



# Classification II

---

COMP 790-90 Seminar

BCB 713 Module

Spring 2009

---

*The UNIVERSITY of NORTH CAROLINA at CHAPEL HILL*



## Association-Based Classification

---

- ▶ Several methods for association-based classification
  - ▶ **ARCS: Quantitative association mining and clustering of association rules (Lent et al'97)**
    - ▶ It beats C4.5 in (mainly) scalability and also accuracy
  - ▶ **Associative classification: (Liu et al'98)**
    - ▶ It mines high support and high confidence rules in the form of “cond\_set => y”, where y is a class label
  - ▶ **CAEP (Classification by aggregating emerging patterns) (Dong et al'99)**
    - ▶ Emerging patterns (EPs): the itemsets whose support increases significantly from one class to another
    - ▶ Mine Eps based on minimum support and growth rate



## Classification based on Association

---

- ▶ Classification rule mining versus Association rule mining
  - ▶ Aim
    - ▶ A small set of rules as classifier
    - ▶ All rules according to minsup and minconf
  - ▶ Syntax
    - ▶  $X \rightarrow y$
    - ▶  $X \rightarrow Y$



## Why & How to Integrate

---

- ▶ Both classification rule mining and association rule mining are indispensable to practical applications.
- ▶ The integration is done by focusing on a special subset of association rules whose right-hand-side are restricted to the classification class attribute.
  - ▶ CARs: class association rules



## CBA: Three Steps

---

- ▶ Discretize continuous attributes, if any
- ▶ Generate all class association rules (CARs)
- ▶ Build a classifier based on the generated CARs.



## Our Objectives

---

- ▶ To generate the complete set of CARs that satisfy the user-specified minimum support (minsup) and minimum confidence (minconf) constraints.
- ▶ To build a classifier from the CARs.



## Three Contributions

---

- ▶ It proposes a new way to build accurate classifiers.
- ▶ It makes association rule mining techniques applicable to classification tasks.
- ▶ It helps to solve a number of important problems with the existing classification systems, including:
  - ▶ *understandability* problem
  - ▶ *discovery of interesting or useful* rules
  - ▶ Disk v.s. Memory



## Schedule

---

- ▶ Introduction
- ▶ **CBA-RG: rule generator**
- ▶ CBA-CB: classifier builder
  - ▶ M1
  - ▶ M2
- ▶ Evaluation



# Rule Generator: Basic Concepts

- ▶ Ruleitem

<condset, y> : condset is a set of items, y is a class label

Each ruleitem represents a rule: condset->y

- ▶ condsupCount

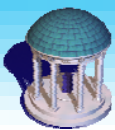
- ▶ The number of cases in D that contain condset

- ▶ rulesupCount

- ▶ The number of cases in D that contain the condset and are labeled with class y

- ▶ Support =  $(\text{rulesupCount}/|D|) * 100\%$

- ▶ Confidence =  $(\text{rulesupCount}/\text{condsupCount}) * 100\%$



# RG: Basic Concepts (Cont.)

- ▶ Frequent ruleitems

- ▶ A ruleitem is frequent if its support is above *minsup*

- ▶ Accurate rule

- ▶ A rule is accurate if its confidence is above *minconf*

- ▶ Possible rule

- ▶ For all ruleitems that have the same condset, the ruleitem with the highest confidence is the possible rule of this set of ruleitems.

- ▶ The set of class association rules (CARs) consists of all the **possible** rules (PRs) that are both **frequent** and **accurate**.



## RG: An Example

- ▶ A ruleitem:  $\langle \{(A,1),(B,1)\},(class,1) \rangle$ 
  - ▶ assume that
    - ▶ the support count of the condset (*condsupCount*) is 3,
    - ▶ the support of this ruleitem (*rulesupCount*) is 2, and
    - ▶  $|D|=10$
  - ▶ then  $(A,1),(B,1) \rightarrow (class,1)$ 
    - ▶  $supt=20\%$   $(rulesupCount/|D|)*100\%$
    - ▶  $confd=66.7\%$   $(rulesupCount/condsupCount)*100\%$



