Integrating Classification and Association Rule Mining

—— the Secret Behind CBA

Written by Bing Liu, etc.

CBA Advantages

- One algorithm performs 3 tasks
- It can find some valuable rules that existing classification systems cannot.
- It can handle both table form data and transaction form data
- It doesn't require the whole database to be fetched into the main memory.

Problem Statement

Classification (predetermined target)

+

Association (no fix targets)

⬇

CBA (Classification Based on Associations)

Input and Output

- Input
  - Table form dataset(transformed needed) or transaction form dataset.
- Output
  - A complete set of CARs.(class association rule) – done by CBA-RG(rule generator)
  - A classifier. – done by CBA-CB(classifier builder)

CBA-RG: Basic concepts (1)

- The key operation of CBA-RG is to find all ruleitems that have support above minsup.
- **ruleitem**: <condset, y>, representing the rule: condset → y
- **condsupCount**: # of cases in D that contain the condset.
- **rulesupCount**: # of cases in D that contain the condset and are labeled with class y.

CBA-RG: Basic concepts (2)

- **support**
  - (rulesupCount / |D|) * 100%.
- **confidence**:
  - (rulesupCount / condsupCount) * 100%
- Example:
  - Ruleitem: <{(A, e), (B, p)}, (C, y)>
  - condsupCount: 3
  - rulesupCount: 2
  - support: (2 / 10) * 100% = 20%
  - confidence: (2 / 3) * 100% = 66.7%

CBA-RG: Basic concepts (3)

- **k-ruleitem**: A ruleitem whose condset has k items.
- **frequent ruleitems**: Ruleitems that satisfy minsup. Denoted as **F**$_k$ in the algorithm.
- **candidate ruleitems**:
  - Possibly frequent ruleitems generated somehow from the frequent ruleitems found in the last pass. Denoted as **C**$_k$.
- A ruleitem is represented in the algorithm in the form:
  - <(condset, condsupCount), (y, rulesupCount)>

The CBA-RG algorithm

```
F_1 = {large 1-ruleitems};
CAR_1 = genRules(F_1);
prCAR_1 = pruneRules(CAR_1);
for (k = 2; F_{k-1} ≠ ∅; k++) do
    C_k = candidateGen(F_{k-1});
    for each data case d ∈ D do
        C_d = ruleSubset(C_k, d);
        for each candidate c ∈ C_d do
            c.condsupCount++;
            if d.class = c.class then c.rulesupCount++
        end
    end
    F_k = {c ∈ C_k | c.rulesupCount ≥ minsup};
    CAR_k = genRules(F_k);
    prCAR_k = pruneRules(CAR_k);
end
CARs = ∪_k CAR_k;
prCARs = ∪_k prCAR_k;
```

A case study

A	B	C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

Attributes: A, B

Class: C

minsup: 15%

minconf: 60%

1st pass	F1	<({(A, e)}, 4), ((C, y), 3)>, <({(A, g)}, 5), ((C, y), 2)>, <({(A, g)}, 5), ((C, n), 3)>, <({(B, p)}, 3), ((C, y), 2)>, <({(B, q)}, 5), ((C, y), 3)>, <({(B, q)}, 5), ((C, n), 2)>, <({(B, w)}, 2), ((C, n), 2)>
2nd pass	C2	<{(A, e), (B, p)}, (C, y)>, <{(A, e), (B, q)}, (C, y)>, <{(A, g), (B, p)}, (C, y)>, <{(A, g), (B, q)}, (C, y)>, <{(A, g), (B, q)}, (C, n)>, <{(A, g), (B, w)}, (C, n)>
	F2	<({(A, e) , (B, p)}, 3), ((C, y), 2)>, <({(A, g) , (B, q)}, 3), ((C, y), 2)>, <({(A, g) , (B, q)}, 3), ((C, n), 1)>, <({(A, g) , (B, w)}, 2), ((C, n), 2)>

CAR$_1$	(A, e)→(C,y), (A, g)→(C,n), (B, p)→(C,y), (B, q)→(C,y), (B, w)→(C,n)
CAR$_2$	{(A, e), (B, p)} → (C, y), {(A, g), (B, q)} → (C, y) {(A, g), (B, w)} → (C, n)
CARs	CAR$_1$ È CAR$_2$

genRules(Fk):

- **possible rule** (**PR**): For all the ruleitem that have the same condset, the ruleitem with the highest confidence is chosen as a **PR**.
- If there are more than one ruleitem with the same highest confidence, we randomly pick one.
- **accurate rule**: confidence >= minconf

pruneRules(CARk):

- Uses pessimistic error rate based pruning method in C4.5. (Quinlan, J.R. 1992. C4.5: program for machine learning. Morgan Kaufmann)

prCAR1	(A, e)→(C,y), (A, g)→(C,n), (B, p)→(C,y), (B, q)→(C,y), (B, w)→(C,n)
prCAR2	{(A, g), (B, q)} → (C, y)
prCARs	prCAR1 È prCAR2

Classifier Builder

A	B	C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

CARs after pruning:

(1) A = e → y sup=3/10 conf=3/4

(2) A = g → n sup=3/10 conf=3/5

(3) B = p → y sup=2/10 conf=2/3

(4) B = q → y sup=3/10 conf=3/5

(5) B = w → n sup=2/10 conf=2/2

(6) A = g, B = q → y sup=2/10 conf=2/3

CBA-classifier builder

- **Goal** : select a small set of rules from the complete CARs as the classifier

<r_1, r_2, …, r_n, default_class>

where $r_i \in R$, $r_a \succ r_b$ if $b > a$. default_class is the default class.

