INTEGRATING CLASSIFICATION AND ASSOCIATION RULE MINING

ACM Special Interest Group on Knowledge Discovery and Data Mining (KDD) 1998

Bing Liu, Wynne Hsu, Yiming Ma

2011.11.28
Joon-Young Paik

OUTLINE

- Introduction
- Generating the Complete Set of CARs
- Building a Classifier
- Empirical Evaluation
- Conclusion
- Appendix

1. Introduction cont'd

- Classification rule mining and association rule mining are two important data mining techniques
- Integrating classification and association rule mining (CBA-Classification Based on Associations)
  - To focus on a special subset of association rules whose right-hand-side are restricted to the classification class attribute
    - The subset of rules is referred as the *class association rules* (CARs)
- Contributions
  - New way to build accurate classifier
  - Association rule mining techniques is used for classification tasks
  - Helping to solve a number of important problems with the existing classification tasks

1. Introduction

- Three steps
  - Discretizing continuous attributes, if any
  - Generating all the class association rules (CARs)
    - Called CBA-RG
  - Building a classifier based on the generated CARs
    - Called CBA-CB
- Objectives
  - Generating the complete set of CARs that satisfy the user-specified minimum support (*minsup*) and minimum confidence (*minconf*) constraints
  - Building a classifier from the CARs

2. Generating the Complete Set of CARs (CBA-RG) cont'd

- Basic concepts
  - CBA-RG finds all *ruleitems* that have support above *minsup*

- *ruleitem* is of the form: <*condset, y*>,
  where $condset\ is\ a\ set\ of\ items, y\ \in Y\ is\ a\ class\ label$
- *condsupCount* is the number of cases in *D* that contain the *condset*
- *rulesupCount* is the number of cases in *D* that contain the *condset* and are labeled with class *y*
- $support\ is$ "$\frac{rulesupCount}{|D|} \times 100\%$"
- $confidence\ is$ "$\frac{rulesupCount}{condsupCount} \times 100\%$"
- *frequent ruleitems* satisfy *minsup*
- example

1. $< \{(A,1),(B,1)\},(class: 1) >.$
2. $< \{(A,1),(B,1)\},(class: 2) >$

Assume the support count of the *condset* is 3

The support count of the first *ruleitem* is 2, and the second *ruleitem* is 1

The confidence of ruleitems is 66.7%, while the confidence of ruleitems 2 is 33.7%.

$\Rightarrow (A,1),(B,1) \longrightarrow (class,1)[supt = 20\%, confd = 66.7\%]$ (|D| is 10)

2. Generating the Complete Set of CARs (CBA-RG) cont'd

- 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.rulesup
        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;
```

- *k-ruleitem*: *ruleitem* whose *condset* has *k* item
- F_k: *the set of frequent k-ruleitem*
- C_k: *The set of candidate k-ruleitem*

- Make F_1
- Generate CAR_1 by *genRules*
- Apply prune operation
  - If rule r's pessimistic error rate is higher than the threshold, the rule r is pruned

2. Generating the Complete Set of CARs (CBA-RG) cont'd

- 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;
```

- *k-ruleitem*: *ruleitem* whose *condset* has *k* item
- F_k: *the set of frequent k-ruleitem*
- C_k: *The set of candidate k-ruleitem*

- Make C_k $from$ F_{k-1}
- Scan the data base and update various support counts of the candidates in C_k
- Make new F_k
- Generate CAR_k using *genRules*
- Apply prune operation