## RG: The Algorithm

```
1  $F_1 = \{\text{large 1-ruleitems}\};$ 
2  $CAR_1 = \text{genRules}(F_1);$ 
3  $prCAR_1 = \text{pruneRules}(CAR_1);$  //count the item and class occurrences to
                                //determine the frequent 1-ruleitems and prune it
4 for ( $k = 2; F_{k-1} \neq \emptyset; k++$ ) do
5      $C_k = \text{candidateGen}(F_{k-1});$  //generate the candidate ruleitems  $C_k$ 
                                //using the frequent ruleitems  $F_{k-1}$ 
6     for each data case  $d \in D$  do //scan the database
7          $C_d = \text{ruleSubset}(C_k, d);$  //find all the ruleitems in  $C_k$  whose condsets
                                //are supported by  $d$ 
8         for each candidate  $c \in C_d$  do
9              $c.condsupCount++;$ 
10            if  $d.class = c.class$  then
11                 $c.rulesupCount++;$  //update various support counts of the candidates in  $C_k$ 
12            end
13        end
```



## RG: The Algorithm(cont.)

```
13   $F_k = \{c \in C_k \mid c.\text{rulesupCount} \geq \text{minsup}\};$   
      //select those new frequent ruleitems to form  $F_k$   
14   $CAR_k = \text{genRules}(F_k);$  //select the ruleitems both accurate and frequent  
15   $prCAR_k = \text{pruneRules}(CAR_k);$   
16  end  
17   $CARs = \cup_k CAR_k;$   
18   $prCARs = \cup_k prCAR_k;$ 
```



## Schedule

- ▶ Introduction
- ▶ CBA-RG: rule generator
- ▶ **CBA-CB: class builder**
  - ▶ **M1** (algorithm for building a classifier using CARs or prCARs)
  - ▶ M2
- ▶ Evaluation



## Class Builder M1: Basic Concepts

- ▶ Given two rules  $r_i$  and  $r_j$ , define:  $r_i \succ r_j$  if
  - ▶ The confidence of  $r_i$  is greater than that of  $r_j$ , or
  - ▶ Their confidences are the same, but the support of  $r_i$  is greater than that of  $r_j$ , or
  - ▶ Both the confidences and supports are the same, but  $r_i$  is generated earlier than  $r_j$ .
- ▶ Our classifier is of the following format:
  - ▶  $\langle r_1, r_2, \dots, r_n, \text{default\_class} \rangle$ ,
    - ▶ where  $r_i \in R, r_a \succ r_b$  if  $b > a$



## M1: Three Steps

The basic idea is to choose a set of high precedence rules in  $R$  to cover  $D$ .

- ▶ Sort the set of generated rules  $R$
- ▶ Select rules for the classifier from  $R$  following the sorted sequence and put in  $C$ .
  - ▶ Each selected rule has to correctly classify at least one additional case.
  - ▶ Also select default class and compute errors.
- ▶ Discard those rules in  $C$  that don't improve the accuracy of the classifier.
  - ▶ Locate the rule with the lowest error rate and discard the rest rules in the sequence.



# Example

A	B	C	D	E	Class
0	0	1	1	0	Y
0	0	0	1	1	N
0	1	1	1	0	Y
1	1	1	1	0	Y
0	1	0	0	1	N

RuleItemsets	Support
BY	40%
CY	60%
DY	60%
EN	40%
BCY	40%
BDY	40%
CDY	60%
BCDY	40%

Min\_support = 40%    Min\_conf = 50%



# Example

Rules	Confidence	Support
B→Y	66.7%	40%
C→Y	100%	60%
D→Y	75%	60%
E→N	100%	40%
BC→Y	100%	40%
BD→Y	100%	40%
CD→Y	100%	60%
BCD→Y	100%	40%



# Example

Rules	Confidence	Support
$C \rightarrow Y$	100%	60%
$CD \rightarrow Y$	100%	60%
$E \rightarrow N$	100%	40%
$BC \rightarrow Y$	100%	40%
$BD \rightarrow Y$	100%	40%
$BCD \rightarrow Y$	100%	40%
$D \rightarrow Y$	75%	60%
$B \rightarrow Y$	66.7%	40%



# Example

A	B	C	D	E	Class	Rules	Confidence	Support
0	0	1	1	0	Y	$C \rightarrow Y$	100%	60%
0	0	0	1	1	N	$CD \rightarrow Y$	100%	60%
0	1	1	1	0	Y	$E \rightarrow N$	100%	40%
1	1	1	1	0	Y	$BC \rightarrow Y$	100%	40%
0	1	0	0	1	N	$BD \rightarrow Y$	100%	40%
						$BCD \rightarrow Y$	100%	40%
						$D \rightarrow Y$	75%	60%
						$B \rightarrow Y$	66.7%	40%



