



Classification: Rules

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

- ❑ Why rules?
- ❑ What are classification rules?
- ❑ What rules?
- ❑ What methods?
- ❑ Direct Methods
 - ▶ The OneRule algorithm
 - ▶ Sequential Covering Algorithms
 - ▶ The RISE algorithm
- ❑ Indirect Methods

Example

Example: to play or not to play?

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
```

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)

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)

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
```

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
- ❑ Which type of conditions?
 - ▶ Propositional, with attribute-value comparisons
 - ▶ First order Horn clauses, with variables

- ❑ Why rules?
 - ▶ One of the most **expressive** and most **human readable** representation for hypotheses is sets of IF-THEN rules

- Represent the knowledge in the form of IF-THEN rules
 - ▶ R: IF (humidity = high) and (outlook = sunny)
THEN play=no (3.0/0.0)
 - ▶ 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
 - ▶ $\text{coverage}(R) = n_{\text{covers}} / |\text{training data set}|$
 - ▶ $\text{accuracy}(R) = n_{\text{correct}} / n_{\text{covers}}$

- ❑ If more than one rule is 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 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

- ❑ Mutually exclusive rules
 - ▶ Classifier contains mutually exclusive rules if the rules are independent of each other
 - ▶ Every record is covered by at most one rule

- ❑ Exhaustive rules
 - ▶ Classifier has exhaustive coverage if it accounts for every possible combination of attribute values
 - ▶ Each record is covered by at least one rule

Just one rule

- ❑ 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

- ❑ 1R: learns a 1-level decision tree
 - ▶ I.e., rules that all test one particular attribute
 - ▶ Assumes nominal attributes
- ❑ 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

```
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
```

- Note: “missing” is treated as a separate attribute value

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

- ❑ Discretize numeric attributes
- ❑ Divide each attribute's range into intervals
- ❑ Sort instances according to attribute's values
- ❑ Place breakpoints where class changes (majority class)
- ❑ This minimizes the total error
- ❑ Example: temperature from weather data

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**

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
...

- ❑ This procedure is very sensitive to noise
- ❑ One instance with an incorrect class label will probably produce a separate interval
- ❑ Also: time stamp attribute will have zero errors
- ❑ Simple solution: enforce minimum number of instances in majority class per interval.
- ❑ As an example, with $\text{min} = 3$,

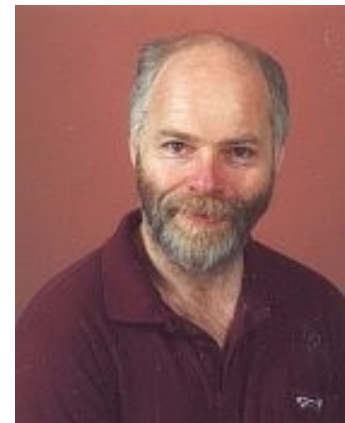
64	65	68	69	70	71	72	72	75	75	80	81	83	85	
Yes	⊘ No	⊘ Yes	Yes	Yes	No	No	Yes	⊘ Yes	Yes	Yes	No	⊘ Yes	Yes	⊘
No														

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

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → 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 → Yes	2/8	5/14
	True → No*	3/6	

- ❑ 1R was described in a paper by Holte (1993)
- ❑ Contains an experimental evaluation on 16 datasets (using cross-validation so that results were representative of performance on future data)
- ❑ Minimum number of instances was set to 6 after some experimentation
- ❑ 1R's simple rules performed not much worse than much more complex decision trees
- ❑ Simplicity first pays off!

Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. Robert C. Holte, Computer Science Department, University of Ottawa

