POLITECNICO DI MILANO

Classification: Rule Induction

Information Retrieval and Data Mining

Prof. Matteo Matteucci

What is Rule Induction?

The Weather Dataset

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

A Rule Set to Classify the Data

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

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

OTHERWISE play=yes (9.0/9.0)

- Corresponding Confusion Matrix

	Classified Yes	Classified No
TRUE YES	9	0
TRUE NO	0	5

Let's Check the First 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 (humidity = high) and (outlook = sunny) THEN play=no (3.0/3.0)

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 (2.0/2.0)

Finally, The Third Rule

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 (9.0/9.0)

A Simpler Solution

IF (outlook = sunny) THEN play IS no

ELSE IF (outlook = overcast) THEN play IS yes

ELSE IF (outlook = rainy) THEN play IS yes

- Corresponding Confusion Matrix

	Classified Yes	Classified No
TRUE YES	4	5
TRUE NO	3	2

- Out of 14 only 6 instances are correct …

What is a Classification Rule?

What Is a Classification Rules? Why Rules?

- A Classification rule is an IF-THEN rule where
  - The IF part states a condition over the data
  - The THEN part includes a class label

```
IF (lotion = no) THEN sunburn
```

- What types of conditions?
  - Propositional, with attribute-value comparisons

$$(\text{Overcast} = \text{sunny}) \wedge (\text{Temperature} > 30) \rightarrow \text{PlayGolf}$$

  - First order Horn clauses, with variables

$$\text{father}(y,x) \wedge \text{female}(x) \rightarrow \text{daughter}(x,y)$$

- The most expressive and most human readable representation for hypotheses is a sets of IF-THEN rules

Coverage and Accuracy

IF (humidity = high) and (outlook = sunny)
THEN play=no (ncorrect/ncovers)

- ncorrect = number of examples correctly classified by the rule
- ncovers = number of examples covered by the rule

- Coverage(R) = ncovers /size of the training data set
- Accuracy(R) = ncorrect /ncovers

Conflict Resolution

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

IF (outlook = rainy) and (windy = TRUE)
THEN play=no (2.0/3.0)

OTHERWISE play=yes (9.0/9.0)

- If more than one rule is triggered, we need conflict resolution
  - Size ordering: assign the highest priority to the triggering rules that has 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 quality or by experts)

Rule Learning

Two Approaches for Rule Learning

- Direct Methods
  - Directly learn the rules from the training data
    - OneRule for each rule
    - Sequential Covering Algorithm
- Indirect Methods
  - Learn decision tree, then convert to rules
    - Full tree grown
    - Extract all rules
    - Rule pruning