# Example

A	B	C	D	E	Class
0	0	1	1	0	Y
0	0	0	1	1	N
0	1	1	1	0	Y
1	1	1	1	0	Y
0	1	0	0	1	N

Rules	Confidence	Support
$C \rightarrow Y$	100%	60%
$CD \rightarrow Y$	100%	60%
$E \rightarrow N$	100%	40%
$BC \rightarrow Y$	100%	40%
$BD \rightarrow Y$	100%	40%
$BCD \rightarrow Y$	100%	40%
$D \rightarrow Y$	75%	60%
$B \rightarrow Y$	66.7%	40%

Default classification accuracy 60%



# Example

A	B	C	D	E	Class
0	0	1	1	0	Y
0	0	0	1	1	N
0	1	1	1	0	Y
1	1	1	1	0	Y
0	1	0	0	1	N

Rules	Confidence	Support
$C \rightarrow Y$	100%	60%
$CD \rightarrow Y$	100%	60%
$E \rightarrow N$	100%	40%
$BC \rightarrow Y$	100%	40%
$BD \rightarrow Y$	100%	40%
$BCD \rightarrow Y$	100%	40%
$D \rightarrow Y$	75%	60%
$B \rightarrow Y$	66.7%	40%

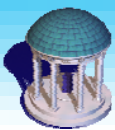




# Example

A	B	C	D	E	Class
0	0	1	1	0	Y
0	0	0	1	1	N
0	1	1	1	0	Y
1	1	1	1	0	Y
0	1	0	0	1	N

Rules	Confidence	Support
$C \rightarrow Y$	100%	60% ✓
$CD \rightarrow Y$	100%	60% ✗
$E \rightarrow N$	100%	40%
$BC \rightarrow Y$	100%	40%
$BD \rightarrow Y$	100%	40%
$BCD \rightarrow Y$	100%	40%
$D \rightarrow Y$	75%	60%
$B \rightarrow Y$	66.7%	40%



# Example

A	B	C	D	E	Class
0	0	1	1	0	Y
0	0	0	1	1	N
0	1	1	1	0	Y
1	1	1	1	0	Y
0	1	0	0	1	N

Rules	Confidence	Support
$C \rightarrow Y$	100%	60% ✓
$CD \rightarrow Y$	100%	60% ✗
$E \rightarrow N$	100%	40% ✓
$BC \rightarrow Y$	100%	40%
$BD \rightarrow Y$	100%	40%
$BCD \rightarrow Y$	100%	40%
$D \rightarrow Y$	75%	60%
$B \rightarrow Y$	66.7%	40%



# Example

A	B	C	D	E	Class	Rules	Confidence	Support	
0	0	1	1	0	Y	$C \rightarrow Y$	100%	60%	✓
0	0	0	1	1	N	$CD \rightarrow Y$	100%	60%	✗
0	1	1	1	0	Y	$E \rightarrow N$	100%	40%	✓
1	1	1	1	0	Y	$BC \rightarrow Y$	100%	40%	✗
0	1	0	0	1	N	$BD \rightarrow Y$	100%	40%	✗
						$BCD \rightarrow Y$	100%	40%	✗
						$D \rightarrow Y$	75%	60%	✗
						$B \rightarrow Y$	66.7%	40%	✗



# M1: Algorithm

- ▶ 1  $R = \text{sort}(R)$ ; //Step1:sort R according to the relation “>”
- ▶ 2 **for** each rule  $r \in R$  in sequence **do**
- ▶ 3    $temp = \emptyset$ ;
- ▶ 4   **for** each case  $d \in D$  **do** //go through D to find those cases covered by each rule r
- ▶ 5     **if**  $d$  satisfies the conditions of  $r$  **then**
- ▶ 6       store  $d.id$  in  $temp$  and mark  $r$  if it correctly classifies  $d$ ;
- ▶ 7     **if**  $r$  is marked **then**
- ▶ 8       insert  $r$  at the end of  $C$ ; //r will be a potential rule because it can correctly classify at least one case  $d$
- ▶ 9     delete all the cases with the ids in  $temp$  from  $D$ ;
- ▶ 10    selecting a default class for the current  $C$ ; //the majority class in the remaining data
- ▶ 11    compute the total number of errors of  $C$ ;
- ▶ 12    **end**
- ▶ 13 **end** // Step 2
- ▶ 14 Find the first rule  $p$  in  $C$  with the lowest total number of errors and drop all the rules after  $p$  in  $C$ ;
- ▶ 15 Add the default class associated with  $p$  to end of  $C$ , and return  $C$  (our classifier). //Step 3



