	604	0.9305
4	0.78937	char_freq_(	3680278	605	0.9305
5	0.83027	word_freq_george	31908135	604	0.9305
6	0.83569	word_freq_mail	54212858	957	**0.9349**
7	0.83673	word_freq_hpl	59496226	604	0.9305
8	0.83897	word_freq_will	23804363	917	0.9305
9	0.83950	char_freq_hash	14989621	861	0.9305
10	0.87196	word_freq_re	17217569	703	0.9305
11	0.87707	word_freq_people	25790388	700	0.9305
12	0.87805	word_freq_1999	20146323	604	0.9305
13	0.88005	word_freq_make	71690336	645	0.9305
14	0.88494	char_freq_;	19102413	1039	**0.9392**
15	0.88756	word_freq_addresses	22861710	725	0.9305
16	0.89806	word_freq_labs	32724505	604	0.9305
17	0.90796	word_freq_lab	28166730	604	0.9305

Có một điều thú vị là sau khi kết thúc giai đoạn 1, thuật toán Phase 1 dừng tại giá trị Entropy = 0.84. Trong bảng 7, có 9 thuộc tính đầu tiên có giá trị Entropy < 0.84 và 8 thuộc tính cuối cùng > 0.84. Trong 9 thuộc tính đầu tiên có thuộc tính "word_freq_mail" có giá trị Entropy = 0.83569 giúp cho tập thuộc tính tối ưu từ Phase 2 tăng độ chính xác phân lớp đến 0.9349. Trong 8 thuộc tính cuối cùng, có thuộc tính "char_freq_;" có giá trị Entropy = 0.88494 giúp cho tập thuộc tính tối ưu từ Phase 2 tăng độ chính xác phân lớp đến 0.9392. Đây cũng là điều hạn chế của phương pháp lọc đặc trưng (Filter Method) mà bài báo đã trình bày ở trên.

IV. KẾT LUẬN

Thuật toán ACAC chỉ hiệu quả trên tập dữ liệu nhỏ như Mushroom [2], đối với tập dữ liệu lớn và có nhiều thuộc tính như SpamMail [3] thuật toán bộc lộ hạn chế. Điều này làm cho thuật toán ACAC nói riêng và các thuật toán CAR nói chung không có tính thực tế khi phân lớp dữ liệu, trong khi các thuật toán CAR là loại thuộc toán phân lớp dễ giải thích (có tính explainable) lý do phân lớp cho người dùng cuối thấu hiểu nguyên nhân phân lớp.

Phương pháp ba giai đoạn trong bài báo đề cập là phương pháp lựa chọn đặc trưng theo hướng học có giám sát. Giai đoạn một thực hiệu việc lọc đặc trưng theo độ đo Entropy, giai đoạn hai thực hiện việc lựa chọn đặc trưng tối ưu theo nhóm nhỏ mô phỏng hình thức Leo đồi, Giai đoạn ba mở rộng tập tối ưu nhằm tìm bộ phân lớp có kết quả chính xác hơn. Cả ba giai đoạn đều có điểm mạnh và điểm hạn chế riêng. Tuy nhiên, theo phương pháp thực hiện linh hoạt như mô tả rõ trong thực nghiệm. Kết quả cuối cùng vẫn tìm ra tập thuộc tính tối ưu về mặt phân lớp cũng như thời gian xử lý chấp nhận được.

Phương pháp này có thể mở rộng để đạt kết quả cao hơn khi vượt qua hạn chế của Leo đồi bằng các phương pháp khác như Luyện thép, Di truyền, Bầy đàn. Bên cạnh đó ta cũng có thể tiến hành trên các tập dữ liệu phức tạp hơn nhằm đánh giá độ ổn định cũng như tính chắc chắn của phương pháp.

V. TÀI LIỆU THAM KHẢO

[1] Z. Huang, Z. Zhou, T. He, and X. Wang, "ACAC: Associative Classification Based on All-Confidence," IEEE International Conference on Granular Computing, pp. 289–293, 2011

[2] https://www.kaggle.com/datasets/uciml/mushroom-classification

[3] https://www.kaggle.com/datasets/yasserh/spamemailsdataset

[4] H. F. Ong, C. Y. M. Neoh, V. K. Vijayaraj, Y. X. Low, "Information-Based Rule Ranking for Associative Classification", pp. , ISPACS 2022

[5] M. Abrar, A. Tze and S. Abbas, "Associative Classification using Automata with Structure based Merging", *International Journal of Advanced Computer Science and Applications*, vol. 10, 2019

[6] D. L. Olson and G. Lauhoff, "Market Basket Analysis" in Descriptive Data Mining, Singapore:Springer Singapore, pp. 31-44, 2019.

[7] K. D. Rajab, "New Associative Classification Method Based on Rule Pruning for Classification of Datasets", *IEEE Access*, vol. 7, pp. 157783-157795, 2019.

[8] H. F. Ong, N. Mustapha, H. Hamdan, R. Rosli and A. Mustapha, "Informative top-k class associative rule for cancer biomarker discovery on microarray data", *Expert Systems with Applications*, vol. 146, 2020.

[9] Majid Seyf, Yue Xu, Richi Nayak, "DAC: Discriminative Associative Classifcation", SN Computer Science (2023) 4:401

[10] E.R.Omiecinski, "Alternative interest measures for mining associations in databases", IEEE Transactions on Knowledge and Data Engineering, vol 15, pp. 57-69, 2003.

An efficient approach for Associative Classification

Nguyễn Quốc Huy , Trần Anh Tuấn, Đỗ Như Tài, Nguyễn Thị Ngọc Thanh

Abstract— *Associative Classification is an interesting approach in data mining to create more accurate and easily interpretable predictive systems. This approach is often built on both association rule mining and classification techniques, to find a set of rules called association rules for classification (CAR) of label attributes. ACAC algorithm belongs to the family of CPAR problems but the classification accuracy is still low on big data. This paper proposes three phases to find the optimal attribute set to significantly improve the performance of ACAC algorithm in terms of time as well as accuracy on large data. Experimental results show that choosing the optimal data set makes the ACAC algorithm more useful on big data, especially the cost of optimizing the data set in these three phases is completely reasonable in practice.*