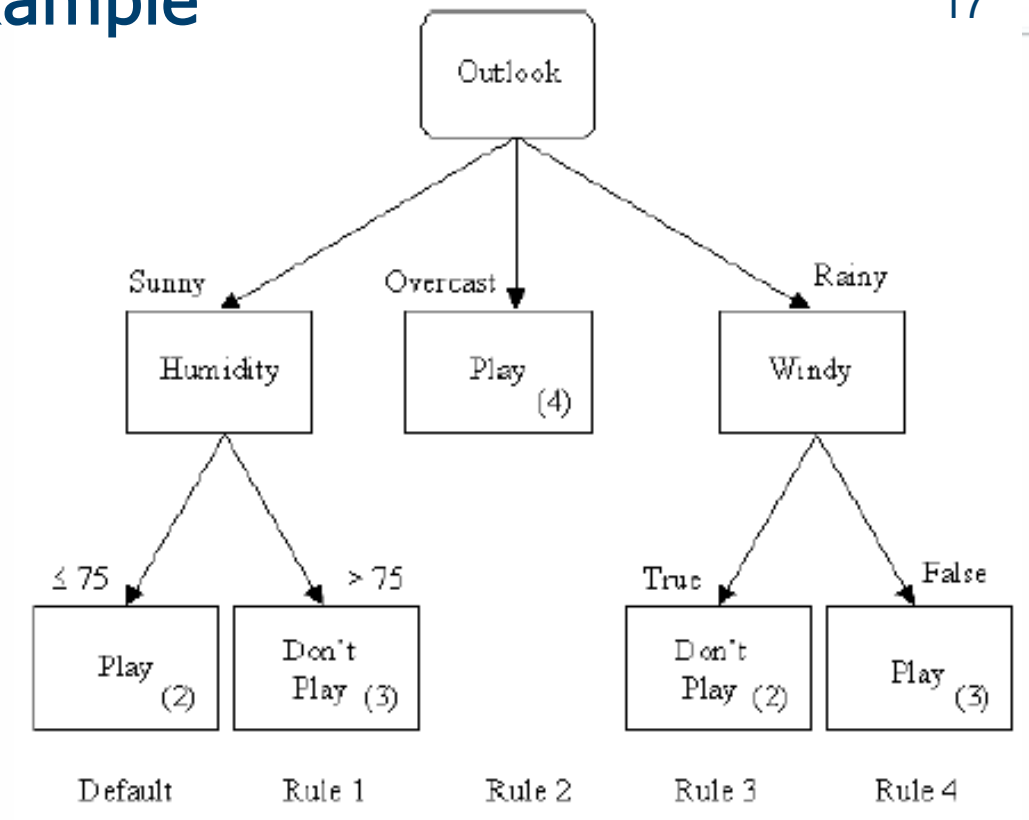
Indirect Methods

Rule Set from a Tree

Trace each path in the decision tree, from root node to leaf node, recording the test outcomes as antecedents and the leaf-node classification as the consequent.

- Allows distinguishing among the different contexts in which a decision node is used
  - pruning decision regarding an attribute test can be made differently for each path.
  - if the tree itself were pruned, the only two choices would be remove the decision node completely, or retain it
- Removes the distinction between attribute tests that occur near the root of the tree and those that occur near the leaves
- Converting to rules improves readability.

Weather Example



1. if (outlook = overcast) then play
2. if (outlook = rain) and (windy = false) then play
3. if (outlook = sunny) and (humidity = high) then don't play
4. if (outlook = rain) and (windy = true) then don't play
5. default class: play

Rule Pruning

1. Eliminate unnecessary rule antecedents to simplify the rules (eliminate antecedents that have no effect on the conclusion using a test for independency)
2. Eliminate unnecessary rules to simplify the rule set.
3. Replace those rules that share the most common consequent by a default rule that is triggered when no other is triggered.

We can use the following independence tests:

- Chi-Square (cell frequencies m > 10): $\chi^2 = \sum_i \sum_j \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$
- Yates' Correction (5 ≤ m ≤ 10): $\chi^2 = \sum_i \sum_j \frac{(|o_{ij} - e_{ij}| - 0.5)^2}{e_{ij}}$
- Fisher's Exact Test (m < 5): *see Winston, pp. 437-442*

Chi Squared Test of Independence

	C_1	C_2	
R_1	x_{11}	x_{12}	$R_{1T} = x_{11} + x_{12}$
R_2	x_{21}	x_{22}	$R_{2T} = x_{21} + x_{22}$
	$C_{T1} = x_{11} + x_{21}$	$C_{T2} = x_{12} + x_{22}$	$T = x_{11} + x_{12} + x_{21} + x_{22}$

- R_1 and R_2 represent the Boolean states of an antecedent for the conclusions C_1 and C_2 (C_2 is the negation of C_1)
- x_{11}, x_{12}, x_{21}, and x_{22} represent the frequencies of each antecedent-consequent pair.
- R_{1T}, R_{2T}, C_{T1}, and C_{T2} are the marginal sums of the rows and columns, respectively.