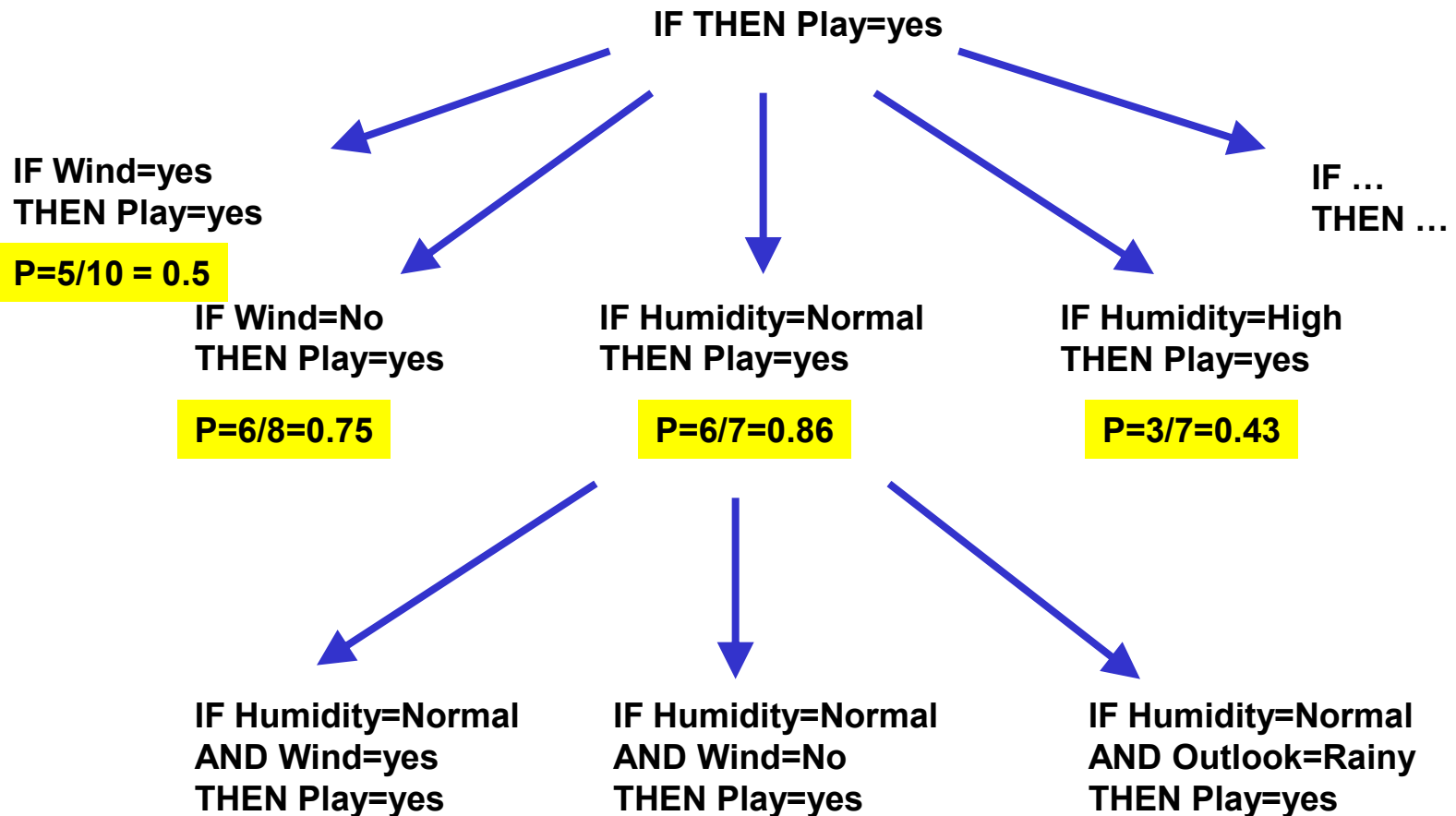


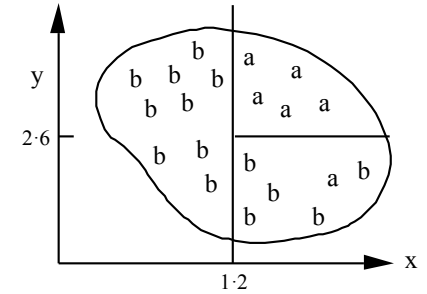
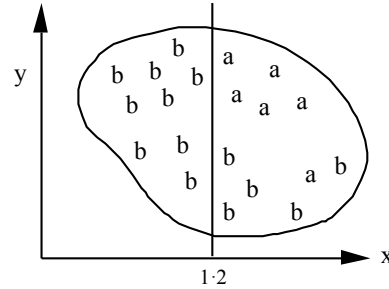
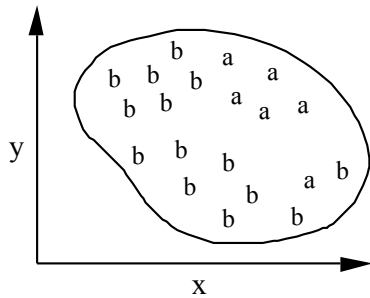
Sequential covering