2. Generating the Complete Set of CARs (CBA-RG) cont'd

- 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;
```

- *k-ruleitem*: *ruleitem* whose *condset* has *k* item
- F_k: *the set of frequent k-ruleitem*
- C_k: *The set of candidate k-ruleitem*

2. Generating the Complete Set of CARs (CBA-RG) cont'd

- Example

A (attribute)	B (attribute)	C (attribute class)
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

- *minsup*: 15%
- *minconf*: 15%

F_1	<({(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)>
CAR_1	$(A,e) \longrightarrow (C,y)$, $(A,g) \longrightarrow (C,n)$, $(B,p) \longrightarrow (C,y)$, $(B,q) \longrightarrow (C,y)$, $(B,w) \longrightarrow (C,n)$
$preCAR_1$	$(A,e) \longrightarrow (C,y)$, $(A,g) \longrightarrow (C,n)$, $(B,p) \longrightarrow (C,y)$, $(B,q) \longrightarrow (C,y)$, $(B,w) \longrightarrow (C,n)$

2. Generating the Complete Set of CARs (CBA-RG)

- Example

A (attribute)	B (attribute)	C (attribute class)
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

- *minsup*: 15%
- *minconf*: 15%

C_2	<{(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)>
F_2	<({(A,e),(B,p)},3),((C,y),2)> <({(A,g),(B,q)},3),((C,y),2)> <({(A,g),(B,q)},3),((C,y),1)> <({(A,g),(B,w)},2),((C,y),2)>
CAR_2	$\{(A,e),(B,p)\} \longrightarrow (C,y)$, $\{(A,g),(B,q)\} \longrightarrow (C,y)$ $\{(A,g),(B,w)\} \longrightarrow (C,y)$
$preCAR_2$	$\{(A,g),(B,w)\} \longrightarrow (C,y)$

$$CAR_S\colon (A,e) \longrightarrow (C,y),\ (A,g) \longrightarrow (C,n),\ (B,p) \longrightarrow (C,y),\ (B,q) \longrightarrow (C,y),\ (B,w) \longrightarrow (C,n), \{(A,e),(B,p)\} \longrightarrow (C,y), \{(A,g),(B,q)\} \longrightarrow (C,y), \{(A,g),(B,w)\} \longrightarrow (C,y)$$

$$preCAR_S\colon (A,e) \longrightarrow (C,y),\ (A,g) \longrightarrow (C,n),\ (B,p) \longrightarrow (C,y),\ (B,q) \longrightarrow (C,y),\ (B,w) \longrightarrow (C,n), \{(A,g),(B,w)\} \longrightarrow (C,y)$$

3. Building a Classifier (CBA-CB) cont'd

- Basic Concepts
  - CBA-CB selects a small set of rules from the complete CARs as the classifier
  - Classifier for CBA is represented as $< r_1, r_2, \ldots, r_n, default\_class >$
  - Two algorithms $where\ r_i \in R, r_a \succ r_b\ if\ b > a$
    - √ M1
      - Suitable for small datasets
    - M2
      - Suitable for huge datasets

$Given\ two\ rules\ r_i\ and\ r_j,$
$r_i \succ r_j$ (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. $both\ the\ confidences\ and\ supports\ of\ r_i\ and\ r_j\ are\ the\ same, but\ r_i\ is\ generated\ earlier\ than\ r_j$

3. Building a Classifier (CBA-CB) cont'd

- CBA-CB: M1
  - Three steps
    - Sort the set of generated rules *R* according to the relation "≻"
    - Select rules for the classifier from *R* following the sorted sequence
    - Discard rules that do not improve the accuracy of the classifier

```
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).
```

3. Building a Classifier (CBA-CB) cont'd

- CBA-CB: M1
  - algorithm

```
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).
```

- Chose the highest precedence rules for the classifier

3. Building a Classifier (CBA-CB) cont'd

- CBA-CB: M1
  - algorithm

```
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).
```

- For each rule r, go through D to find those cases covered by r
- Mark r if it correctly classifier a case d
- If r can correctly classify at least one case, it will be a potential rule
- Those cases it covers are then removed from D
- A default class is selected
- Compute the total number of errors of C

3. Building a Classifier (CBA-CB) cont'd

- CBA-CB: M1
  - algorithm