## M1: Two conditions it satisfies

---

- ▶ Each training case is covered by the rule with the highest precedence among the rules that can cover the case.
- ▶ Every rule in  $C$  correctly classifies at least one remaining training case when it is chosen.



## M1: Conclusion

---

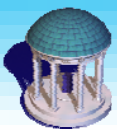
- ▶ The algorithm is simple, but inefficient especially when the database is not resident in the main memory. It needs too many passes over the database.
- ▶ The improved algorithm M2 takes slightly more than one pass.



# Schedule

---

- ▶ Introduction
- ▶ CBA-RG: rule generator
- ▶ **CBA-CB: class builder**
  - ▶ M1
  - ▶ M2
- ▶ Evaluation



# M2: Basic Concepts

---

- ▶ Key trick: instead of making one pass over the remaining data for each rule, we find the best rule in  $R$  to cover each case.
- ▶ cRule: highest precedence rule correctly classifying  $d$
- ▶ wRule: highest precedence rule wrongly classifying  $d$
- ▶ Three steps
  - ▶ Find all cRules needed (when  $cRule \succ wRule$ )
  - ▶ Find all wRules needed (when  $wRule \succ cRule$ )
  - ▶ Remove rules with high error



## M2: Stage 1

- ▶ 1  $Q = \emptyset; U = \emptyset; A = \emptyset;$
- ▶ 2 **for** each case  $d \in D$  **do**
- ▶ 3    $cRule = \text{maxCoverRule}(C_c, d);$
- ▶ 4    $wRule = \text{maxCoverRule}(C_w, d);$
- ▶ 5    $U = U \cup \{cRule\};$
- ▶ 6    $cRule.\text{classCasesCovered}[d.\text{class}]++;$
- ▶ 7   **if**  $cRule > wRule$  **then**
- ▶ 8      $Q = Q \cup \{cRule\};$
- ▶ 9     mark  $cRule;$
- ▶ 10  **else**  $A = A \cup \langle d.\text{id}, d.\text{class}, cRule, wRule \rangle$
- ▶ 11 **end**



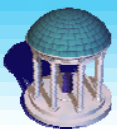
## Funs & Vars of Stage 1 (M2)

- ▶ **maxCoverRule** finds the highest precedence rule that covers the case  $d$ .
- ▶ **d.id** represent the identification number of  $d$
- ▶ **d.class** represent the class of  $d$
- ▶ **r.classCasesCovered[d.class]** record how many cases rule  $r$  covers in  $d.class$



## M2: Stage 2

- ▶ 1 **for** each entry  $\langle dID, y, cRule, wRule \rangle \in A$  **do**
- ▶ 2   **if**  $wRule$  is marked **then**
- ▶ 3      $cRule.classCasesCovered[y]--;$
- ▶ 4      $wRule.classCasesCovered[y]++;$
- ▶ 5   **else**  $wSet = allCoverRules(U, dID.case, cRule);$
- ▶ 6     **for** each rule  $w \in wSet$  **do**
- ▶ 7        $w.replace = w.replace \cup \{ \langle cRule, dID, y \rangle \};$
- ▶ 8        $w.classCasesCovered[y]++;$
- ▶ 9     **end**
- ▶ 10     $Q = Q \cup wSet$
- ▶ 11   **end**
- ▶ 12 **end**



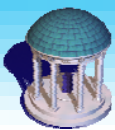
## Funs & Vars of Stage 2 (M2)

- ▶ **allCoverRules** find all the rules that wrongly classify the specified case and have higher precedences than that of its  $cRule$ .
- ▶ **r.replace** record the information that rule  $r$  can replace some  $cRule$  of a case