- ❑ 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

1. LearnedRules := {}
 2. Rule := LearnOneRule(class, Attributes, Threshold)
 3. while(Performance(Rule, Examples) > Threshold)
 4. LearnedRules = LearnedRules + Rule
 5. Examples = Examples - Covered(Examples, Rule)
 6. Rule = LearnOneRule(class, Attributes, Examples)
 7. Sort(LearnedRules, Performance)
 8. Return LearnedRules
- The core of the algorithm is LearnOneRule

- ❑ General to Specific
 - ▶ Start with the most general hypothesis and then go on through specialization steps
- ❑ Specific to General
 - ▶ Start with the set of the most specific hypothesis and then go on through generalization steps

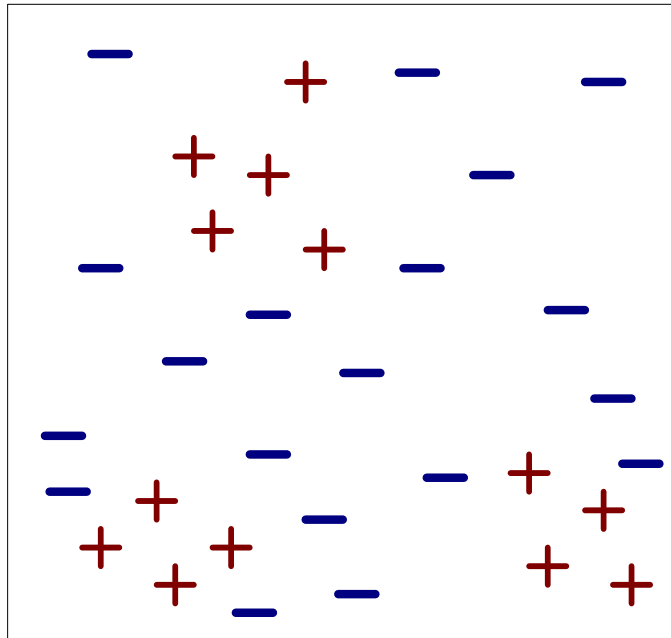




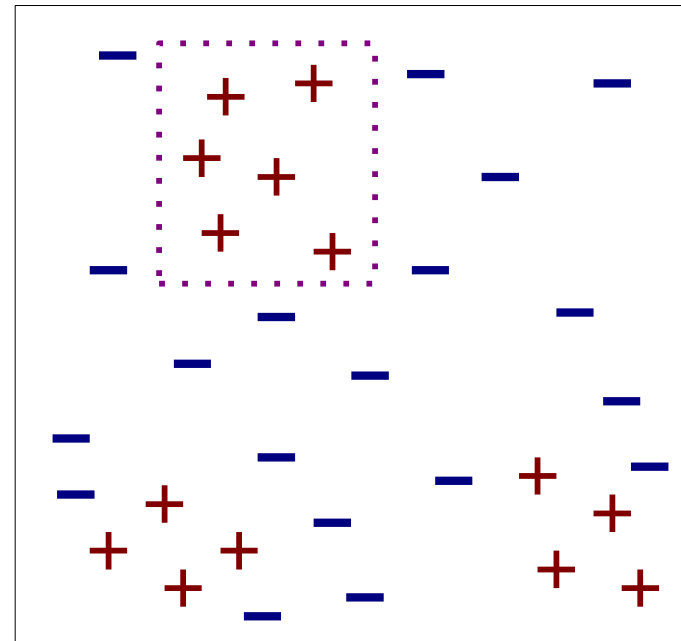
If true then class = a

If $x > 1.2$ then class = a

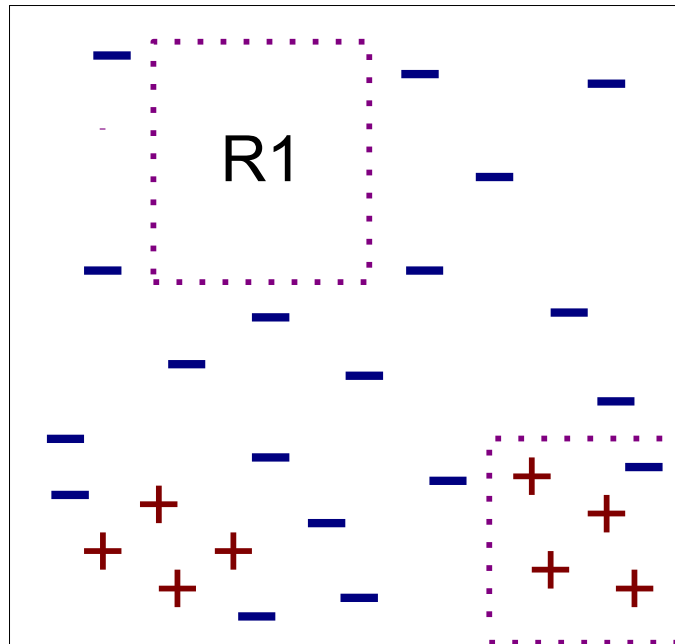
