



Classification: Rule Induction

Data Mining and Text Mining (UIC 583 @ Politecnico di Milano)

- What is rule induction?
- What is a classification rule?
- Rule Learning
 - The OneRule Algorithm
 - Sequential Covering Algorithms
- Indirect Methods

What is Rule Induction?

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

- IF (humidity = high) and (outlook = sunny)
THEN play=no (3.0/0.0)
- IF (outlook = rainy) and (windy = TRUE)
THEN play=no (2.0/0.0)
- OTHERWISE play=yes (9.0/0.0)

- Confusion Matrix
 - yes no <-- classified as
 - 7 2 | yes
 - 3 2 | no

Let's Check the First Rule

6

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

IF (humidity = high) and (outlook = sunny) THEN play=no (3.0/0.0)

Then, The Second Rule

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

IF (outlook = rainy) and (windy = TRUE) THEN play=no (2.0/0.0)

Finally, The Third Rule

8

Outlook	Temp	Humidity	Windy	Play
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Overcast	Cool	Normal	True	Yes
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes

OTHERWISE play=yes (9.0/0.0)

- IF (outlook = sunny) THEN play IS no
ELSE IF (outlook = overcast) THEN play IS yes
ELSE IF (outlook = rainy) THEN play IS yes
(10/14 instances correct)
- Confusion Matrix

yes no <-- classified as

4 5 | yes

3 2 | no

What is a Classification Rule?

What Is A Classification Rules?

Why Rules?

- They are IF-THEN rules
 - The IF part states a condition over the data
 - The THEN part includes a class label
- What types of conditions?
 - Propositional, with attribute-value comparisons
 - First order Horn clauses, with variables
- Why rules? Because they are one of the most expressive and most human readable representation for hypotheses is sets of IF-THEN rules

- IF (humidity = high) and (outlook = sunny)
THEN play=no (3.0/0.0)
- n_{covers} = number of examples covered by the rule
- n_{correct} = number of examples correctly classified by the rule
- $\text{coverage}(R) = n_{\text{covers}} / \text{size of the training data set}$
- $\text{accuracy}(R) = n_{\text{correct}} / n_{\text{covers}}$

- If more than one rule is triggered, we need conflict resolution
- Size ordering: assign the highest priority to the triggering rules that has the “toughest” requirement (i.e., with the most attribute test)
- 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

Rule Learning

- Direct Methods
 - Directly learn the rules from the training data
- Indirect Methods
 - Learn decision tree, then convert to rules
 - Learn neural networks, then extract rules

One Rule

- OneRule (IR) learns a simple rule involving one attribute
 - Assumes nominal attributes
 - The rule that all the values of one particular attribute
- Basic version
 - One branch for each value
 - Each branch assigns most frequent class
 - Error rate: proportion of instances that don't belong to the majority class of their corresponding branch
 - Choose attribute with lowest error rate
 - “missing” is treated as a separate value

```
For each attribute,  
  For each value of the attribute,  
  make a rule as follows:  
    count how often each class appears  
    find the most frequent class  
    make the rule assign that class to  
    this attribute-value  
  Calculate the error rate of the rules  
  
Choose the rules with the smallest error rate
```

Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temp	Hot → No*	2/4	5/14
	Mild → Yes	2/6	
	Cool → Yes	1/4	
Humidity	High → No	3/7	4/14
	Normal → Yes	1/7	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

* indicates a tie

- Applies simple supervised discretization
- Sort instances according to attribute's values
- Place breakpoints where class changes (majority class)
- This procedure is however very sensitive to noise since one example with an incorrect class label may produce a separate interval. **This is likely to lead to overfitting.**
- In the case of the temperature,

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

- To limit overfitting, enforce minimum number of instances in majority class per interval.
- For instance, in the case of the temperature, if we set the minimum number of majority class instances to 3, we have

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

join the intervals to get at least 3 examples

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	⊘ No	⊘ Yes	Yes	Yes	No	No	Yes	⊘ Yes	Yes	No	⊘ Yes	Yes	⊘ No

join the intervals with the same majority class

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	⊘ No	No	Yes	Yes	Yes	No	Yes	Yes	No

