


Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification 
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy:
Ensemble Methods
- Summary

41

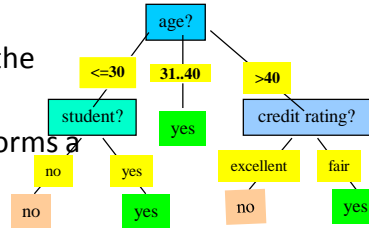
Using IF-THEN Rules for Classification

- Represent the knowledge in the form of **IF-THEN** rules
 - R: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes
 - Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: *coverage* and *accuracy*
 - n_{covers} = # of tuples covered by R
 - n_{correct} = # of tuples correctly classified by R
 - coverage(R) = $n_{\text{covers}} / |D|$ /* D: training data set */
 - accuracy(R) = $n_{\text{correct}} / n_{\text{covers}}$
- If more than one rule are triggered, need **conflict resolution**
 - Size ordering: assign the highest priority to the triggering rules that has the “toughest” requirement (i.e., with the *most attribute tests*)
 - Class-based ordering: decreasing order of *prevalence* or *misclassification cost per class*
 - Rule-based ordering (**decision list**): rules are organized into one long priority list, according to some measure of rule quality or by experts

42

Rule Extraction from a Decision Tree

- Rules are *easier to understand* than large trees
- One rule is created *for each path* from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive
- Example: Rule extraction from our *buys_computer* decision-tree



IF <i>age</i> = young AND <i>student</i> = no	THEN <i>buys_computer</i> = no
IF <i>age</i> = young AND <i>student</i> = yes	THEN <i>buys_computer</i> = yes
IF <i>age</i> = mid-age	THEN <i>buys_computer</i> = yes
IF <i>age</i> = old AND <i>credit_rating</i> = excellent	THEN <i>buys_computer</i> = no
IF <i>age</i> = old AND <i>credit_rating</i> = fair	THEN <i>buys_computer</i> = yes

43

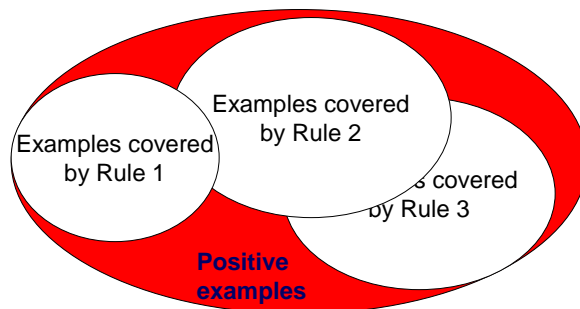
Rule Induction: Sequential Covering Method

- Sequential covering algorithm: Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Steps:
 - Rules are learned one at a time
 - Each time a rule is learned, the tuples covered by the rules are removed
 - Repeat the process on the remaining tuples until *termination condition*, e.g., when no more training examples or when the quality of a rule returned is below a user-specified threshold
- Comp. w. decision-tree induction: learning a set of rules *simultaneously*

44

Sequential Covering Algorithm

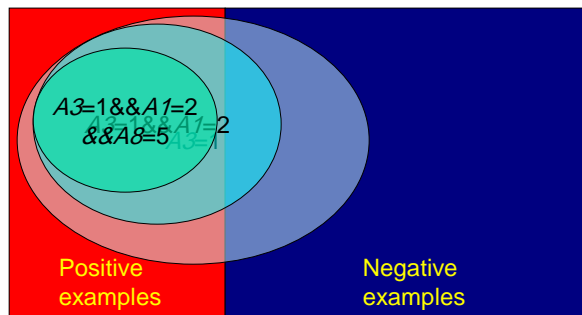
while (enough target tuples left)
 generate a rule
 remove positive target tuples satisfying this rule



45

Rule Generation

- To generate a rule
 while(true)
 find the best predicate p
 if $\text{foil-gain}(p) > \text{threshold}$ **then** add p to current rule
 else break



46

Example

I D	A	B	C	Y
1	a ₁	b ₁	c ₁	y ₁
2	a ₁	b ₂	c ₁	y ₁
3	a ₁	b ₂	c ₂	y ₂
4	a ₂	b ₁	c ₁	y ₂
5	a ₂	b ₂	c ₃	y ₂
6	a ₃	b ₁	c ₂	y ₂
7	a ₃	b ₁	c ₁	y ₁
8	a ₃	b ₂	c ₁	y ₂
9	a ₃	b ₂	c ₃	y ₂
10	a ₃	b ₁	c ₃	y ₁

If ? Then Y=y₁

A=a ₁	Accuracy(A=a ₁ , Y=y ₁)=n _{correct} /n _{cover} =2/3
A=a ₂	Accuracy(A=a ₂ , Y=y ₁)=n _{correct} /n _{cover} =0/2
A=a ₃	Accuracy(A=a ₃ , Y=y ₁)=n _{correct} /n _{cover} =2/5
B=b ₁	Accuracy(B=b ₁ , Y=y ₁)=n _{correct} /n _{cover} =3/5
B=b ₂	Accuracy(B=b ₂ , Y=y ₁)=n _{correct} /n _{cover} =1/5
C=c ₁	Accuracy(C=c ₁ , Y=y ₁)=n _{correct} /n _{cover} =3/5
C=c ₂	Accuracy(C=c ₂ , Y=y ₁)=n _{correct} /n _{cover} =0/2
C=c ₃	Accuracy(C=c ₃ , Y=y ₁)=n _{correct} /n _{cover} =1/3

If A=a₁ Then Y=y₁

47

Example

I D	A	B	C	Y
1	a ₁	b ₁	c ₁	y ₁
2	a ₁	b ₂	c ₁	y ₁
3	a ₁	b ₂	c ₂	y ₂
4	a ₂	b ₁	c ₁	y ₂
5	a ₂	b ₂	c ₃	y ₂
6	a ₃	b ₁	c ₂	y ₂
7	a ₃	b ₁	c ₁	y ₁
8	a ₃	b ₂	c ₁	y ₂
9	a ₃	b ₂	c ₃	y ₂
10	a ₃	b ₁	c ₃	y ₁

If A=a₁ Then Y=y₁

B=b ₁	Accuracy(A=a ₁ and B=b ₁ , Y=y ₁)=n _{correct} /n _{cover} =1/1
B=b ₂	Accuracy(A=a ₁ and B=b ₂ , Y=y ₁)=n _{correct} /n _{cover} =1/2
C=c ₁	Accuracy(A=a ₁ and C=c ₁ , Y=y ₁)=n _{correct} /n _{cover} =2/2
C=c ₂	Accuracy(A=a ₁ and C=c ₂ , Y=y ₁)=n _{correct} /n _{cover} =0/1
C=c ₃	Accuracy(A=a ₁ and C=c ₃ , Y=y ₁)=n _{correct} /n _{cover} =0/0

If (A=a₁ and C=c₁) Then Y=y₁

48

How to Learn-One-Rule?

- Start with the *most general rule* possible: condition = empty
- *Adding new attributes* by adopting a greedy depth-first strategy
 - Picks the one that most improves the rule quality
- Rule-Quality measures: consider both coverage and accuracy
 - Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition

$$FOIL_Gain = pos' \times (\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg})$$

- favors rules that have high accuracy and cover many positive tuples
- Rule pruning based on an independent set of test tuples

$$FOIL_Prune(R) = \frac{pos - neg}{pos + neg}$$

Pos/neg are # of positive/negative tuples covered by R.

If *FOIL_Prune* is higher for the pruned version of R, prune R

49