If $x > 1.2$ and $y > 2.6$ then class = a



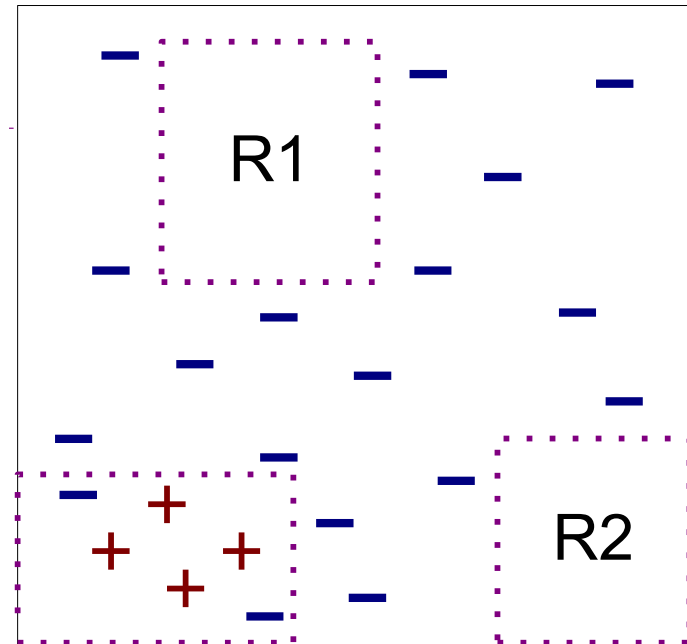
(i) Original Data



(ii) Step 1



(iii) Step 2



(iv) Step 3

```
Learn-one-rule(Target, Attributes, Example, 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
  Return a rule "IF BH THEN prediction"
```

- ❑ The algorithm to explore the hypothesis space is greedy and might tend to local optima
- ❑ To improve the exploration of the hypothesis space, we can **beam search**
- ❑ At each step **k** candidate hypotheses are considered.

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:

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

- ❑ 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)

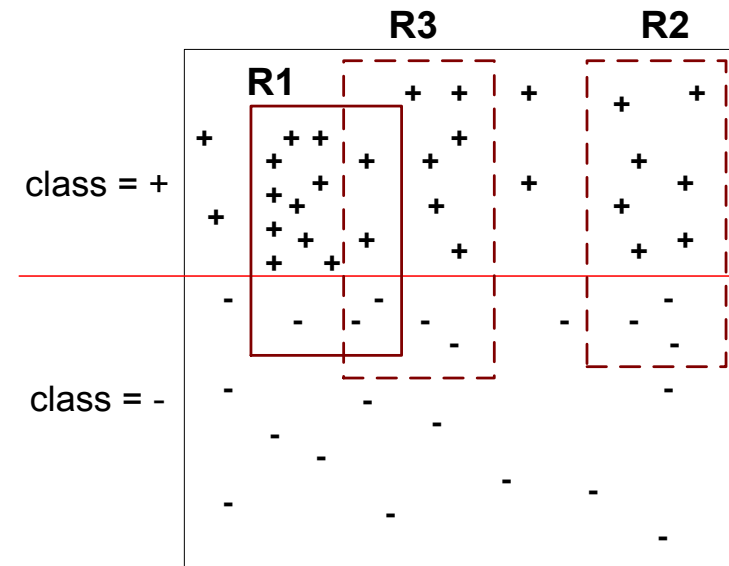
```
If age = young and astigmatism = yes
and tear production rate = normal
then recommendation = hard
```