OneRule Applied to the Numerical Version of the Weather Dataset

Attribute	Rules	Errors	Total errors
Outlook	Sunny \rightarrow No	2/5	4/14
	Overcast \rightarrow Yes	0/4	
	Rainy \rightarrow Yes	2/5	
Temperature	$\leq 77.5 \rightarrow$ Yes	3/10	5/14
	$> 77.5 \rightarrow$ No*	2/4	
Humidity	$\leq 82.5 \rightarrow$ Yes	1/7	3/14
	> 82.5 and $\leq 95.5 \rightarrow$ No	2/6	
	$> 95.5 \rightarrow$ Yes	0/1	
Windy	False \rightarrow Yes	2/8	5/14
	True \rightarrow No*	3/6	

Sequential Covering Algorithm

- Consider the set E of positive and negative examples
- Repeat
 - Learn one rule with high accuracy, any coverage
 - Remove positive examples covered by this rule
- Until all the examples are covered

```
procedure Covering (Examples, Classifier)
input: a set of positive and negative examples for class c

// rule set is initially empty
classifier = {}

while PositiveExamples(Examples) != {}
    // find the best rule possible
    rule = FindBestRule(Examples)
    // check if we need more rules
    if Stop(Examples, rule, Classifier) breakwhile
    // remove covered examples and update the model
    Examples = Examples \ Cover(Rule, Examples)
    Classifier = Classifier U {rule}
Enbwhile
// post-process the rules (sort them, simplify them, etc.)
Classifier = PostProcessing(Classifier)
output: Classifier
```

IF THEN Play=yes

IF Wind=yes
THEN Play=yes

$P=5/10 = 0.5$

IF Wind=No
THEN Play=yes

$P=6/8=0.75$

IF Humidity=Normal
THEN Play=yes

$P=6/7=0.86$

IF Humidity=High
THEN Play=yes

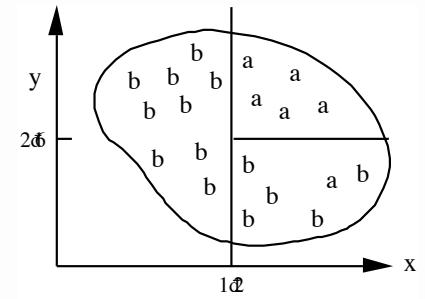
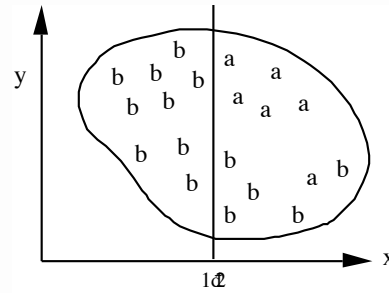
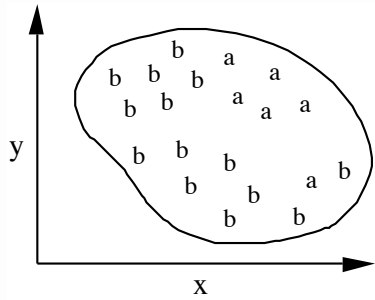
$P=3/7=0.43$

IF ...
THEN ...

IF Humidity=Normal
AND Wind=yes
THEN Play=yes

IF Humidity=Normal
AND Wind=No
THEN Play=yes

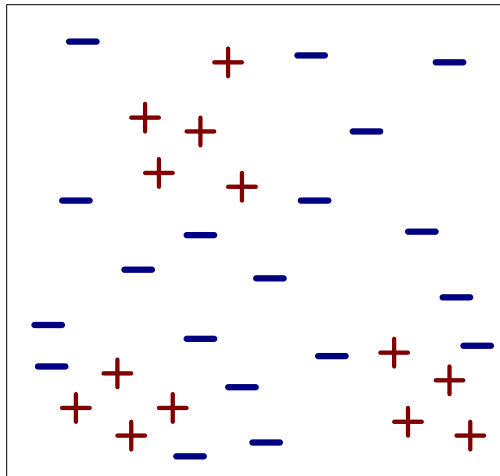
IF Humidity=Normal
AND Outlook=Rainy
THEN Play=yes