## M2: Stage 3

- ▶ 1 *classDistr* = compClassDistr(D); 2 *ruleErrors* = 0; 3 *Q* = sort(Q);
- ▶ 4 **for** each rule *r* in *Q* in sequence **do**
- ▶ 5   **if** *r.classCasesCovered*[*r.class*] ≠ 0 **then**
- ▶ 6     **for** each entry *<rul, dID, y>* in *r.replace* **do**
- ▶ 7       **if** the *dID* case has been covered by a previous *r* **then** *r.classCasesCovered*[*y*]--;
- ▶ 9       **else** *rul.classCasesCovered*[*y*]--;
- ▶ 10    *ruleErrors* = *ruleErrors* + *errorsOfRule*(*r*);
- ▶ 11    *classDistr* = *update*(*r*, *classDistr*);
- ▶ 12    *defaultClass* = *selectDefault*(*classDistr*);
- ▶ 13    *defaultErrors* = *defErr*(*defaultClass*, *classDistr*);
- ▶ 14    *totalErrors* = *ruleErrors* + *defaultErrors*;
- ▶ 15    Insert *<r, default-class, totalErrors>* at end of *C*
- ▶ 16 **end**
- ▶ 17 **end**
- ▶ 18 Find the first rule *p* in *C* with the lowest *totalErrors*, and then discard all the rules after *p* from *C*;
- ▶ 19 Add the default class associated with *p* to end of *C*;
- ▶ 20 Return *C* without *totalErrors* and *default-class*;



## Funs & Vars of Stage 3 (M2)

- ▶ **compClassDistr** counts the number of training cases in each class in the initial training data.
- ▶ **ruleErrors** records the number of errors made so far by the selected rules on the training data.
- ▶ **defaultClass** number of errors of the chosen default Class.
- ▶ **totalErrors** the total number of errors of selected rules in *C* and the default class.



## Schedule

---

- ▶ Introduction
- ▶ CBA-RG: rule generator
- ▶ CBA-CB: class builder
  - ▶ M1
  - ▶ M2
- ▶ **Evaluation**



## Empirical Evaluation

---

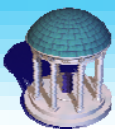
- ▶ Compare with C4.5
- ▶ Selection of minconf and minsup
- ▶ Limit candidates in memory
- ▶ Discretization (Entropy method 1993)
- ▶ DEC alpha 500, 192MB



# Evaluation

DataSets	C4.5 rules w/o discr.	C4.5 rules discr.	CBA (CARs + infreq)		CBA (CARs) w/o pru		No. of CARs		Run time (sec) (CBA-RG)		Run time (sec) (CBA-CB)		No. of Rules in C
			w/o pru.	Pru.	w/o pru	pru.	w/o pru.	pru.	w/o pru	pru.	M1	M2	
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	5.05	0.2	0.22	148
auto*	19.9	29.2	21	23.1	24	27.2	50236	3969	3.3	3.55	0.12	0.06	54
breast-w	5	3.9	3.9	3.9	4.2	4.2	2831	399	0.3	0.33	0.02	0.03	49
cleve*	21.8	18.2	18.1	19.1	16.7	16.7	48854	1634	4	4.3	0.04	0.06	78
crx*	15.1	15.9	14.3	14.3	14.1	14.1	42877	4717	4.9	5.06	0.43	0.3	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	5.6	6	1.04	0.28	172
glass	31.3	27.5	27.4	27.4	27.4	27.4	4234	291	0.2	0.22	0.02	0	27
heart	19.2	18.9	19.6	19.6	18.5	18.5	52309	624	4.7	4.6	0.04	0.03	52
hepatitis*	19.4	22.6	15.1	15.1	15.1	15.1	63134	2275	2.8	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	45.6	45.3	1.02	0.4	35
ionosphere*	10	8	7.9	7.9	8.2	8.2	55701	10055	3.75	4	0.56	0.41	45
iris	4.7	5.3	7.1	7.1	7.1	7.1	72	23	0	0	0	0	5
labor	20.7	21	17	17	17	17	5565	313	0.17	0.2	0	0	12
led7	26.5	26.5	27.8	27.8	27.8	27.8	464	336	0.4	0.45	0.11	0.1	71
lymph*	26.5	21	20.3	18.9	20.3	19.6	40401	2965	2.7	2.7	0.07	0.05	36
pima	24.5	27.5	26.9	27	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	32.6	33.4	0.62	0.4	46
sonar*	29.8	27.8	24.3	21.7	24.3	21.7	57061	1693	5.34	5.22	0.3	0.12	37
tic-tac-toe	0.6	0.6	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	1.4	0.4	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	2.7	0.61	0.32	7
<b>Average</b>	<b>16.7</b>	<b>17.1</b>	<b>15.6</b>	<b>15.6</b>	<b>15.7</b>	<b>15.8</b>	<b>35140</b>	<b>2377</b>	<b>6.35</b>	<b>6.44</b>	<b>0.39</b>	<b>0.18</b>	<b>69</b>