- ❑ These two rules cover all "hard lenses":
- ❑ Process is repeated with other two classes

```
For each class C
  Initialize E to the instance set
  While E contains instances in class C
    Create a rule R with an empty left-hand side
      that predicts class C
    Until R is perfect (or there are no more attributes
      to use) do
      For each attribute A not mentioned in R,
        and each value v,
        Consider adding the condition A = v
          to the left-hand side of R
        Select A and v to maximize the accuracy p/t
          (break ties by choosing the condition
            with the largest p)
      Add A = v to R
    Remove the instances covered by R from E
```

- ❑ Basic covering algorithm:
 - ▶ keep adding conditions to a rule to improve its accuracy
 - ▶ Add the condition that improves accuracy the most
- ❑ Measure 1: p/t
 - ▶ t total instances covered by rule
number of these that are positive
 - ▶ Produce rules that don't 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?
 - ▶ 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 diagram



- ❑ Common treatment of missing values: for any test, they fail
- ❑ 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 just like they are in decision trees

Pruning

- ❑ Stopping criterion
 - ▶ Compute the gain
 - ▶ If gain is not significant, discard the new rule

- ❑ Rule Pruning: Similar to post-pruning of decision trees

- ❑ Reduced Error Pruning:
 - ▶ Remove one of the conjuncts in the rule
 - ▶ Compare error rate on validation set before and after pruning
 - ▶ If error improves, prune the conjunct

- ❑ Two main strategies:
 - ▶ Incremental pruning
 - ▶ Global pruning

- ❑ Other difference: pruning criterion
 - ▶ Error on hold-out set (reduced-error pruning)
 - ▶ Statistical significance
 - ▶ MDL principle

- ❑ Also: post-pruning vs. pre-pruning

- ❑ Rules are no longer mutually exclusive
 - ▶ A record may trigger more than one rule
 - ▶ Solution?
 - Ordered rule set
 - Unordered rule set – use voting schemes

- ❑ Rules are no longer exhaustive
 - ▶ A record may not trigger any rules
 - ▶ Solution?
 - Use a default class

- ❑ Rule-based ordering
 - ▶ Individual rules are ranked based on their quality
- ❑ Class-based ordering
 - ▶ Rules that belong to the same class appear together

Rule-based Ordering

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

Class-based Ordering

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Married}) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

RISE Algorithm

- ❑ It works in a specific-to-general approach
- ❑ Initially, it creates one rule for each training example
- ❑ Then it goes on through elementary generalization steps until the overall accuracy does not decrease

Input: ES is the training set

Let RS be ES

Compute $\text{Acc}(\text{RS})$

Repeat

 For each rule R in RS,

 find the nearest example E not covered by R

 (E is of the same class as R)

$R' = \text{MostSpecificGeneralization}(R, E)$

$\text{RS}' = \text{RS}$ with R replaced by R'

 if $(\text{Acc}(\text{RS}') \geq \text{Acc}(\text{RS}))$ then

$\text{RS} = \text{RS}'$

 if R' is identical to another rule in RS

 then, delete R' from RS

Until no increase in $\text{Acc}(\text{RS})$ is obtained

RIPPER

- ❑ Start from an empty conjunct: $\{\}$
- ❑ Add conjuncts that minimizes the entropy measure:
 $\{A\}, \{A,B\}, \dots$
- ❑ Determine the rule consequent by taking majority class of instances covered by the rule

- ❑ For 2-class problem, choose one of the classes as positive class, and the other as negative class
 - ▶ Learn rules for positive class
 - ▶ Negative class will be default class
- ❑ For multi-class problem
 - ▶ Order the classes according to increasing class prevalence (fraction of instances that belong to a particular class)
 - ▶ Learn the rule set for smallest class first, treat the rest as negative class
 - ▶ Repeat with next smallest class as positive class