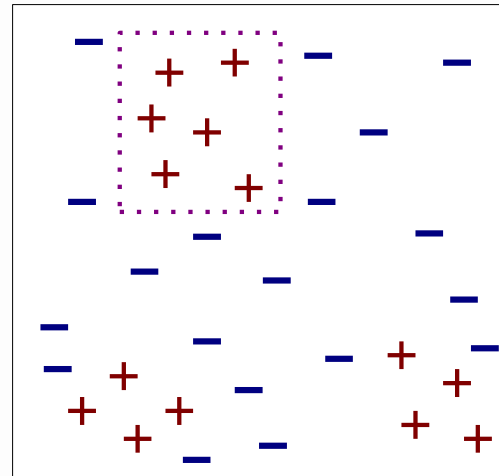
If true then class = a

If $x > 1.2$ then class = a

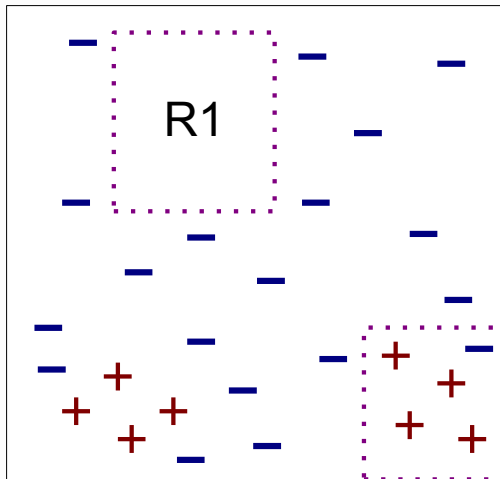
If $x > 1.2$ and $y > 2.6$
then class = a



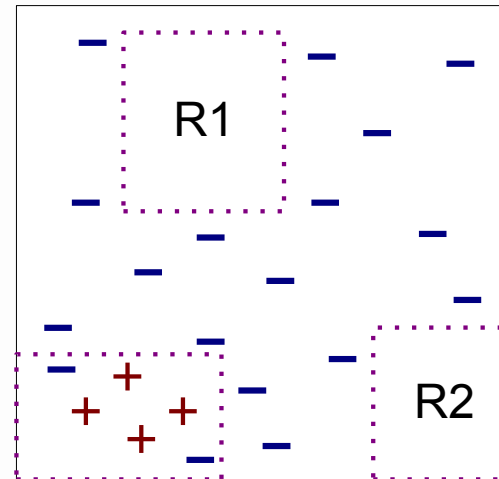
(i) Original Data



(ii) Step 1



(iii) Step 2



(iv) Step 3

```
LearnOneRule(Attributes, Examples, k)
  init BH to the most general hypothesis
  init CH to {BH}

  while CH not empty Do
    Generate Next More Specific CH in NCH
    // check all the NCH for an hypothesis that
    // improves the performance of BH
    Update BH
    Update CH with the k best NCH
  endwhile
  return a rule "IF BH THEN prediction"
```

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	No	Reduced	None
Young	Myope	No	Normal	Soft
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	No	Reduced	None
Young	Hypermetrope	No	Normal	Soft
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	No	Reduced	None
Pre-presbyopic	Myope	No	Normal	Soft
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	No	Reduced	None
Pre-presbyopic	Hypermetrope	No	Normal	Soft
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	No	Reduced	None
Presbyopic	Myope	No	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	No	Reduced	None
Presbyopic	Hypermetrope	No	Normal	Soft
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

- Rule we seek:

```
If ?  
    then recommendation = hard
```

- Possible tests:

Age = Young	2/8
Age = Pre-presbyopic	1/8
Age = Presbyopic	1/8
Spectacle prescription = Myope	3/12
Spectacle prescription = Hypermetrope	1/12
Astigmatism = no	0/12
Astigmatism = yes	4/12
Tear production rate = Reduced	0/12
Tear production rate = Normal	4/12

- Rule with best test added,

```
If astigmatism = yes  
then recommendation = hard
```

- Instances covered by modified rule,

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Reduced	None
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Reduced	None
Young	Hypermetrope	Yes	Normal	hard
Pre-presbyopic	Myope	Yes	Reduced	None
Pre-presbyopic	Myope	Yes	Normal	Hard
Pre-presbyopic	Hypermetrope	Yes	Reduced	None
Pre-presbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

- Current state,

```
If astigmatism = yes  
and ?  
then recommendation = hard
```

- Possible tests,

Age = Young	2/4
Age = Pre-presbyopic	1/4
Age = Presbyopic	1/4
Spectacle prescription = Myope	3/6
Spectacle prescription = Hypermetrope	1/6
Tear production rate = Reduced	0/6
Tear production rate = Normal	4/6

- Rule with best test added:

```
If astigmatism = yes
    and tear production rate = normal
then recommendation = Hard
```