CBA-CB specification

- **$\succ$ (Precedence) definition**

Given two rules, r_i and r_j, $r_i \succ r_j$ (also called r_i precedes r_j or r_i has a higher precedence than r_j) if

1. the confidence of r_i is greater than that of r_j, or
2. their confidences are the same, but the support of r_i is greater than that of r_j, or
3. both the confidences and supports of r_i and r_j are the same, but r_i is generated earlier than r_j;

CBA-CB two algorithms

- Two algorithms
  - **M1** (the database can be fetched into and processed in main memory). Suitable for **small datasets**

  - **M2**(the database can be resident in hard disk.) suitable for **huge datasets**

CBA-CB satisfaction conditions

- **Two conditions**

  Condition 1. Each training case is covered by the rule with the highest precedence among the rules that can cover the case.

  Condition 2. Every rule in C correctly classifies at least one remaining training case when it is chosen.

A	B	C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

CARs after pruning:

(1) $A = e \rightarrow y$ sup=3/10 conf=3/4

(2) $A = g \rightarrow n$ sup=3/10 conf=3/5

(3) $B = p \rightarrow y$ sup=2/10 conf=2/3

(4) $B = q \rightarrow y$ sup=3/10 conf=3/5

(5) $B = w \rightarrow n$ sup=2/10 conf=2/2

(6) $A = g, B = q \rightarrow y$ sup=2/10 conf=2/3

rule	#covCases	#cCovered	#wCovered	defClass	#errors

```
R = sort(R);
for each rule r ∈ R in sequence do
    temp = ∅;
    for each case d ∈ D do
        if d satisfies the conditions of r then
            store d.id in temp and mark r if it correctly classifies d;
        if r is marked then
            insert r at the end of C;
            delete all the cases with the ids in temp from D;
            selecting a default class for the current C;
            compute the total number of errors of C;
    end
end
Find the first rule p in C with the lowest total number of errors and drop all the rules after p in C;
Add the default class associated with p to end of C, and return C (our classifier).
```

CBA-CB M2

- M2 (more efficient algorithm for large datasets)

  Key point: instead of making one pass over the remaining data for each rule (in M1), we find the best rule in R to cover each case.

A	B	C
e	p	y
e	p	y
e	q	y
g	q	y
g	q	y
g	q	n
g	w	n
g	w	n
e	p	n
f	q	n

CARs after pruning:

(1) A = e → y sup=3/10 conf=3/4

(2) A = g → n sup=3/10 conf=3/5

(3) B = p → y sup=2/10 conf=2/3

(4) B = q → y sup=3/10 conf=3/5

(5) B = w → n sup=2/10 conf=2/2

(6) A = g, B = q → y sup=2/10 conf=2/3

A	B	C	covRules	cRule	wRule	U	Q	A
e	p	y	1, 3	1	null	1	1	
e	p	y	1, 3	1	null	1	1	
e	q	y	1, 3	1	null	1	1	
g	q	y	2, 4, 6	6	2	1,6	1,6	
g	q	y	2, 4, 6	6	2	1,6	1,6	
g	q	n	2, 4, 6	2	6	1,6,2	1,6	(6,n,2,6)
g	w	n	2, 5	5	null	1,6,2,5	1,6,5	(6,n,2,6)
g	w	n	2, 5	5	null	1,6,2,5	1,6,5	(6,n,2,6)
e	p	n	1, 3	null	1	1,6,2,5	1,6,5	(6,n,2,6),(9,n,null,1)
f	q	n	4	null	4	1,6,2,5	1,6,5	(6,n,2,6),(9,n,null,1)(10,n,null,4)

Empirical Evaluation

- 26 datasets from UIC ML Repository
- The results show that CBA produces more accurate classifiers.
- On average, the error rate decreases from 16.7% for C4.5rules to 15.6%-15.8% for CBA.
- Without or with rule pruning the accuracy of the resultant classifier is almost the same. So, the prCARs are sufficient for building accurae classifiers.
- Experiments show that both CBA-RG and CBA-CB(M2) have linear scaleup.

Conclusion

- Proposing a framework to integrate classification and association rule mining.
- An algorithm that generate all class association rules (CARs) and to build an accurate classifier.
- Contributions:
  - A new way to construct accurate classifiers;
  - It makes association rule mining techniques applicable to classification tasks;
  - It helps to solve a number of questions existing in current classification systems.