```
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).
```

- Discard rules that do not improve the accuracy of the classifier
- The undiscarded rules and the default class form classifier

3. Building a Classifier (CBA-CB)

- CBA-CB: M1
  - Example

A (attribute)	B (attribute)	C (attribute class)
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

No.	Rules	support	confidence
1	$<(A,e)>\longrightarrow y$	3/10	3/4
2	$<(A,g)>\longrightarrow n$	3/10	3/5
3	$<(B,p)>\longrightarrow y$	2/10	2/3
4	$<(B,q)>\longrightarrow y$	3/10	3/5
5	$<(B,w)>\longrightarrow n$	2/10	2/2
6	$<(A,g),(B,q)>\longrightarrow y$	2/10	2/3

$$\mathbf{5 \succ 1 \succ 3 \succ 6 \succ 2 \succ 4}$$

- *minsup*: 15%
- *minconf*: 15%

3. Building a Classifier (CBA-CB)

- CBA-CB: M1
  - Example

A (attribute)	B (attribute)	C (attribute class)
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

- *minsup*: 15%
- *minconf*: 15%

$$\mathbf{5} \succ \mathbf{1} \succ \mathbf{3} \succ \mathbf{6} \succ \mathbf{2} \succ \mathbf{4}$$

No.	Rules	support	confidence
5	$<(B,w)>\longrightarrow n$	2/10	2/2

rule	No. of cases covered by condset	No. of cases correctly classified by the rule	No. of cases wrongly classified by the rule	Default class	No. of error
1	2	2	0	y	{0+3}=3

$$"C = \{\ \}" \quad \Rightarrow \quad "C = \{5\}"$$

3. Building a Classifier (CBA-CB)

- CBA-CB: M1
  - Example

A (attribute)	B (attribute)	C (attribute class)
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

- *minsup*: 15%
- *minconf*: 15%

$$5 \succ 1 \succ 3 \succ 6 \succ 2 \succ 4$$

No.	Rules	support	confidence
5	$<(B,w)>\longrightarrow n$	2/10	2/2
1	$<(A,e)>\longrightarrow y$	3/10	3/4

rule	No. of cases covered by condset	No. of cases correctly classified by the rule	No. of cases wrongly classified by the rule	Default class	No. of error
5	2	2	0	y	{0+3}=3
1	4	3	1	y	0+{1+2}=3

$$"C = \{\,5\,\}" \Rightarrow "C = \{5, 1\}"$$

3. Building a Classifier (CBA-CB)

- CBA-CB: M1
  - Example

A (attribute)	B (attribute)	C (attribute class)
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

- *minsup*: 15%
- *minconf*: 15%

$$5 \succ 1 \succ 3 \succ 6 \succ 2 \succ 4$$

No.	Rules	support	confidence
5	$<(B,w)> \longrightarrow n$	2/10	2/2
1	$<(A,e)> \longrightarrow y$	3/10	3/4
3	$<(B,p)> \longrightarrow y$	2/10	2/3

rule	No. of cases covered by condset	No. of cases correctly classified by the rule	No. of cases wrongly classified by the rule	Default class	No. of error
5	2	2	0	y	{0+3}=3
1	4	3	1	y	0+{1+2}=3
3	0	0	0	none	0

$$"C = \{\ 5, 1\ \}" \Rightarrow "C = \{5, 1\}"$$

3. Building a Classifier (CBA-CB)

- CBA-CB: M1
  - Example

A (attribute)	B (attribute)	C (attribute class)
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

- *minsup*: 15%
- *minconf*: 15%

$$5 \succ 1 \succ 3 \succ 6 \succ 2 \succ 4$$

No.	Rules	support	confidence
5	$<(B,w)>\longrightarrow n$	2/10	2/2
1	$<(A,e)>\longrightarrow y$	3/10	3/4
3	$<(B,p)>\longrightarrow y$	2/10	2/3
6	$<(A,g),(B,q)>\longrightarrow y$	2/10	2/3

rule	No. of cases covered by condset	No. of cases correctly classified by the rule	No. of cases wrongly classified by the rule	Default class	No. of error
5	2	2	0	y	{0+3}=3
1	4	3	1	y	0+{1+2}=3
6	3	2	1	n	0+1+{1+0}=2

$$"C = \{5, 1\}" \Rightarrow "C = \{5, 1, 6\}"$$

3. Building a Classifier (CBA-CB)

- CBA-CB: M1
  - Example

A (attribute)	B (attribute)	C (attribute class)
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

- *minsup*: 15%
- *minconf*: 15%

$$5 \succ 1 \succ 3 \succ 6 \succ 2 \succ 4$$

No.	Rules	support	confidence
5	$<(B,w)>\longrightarrow n$	2/10	2/2