- Instances covered by modified rule

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
Young	Myope	Yes	Normal	Hard
Young	Hypermetrope	Yes	Normal	Hard
Prepresbyopic	Myope	Yes	Normal	Hard
Prepresbyopic	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Normal	None

- Current state:

```
If astigmatism = yes
    and tear production rate = normal
    and ?
then recommendation = hard
```

- Possible tests:

Age = Young	2/2
Age = Pre-presbyopic	1/2
Age = Presbyopic	1/2
Spectacle prescription = Myope	3/3
Spectacle prescription = Hypermetrope	1/3

- Tie between the first and the fourth test,
we choose the one with greater coverage

- Final rule:

```
If astigmatism = yes  
and tear production rate = normal  
and spectacle prescription = myope  
then recommendation = hard
```

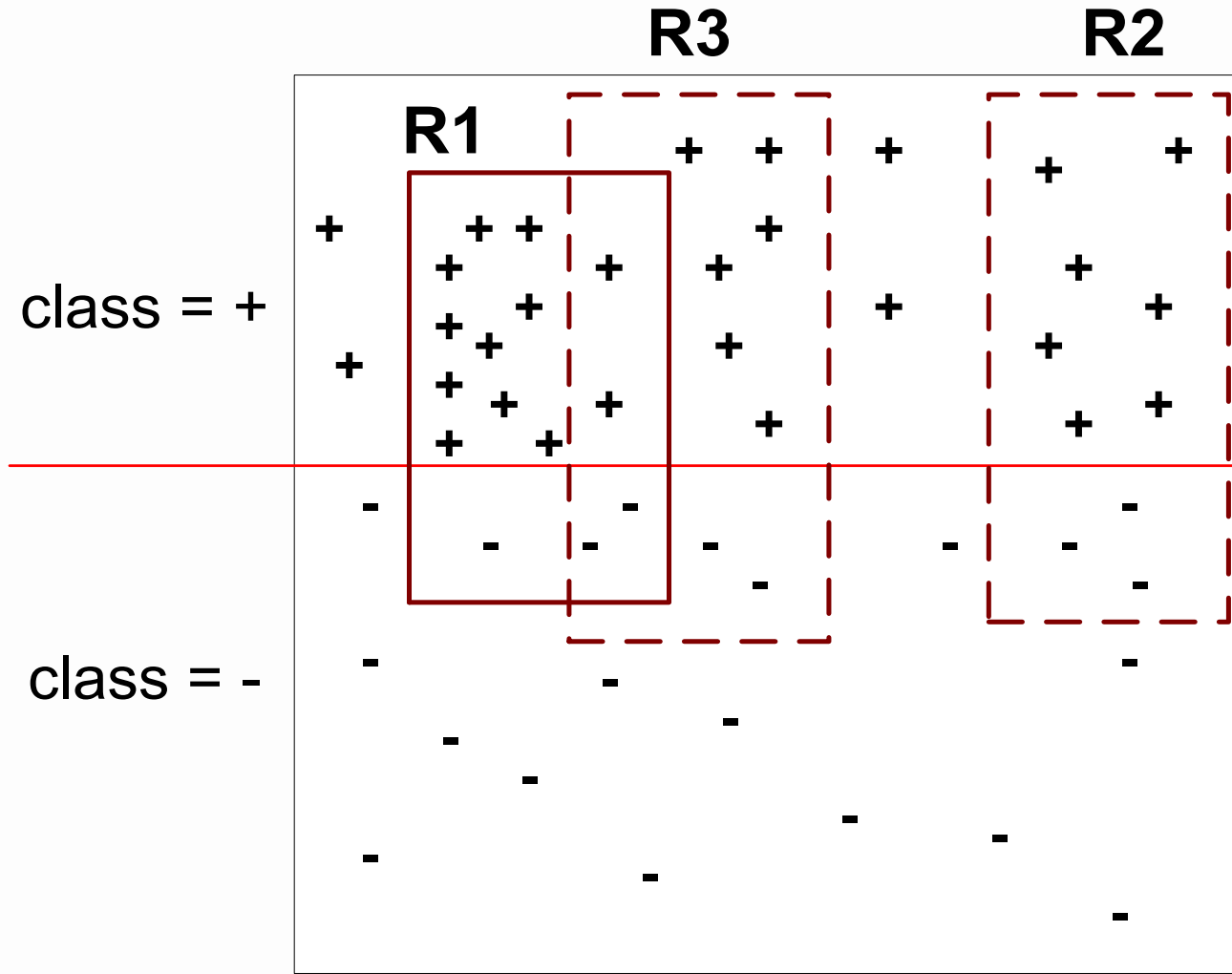
- Second rule for recommending “hard lenses”:
(built from instances not covered by first rule)

```
If age = young and astigmatism = yes  
and tear production rate = normal  
then recommendation = hard
```