- Growing a rule:
 - ▶ Start from empty rule
 - ▶ Add conjuncts as long as they improve FOIL's information gain
 - ▶ Stop when rule no longer covers negative examples
 - ▶ Prune the rule immediately using incremental reduced error pruning
 - ▶ Measure for pruning: $v = (p-n)/(p+n)$
 - p: number of positive examples covered by the rule in the validation set
 - n: number of negative examples covered by the rule in the validation set
 - ▶ Pruning method: delete any final sequence of conditions that maximizes v

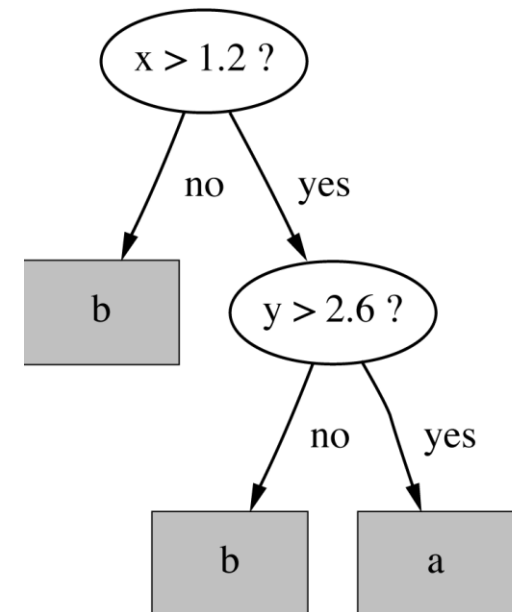
□ Building a Rule Set:

- ▶ Use sequential covering algorithm
 - Finds the best rule that covers the current set of positive examples
 - Eliminate both positive and negative examples covered by the rule
- ▶ Each time a rule is added to the rule set, compute the new description length
 - stop adding new rules when the new description length is d bits longer than the smallest description length obtained so far

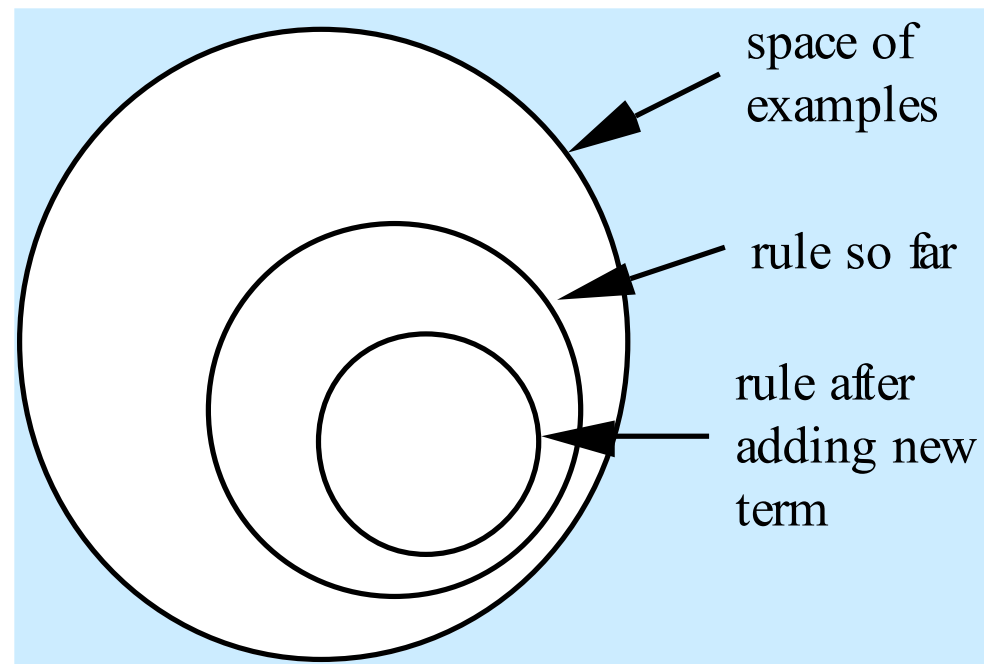
- Optimize the rule set:
 - ▶ For each rule r in the rule set R
 - Consider 2 alternative rules:
 - Replacement rule (r^*): grow new rule from scratch
 - Revised rule(r'): add conjuncts to extend the rule r
 - Compare the rule set for r against the rule set for r^* and r'
 - Choose rule set that minimizes MDL principle
 - ▶ Repeat rule generation and rule optimization for the remaining positive examples