39



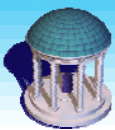
# Summary

- ▶ Classification is an **extensively studied** problem (mainly in statistics, machine learning & neural networks)
- ▶ Classification is probably one of the most **widely used** data mining techniques with a lot of extensions
- ▶ **Scalability** is still an important issue for database applications: thus combining classification **with database techniques** should be a promising topic
- ▶ Research directions: classification of **non-relational data**, e.g., text, spatial, multimedia, etc..



## References (1)

- ▶ C. Apte and S. Weiss. Data mining with decision trees and decision rules. *Future Generation Computer Systems*, 13, 1997.
- ▶ L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.
- ▶ C. J. C. Burges. [A Tutorial on Support Vector Machines for Pattern Recognition](#). *Data Mining and Knowledge Discovery*, 2(2): 121-168, 1998.
- ▶ P. K. Chan and S. J. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In Proc. 1st Int. Conf. Knowledge Discovery and Data Mining (KDD'95), pages 39-44, Montreal, Canada, August 1995.
- ▶ U. M. Fayyad. Branching on attribute values in decision tree generation. In Proc. 1994 AAAI Conf., pages 601-606, AAAI Press, 1994.
- ▶ J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest: A framework for fast decision tree construction of large datasets. In Proc. 1998 Int. Conf. Very Large Data Bases, pages 416-427, New York, NY, August 1998.
- ▶ J. Gehrke, V. Gant, R. Ramakrishnan, and W.-Y. Loh, [BOAT -- Optimistic Decision Tree Construction](#). In *SIGMOD'99*, Philadelphia, Pennsylvania, 1999



## References (2)

- ▶ M. Kamber, L. Winstone, W. Gong, S. Cheng, and J. Han. Generalization and decision tree induction: Efficient classification in data mining. In Proc. 1997 Int. Workshop Research Issues on Data Engineering (RIDE'97), Birmingham, England, April 1997.
- ▶ B. Liu, W. Hsu, and Y. Ma. [Integrating Classification and Association Rule Mining](#). *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98)* New York, NY, Aug. 1998.
- ▶ W. Li, J. Han, and J. Pei, [CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules](#), Proc. 2001 Int. Conf. on Data Mining (ICDM'01), San Jose, CA, Nov. 2001.
- ▶ J. Magidson. The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection. In R. P. Bagozzi, editor, *Advanced Methods of Marketing Research*, pages 118-159. Blackwell Business, Cambridge Massachusetts, 1994.
- ▶ M. Mehta, R. Agrawal, and J. Rissanen. SLIQ : A fast scalable classifier for data mining. (EDBT'96), Avignon, France, March 1996.



## References (3)

- ▶ T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- ▶ S. K. Murthy, Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey, *Data Mining and Knowledge Discovery* 2(4): 345-389, 1998
- ▶ J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81-106, 1986.
- ▶ J. R. Quinlan. Bagging, boosting, and c4.5. In Proc. 13th Natl. Conf. on Artificial Intelligence (AAAI'96), 725-730, Portland, OR, Aug. 1996.
- ▶ R. Rastogi and K. Shim. Public: A decision tree classifier that integrates building and pruning. In Proc. 1998 Int. Conf. Very Large Data Bases, 404-415, New York, NY, August 1998.
- ▶ J. Shafer, R. Agrawal, and M. Mehta. SPRINT : A scalable parallel classifier for data mining. In Proc. 1996 Int. Conf. Very Large Data Bases, 544-555, Bombay, India, Sept. 1996.
- ▶ S. M. Weiss and C. A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufman, 1991.
- ▶ S. M. Weiss and N. Indurkha. *Predictive Data Mining*. Morgan Kaufmann, 1997.



# Thank you !!!