- These two rules cover all “hard lenses”:
- Process is repeated with other two classes

- Measure 1: Accuracy (p/t)
 - t total instances covered by rule
 - p number of these that are positive
 - Produce rules that do not cover negative instances, as quickly as possible
 - May produce rules with very small coverage —special cases or noise?
- Measure 2: Information gain $p (\log(p/t) - \log(P/T))$
 - P and T the positive and total numbers before the new condition was added
 - Information gain emphasizes positive rather than negative instances
- These interact with the pruning mechanism used

- Why do we need to eliminate instances?
 - Otherwise, the next rule is identical to previous rule
- Why do we remove positive instances?
 - To ensure that the next rule is different
- Why do we remove negative instances?
 - Prevent underestimating accuracy of rule
 - Compare rules R2 and R3 in the following diagram

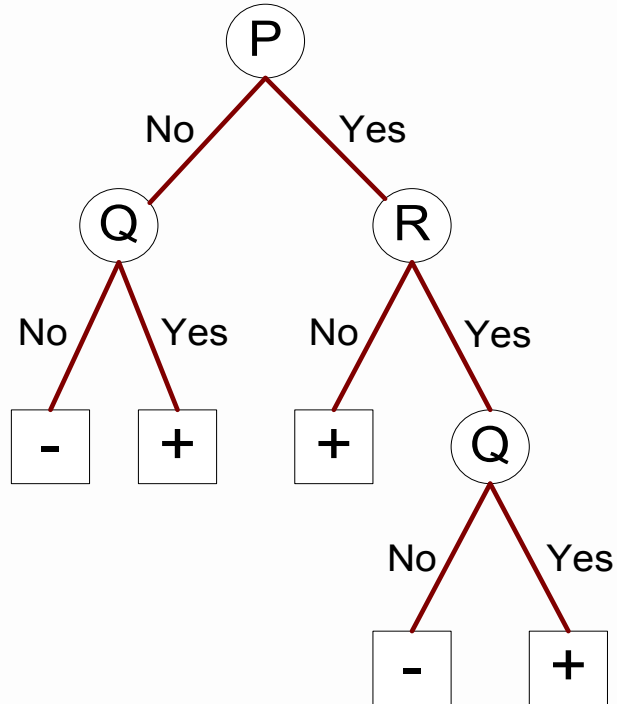


- Missing values usually fail the test
- Covering algorithm must either
 - Use other tests to separate out positive instances
 - Leave them uncovered until later in the process
- In some cases it is better to treat “missing” as a separate value
- Numeric attributes are treated as in decision trees

- The process usually stops when there is no significant improvement by adding the new rule
- Rule pruning is similar to post-pruning of decision trees
- Reduced Error Pruning:
 - Remove one of the conjuncts in the rule
 - Compare error rate on validation set
 - If error improves, prune the conjunct

- Rule sets can be more readable
- Decision trees suffer from replicated subtrees
- Rule sets are collections of local models, trees represent models over the whole domain
- The covering algorithm concentrates on one class at a time whereas decision tree learner takes all classes into account

Indirect Methods



Rule Set

r1: (P=No,Q=No) ==> -

r2: (P=No,Q=Yes) ==> +

r3: (P=Yes,R=No) ==> +

r4: (P=Yes,R=Yes,Q=No) ==> -

r5: (P=Yes,R=Yes,Q=Yes) ==> +

Summary

- Advantages of Rule-Based Classifiers
 - As highly expressive as decision trees
 - Easy to interpret
 - Easy to generate
 - Can classify new instances rapidly
 - Performance comparable to decision trees
- Two approaches: direct and indirect methods

- Direct Methods, typically apply sequential covering approach
 - Grow a single rule
 - Remove Instances from rule
 - Prune the rule (if necessary)
 - Add rule to Current Rule Set
 - Repeat
- Other approaches exist
 - Specific to general exploration (RISE)
 - Post processing of neural networks, association rules, decision trees, etc.

- Generate the rule set for the Weather dataset by repeatedly applying the procedure to learn one rule until no improvement can be produced or the covered examples are too few