Rules vs. trees

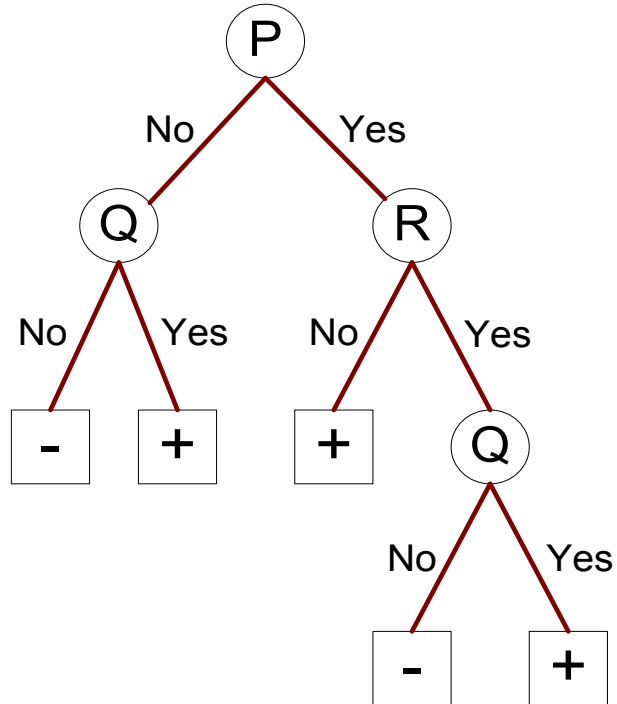
- ❑ Corresponding decision tree:
produces exactly the same predictions
- ❑ But rule sets can be more clear
when decision trees suffer from replicated subtrees
- ❑ Also, in multi-class situations, covering algorithm
concentrates on one class at a time whereas
decision tree learner takes all classes into account



- ❑ Sequential covering generates a rule by adding tests that maximize rule's accuracy
- ❑ Similar to situation in decision trees: problem of selecting an attribute to split on
- ❑ But decision tree inducer maximizes overall purity
- ❑ Each new test reduces rule's coverage