1	$<(A,e)>\longrightarrow y$	3/10	3/4
3	$<(B,p)>\longrightarrow y$	2/10	2/3
6	$<(A,g),(B,q)>\longrightarrow y$	2/10	2/3
2	$<(A,g)>\longrightarrow n$	3/10	3/5

rule	No. of cases covered by condset	No. of cases correctly classified by the rule	No. of cases wrongly classified by the rule	Default class	No. of error
5	2	2	0	y	{0+3}=3
1	4	3	1	y	0+{1+2}=3
6	3	2	1	n	0+1+{1+0}=2
2	0	0	0	none	0

$$"C = \{\, 5,1,6 \,\}" \Rightarrow "C = \{5,1,6\}"$$

3. Building a Classifier (CBA-CB)

- CBA-CB: M1
  - Example

A (attribute)	B (attribute)	C (attribute class)
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

- *minsup*: 15%
- *minconf*: 15%

$5 \succ 1 \succ 3 \succ 6 \succ 2 \succ 4$

No.	Rules	support	confidence
5	$<(B,w)>\longrightarrow n$	2/10	2/2
1	$<(A,e)>\longrightarrow y$	3/10	3/4
3	$<(B,p)>\longrightarrow y$	2/10	2/3
6	$<(A,g),(B,q)>\longrightarrow y$	2/10	2/3
2	$<(A,g)>\longrightarrow n$	3/10	3/5
4	$<(B,q)>\longrightarrow y$	3/10	3/5

rule	No. of cases covered by condset	No. of cases correctly classified by the rule	No. of cases wrongly classified by the rule	Default class	No. of error
5	2	2	0	y	{0+3}=3
1	4	3	1	y	0+{1+2}=3
6	3	2	1	n	0+1+{1+0}=2
4	1	0	1	none	0+1+1+{1+0}=3

$"C=\{5,1,6\}" \Rightarrow "C=\{5,1,6\}"$

4. Empirical Evaluation cont'd

- Comparisons between CBA and C4.5
- Experimental environments
  - 26 datasets were used from UCI ML Repository
  - *minconf* is set to 50 %
  - *minsup* is set to 1 %
  - The total number of candidate rules in memory is set to a limit of 80,000
  - Discretization of continuous attributes is done using the Entropy method
  - All the error rates on each dataset are obtained from 10-fold cross-validations

4. Empirical Evaluation cont'd

Datasets	c4.5rules w/o discr.	c4.5rules discr.	CBA (CARs + infreq) w/o pru.	CBA (CARs + infreq) pru.	CBA (CARs) w/o pru	CBA (CARs) pru.	No. of CARs w/o pru.	No. of CARs pru.	Run time (sec) (CBA-RG) w/o pru	Run time (sec) (CBA-RG) pru.	Run time (sec) (CBA-CB) M1	Run time (sec) (CBA-CB) M2	No. of Rules in C
anneal*	5.2	6.5	1.9	1.9	3.2	3.6	65081	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
australian*	15.3	13.5	13.5	13.4	13.2	13.4	46564	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
auto*	19.9	29.2	21.0	23.1	24.0	27.2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
breast-w	5.0	3.9	3.9	3.9	4.2	4.2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
cleve*	21.8	18.2	18.1	19.1	16.7	16.7	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
crx*	15.1	15.9	14.3	14.3	14.1	14.1	42877	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
diabetes	25.8	27.6	24.8	25.5	24.7	25.3	3315	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
german*	27.7	29.5	27.2	26.5	25.2	26.5	69277	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
glass	31.3	27.5	27.4	27.4	27.4	27.4	4234	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
heart	19.2	18.9	19.6	19.6	18.5	18.5	52309	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hepatitis*	19.4	22.6	15.1	15.1	15.1	15.1	63134	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
horse*	17.4	16.3	18.2	17.9	18.7	18.7	62745	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hypo*	0.8	1.2	1.6	1.6	1.9	1.7	37631	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
ionosphere*	10.0	8.0	7.9	7.9	8.2	8.2	55701	10055	3.75	4.00	0.56	0.41	45
iris	4.7	5.3	7.1	7.1	7.1	7.1	72	23	0.00	0.00	0.00	0.00	5
labor	20.7	21.0	17.0	17.0	17.0	17.0	5565	313	0.17	0.20	0.00	0.00	12
led7	26.5	26.5	27.8	27.8	27.8	27.8	464	336	0.40	0.45	0.11	0.10	71
lymph*	26.5	21.0	20.3	18.9	20.3	19.6	40401	2965	2.70	2.70	0.07	0.05	36