Indirect methods



Rule Set

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

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

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

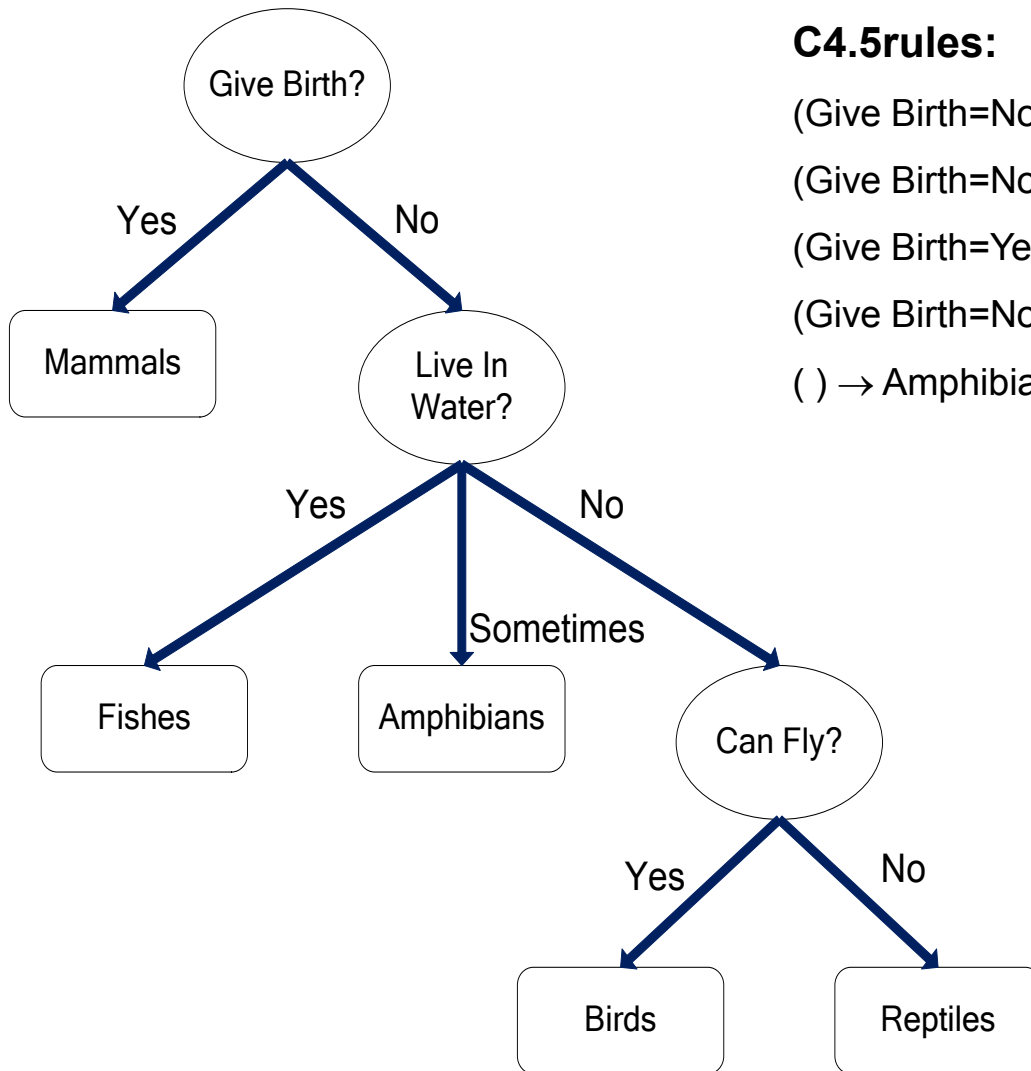
r4: (P=Yes,R=Yes,Q=No) ==> -

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

- ❑ Extract rules from an unpruned decision tree
- ❑ For each rule, $r: A \rightarrow y$,
 - ▶ Consider an alternative rule $r': A' \rightarrow y$ where A' is obtained by removing one of the conjuncts in A
 - ▶ Compare the pessimistic error rate for r against all r' 's
 - ▶ Prune if one of the r' 's has lower pessimistic error rate
 - ▶ Repeat until we can no longer improve generalization error

- ❑ Instead of ordering the rules, order subsets of rules (class ordering)
- ❑ Each subset is a collection of rules with the same rule consequent (class)
- ❑ Compute description length of each subset
 - ▶ Description length = $L(\text{error}) + g L(\text{model})$
 - ▶ g is a parameter that takes into account the presence of redundant attributes in a rule set (default value = 0.5)

Name	Give Birth	Lay Eggs	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	no	yes	mammals
python	no	yes	no	no	no	reptiles
salmon	no	yes	no	yes	no	fishes
whale	yes	no	no	yes	no	mammals
frog	no	yes	no	sometimes	yes	amphibians
komodo	no	yes	no	no	yes	reptiles
bat	yes	no	yes	no	yes	mammals
pigeon	no	yes	yes	no	yes	birds
cat	yes	no	no	no	yes	mammals
leopard shark	yes	no	no	yes	no	fishes
turtle	no	yes	no	sometimes	yes	reptiles
penguin	no	yes	no	sometimes	yes	birds
porcupine	yes	no	no	no	yes	mammals
eel	no	yes	no	yes	no	fishes
salamander	no	yes	no	sometimes	yes	amphibians
gila monster	no	yes	no	no	yes	reptiles
platypus	no	yes	no	no	yes	mammals
owl	no	yes	yes	no	yes	birds
dolphin	yes	no	no	yes	no	mammals
eagle	no	yes	yes	no	yes	birds



C4.5rules:

(Give Birth=No, Can Fly=Yes) → Birds

(Give Birth=No, Live in Water=Yes) → Fishes

(Give Birth=Yes) → Mammals

(Give Birth=No, Can Fly=No, Live in Water=No) → Reptiles

() → Amphibians

RIPPER:

(Live in Water=Yes) → Fishes

(Have Legs=No) → Reptiles

(Give Birth=No, Can Fly=No, Live In Water=No) → Reptiles

(Can Fly=Yes, Give Birth=No) → Birds

() → Mammals

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.