pima	24.5	27.5	26.9	27.0	27.4	27.6	2977	125	0.23	0.25	0.04	0.02	45
sick*	1.5	2.1	2.8	2.8	2.7	2.7	71828	627	32.60	33.40	0.62	0.40	46
sonar*	29.8	27.8	24.3	21.7	24.3	21.7	57061	1693	5.34	5.22	0.30	0.12	37
tic-tac-toe	0.6	0.6	0.0	0.0	0.0	0.0	7063	1378	0.62	0.71	0.12	0.08	8
vehicle*	27.4	33.6	31.3	31.2	31.5	31.3	23446	5704	6.33	6.33	1.40	0.40	125
waveform*	21.9	24.6	20.2	20.2	20.4	20.6	9699	3396	13.65	13.55	2.72	1.12	386
wine	7.3	7.9	8.4	8.4	8.4	8.4	38070	1494	2.34	2.65	0.11	0.04	10
zoo*	7.8	7.8	5.4	5.4	5.4	5.4	52198	2049	2.73	2.70	0.61	0.32	7
Average	16.7	17.1	15.6	15.6	15.7	15.8	35140	2377	6.35	6.44	0.39	0.18	69

- CBA produces more accurate classifiers than C4.5 on average
- CBA is superior to C4.5 on 16 of the 26 datasets
- Without or with rule pruning, the accuracy of the resultant classifier is almost the same

4. Empirical Evaluation

Datasets	c4.5rules w/o discr.	c4.5rules discr.	CBA (CARs + infreq) w/o pru.	CBA (CARs + infreq) pru.	CBA (CARs) w/o pru	CBA (CARs) pru.	No. of CARs w/o pru.	No. of CARs pru.	Run time (sec) (CBA-RG) w/o pru	Run time (sec) (CBA-RG) pru.	Run time (sec) (CBA-CB) M1	Run time (sec) (CBA-CB) M2	No. of Rules in C
anneal*	5.2	6.5	1.9	1.9	3.2	3.6	65081	611	14.33	14.36	0.08	0.06	34
australian*	15.3	13.5	13.5	13.4	13.2	13.4	46564	4064	5.00	5.05	0.20	0.22	148
auto*	19.9	29.2	21.0	23.1	24.0	27.2	50236	3969	3.30	3.55	0.12	0.06	54
breast-w	5.0	3.9	3.9	3.9	4.2	4.2	2831	399	0.30	0.33	0.02	0.03	49
cleve*	21.8	18.2	18.1	19.1	16.7	16.7	48854	1634	4.00	4.30	0.04	0.06	78
crx*	15.1	15.9	14.3	14.3	14.1	14.1	42877	4717	4.90	5.06	0.43	0.30	142
diabetes	25.8	27.6	24.8	25.5	24.7	25.3	3315	162	0.25	0.28	0.03	0.01	57
german*	27.7	29.5	27.2	26.5	25.2	26.5	69277	4561	5.60	6.00	1.04	0.28	172
glass	31.3	27.5	27.4	27.4	27.4	27.4	4234	291	0.20	0.22	0.02	0.00	27
heart	19.2	18.9	19.6	19.6	18.5	18.5	52309	624	4.70	4.60	0.04	0.03	52
hepatitis*	19.4	22.6	15.1	15.1	15.1	15.1	63134	2275	2.80	2.79	0.09	0.05	23
horse*	17.4	16.3	18.2	17.9	18.7	18.7	62745	7846	3.2	3.33	0.35	0.19	97
hypo*	0.8	1.2	1.6	1.6	1.9	1.7	37631	493	45.60	45.30	1.02	0.40	35
ionosphere*	10.0	8.0	7.9	7.9	8.2	8.2	55701	10055	3.75	4.00	0.56	0.41	45
iris	4.7	5.3	7.1	7.1	7.1	7.1	72	23	0.00	0.00	0.00	0.00	5
labor	20.7	21.0	17.0	17.0	17.0	17.0	5565	313	0.17	0.20	0.00	0.00	12
led7	26.5	26.5	27.8	27.8	27.8	27.8	464	336	0.40	0.45	0.11	0.10	71
lymph*	26.5	21.0	20.3	18.9	20.3	19.6	40401	2965	2.70	2.70	0.07	0.05	36
pima	24.5	27.5	26.9	27.0	27.4	27.6	2977	125	0.23	0.25	0.04	0.02	45
sick*	1.5	2.1	2.8	2.8	2.7	2.7	71828	627	32.60	33.40	0.62	0.40	46
sonar*	29.8	27.8	24.3	21.7	24.3	21.7	57061	1693	5.34	5.22	0.30	0.12	37
tic-tac-toe	0.6	0.6	0.0	0.0	0.0	0.0	7063	1378	0.62	0.71	0.12	0.08	8
vehicle*	27.4	33.6	31.3	31.2	31.5	31.3	23446	5704	6.33	6.33	1.40	0.40	125
waveform*	21.9	24.6	20.2	20.2	20.4	20.6	9699	3396	13.65	13.55	2.72	1.12	386
wine	7.3	7.9	8.4	8.4	8.4	8.4	38070	1494	2.34	2.65	0.11	0.04	10
zoo*	7.8	7.8	5.4	5.4	5.4	5.4	52198	2049	2.73	2.70	0.61	0.32	7
Average	16.7	17.1	15.6	15.6	15.7	15.8	35140	2377	6.35	6.44	0.39	0.18	69

5. Conclusion

- Propose a framework to integrate classification and association rule mining
- Framework has two steps
  - To generate all class association rules (CARs)
  - To build an accurate classifier
- Future work
  - To focus on building more accurate classifiers by using more sophisticated techniques and to mine CARs without pre-discretization

Appendix cont’d

- CBA-CB: M2 (cont’d)
  - Instead of making one pass over the remaining data for each rule (M1),
    - To find the best rule in *R* to cover each case
  - There are three steps (cont’d)
    - Stage 1

```
Q = ∅; U = ∅; A = ∅;
for each case d ∈ D do
    cRule = maxCoverRule(C_c, d);
    wRule = maxCoverRule(C_w, d);
    U = U ∪ {cRule};
    cRule.classCasesCovered[d.class]++;
    if cRule ≻ wRule then
        Q = Q ∪ {cRule};
        mark cRule;
    else  A = A ∪ <d.id, d.class, cRule, wRule>
end
```

Appendix cont’d

- CBA-CB: M2 (cont’d)
  - There are three steps (cont’d)
    - Stage 2

```
for each entry <dID, y, cRule, wRule> ∈ A do
    if wRule is marked then
        cRule.classCasesCovered[y]--;
        wRule.classCasesCovered[y]++;
    else  wSet = allCoverRules(U, dID.case, cRule);
          for each rule w ∈ wSet do
              w.replace = w.replace ∪ {<cRule, dID, y>};
              w.classCasesCovered[y]++;
          end
          Q = Q ∪ wSet
    end
end
```

Appendix cont'd

- CBA-CB: M2 (cont'd)
  - There are three steps
    - Stage 3

```
classDistr = compClassDistri(D);
ruleErrors = 0;
Q = sort(Q);
for each rule r in Q in sequence do
   if r.classCasesCovered[r.class] ≠ 0 then
         for each entry <rul, dID, y> in r.replace do
              if the dID case has been covered by a
                  previous r then
                 r.classCasesCovered[y]--;
              else  rul.classCasesCovered[y]--;
         ruleErrors = ruleErrors + errorsOfRule(r);
         classDistr = update(r, classDistr);
         defaultClass = selectDefault(classDistr);
         defaultErrors = defErr(defaultClass, classDistr);
         totalErrors = ruleErrors + defaultErrors;
         Insert <r, default-class, totalErrors> at end of C
   end
end
Find the first rule p in C with the lowest totalErrors,
    and then discard all the rules after p from C;
Add the default class associated with p to end of C;
Return C without totalErrors and default-class;
```

Appendix

- CBA-CB: M2 (cont'd)
  - Example

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)