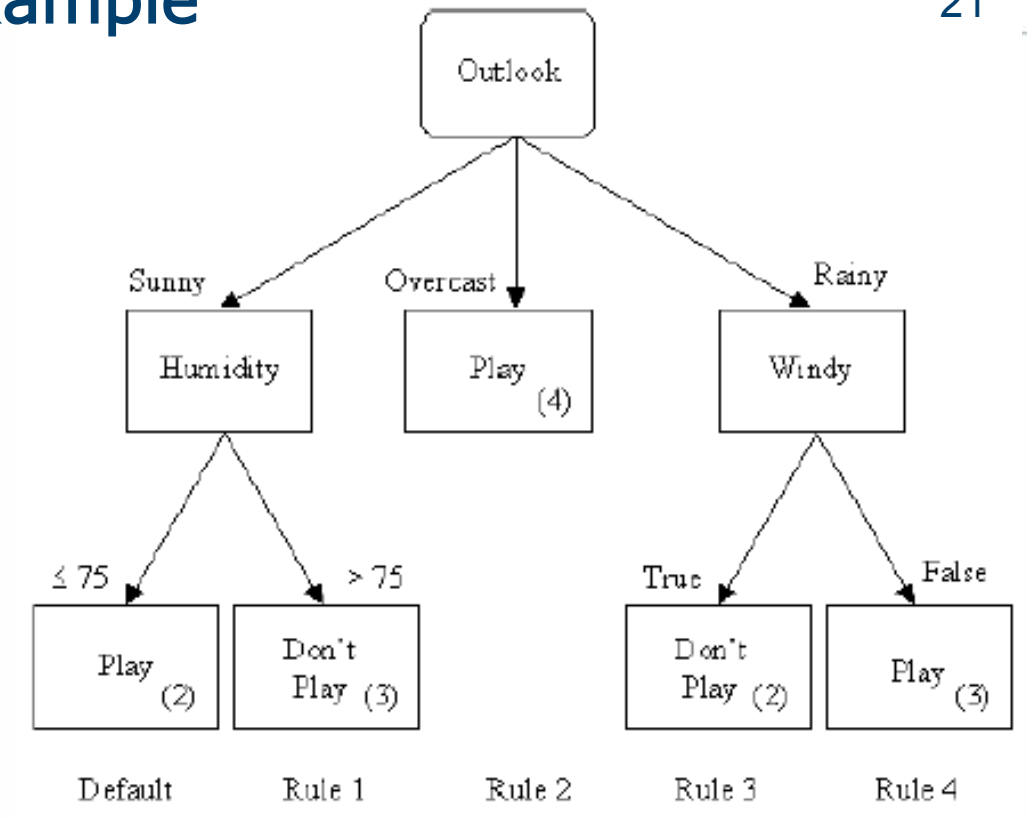
The marginal sums and T, the total frequency of the table, are used to calculate expected cell values in test for independence.

Test of Independence

Given a contingency table of dimensions r by c (rows x columns):

1. Calculate the marginal sums.
2. Calculate the total frequency, T, using the marginal sums.
3. Calculate expected frequencies for each cell: $e_{ij} = R_{iT} \cdot C_{Tj}/T$
4. Select the test based on highest expected frequency m
5. Calculate χ^2 using the chosen test
6. Calculate the degrees of freedom: $df = (r-1)(c-1)$
7. Use a chi-square table with χ^2 and df to determine if the conclusions are independent from the antecedent at the selected level of significance (usually $\alpha = 0.05$).
   - $\chi^2 > \chi^2_\alpha$ reject the null hypothesis (keep antecedents)
   - $\chi^2 \leq \chi^2_\alpha$ accept the null hypothesis (discard antecedents)

Weather Example



1. if (outlook = overcast) then play
2. if (outlook = rain) and (windy = false) then play
3. if (outlook = sunny) and (humidity = high) then don't play
4. if (outlook = rain) and (windy = true) then don't play
5. default class: play

The Weather Dataset

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

A Complete Example

Name	Hair	Height	Weight	Lotion	Result
Sarah	blonde	average	light	no	sunburned
Dana	blonde	tall	average	yes	none
Alex	brown	short	average	yes	none
Annie	blonde	short	average	no	sunburned
Emily	red	average	heavy	no	sunburned
Pete	brown	tall	heavy	no	none
John	brown	average	heavy	no	none
Katie	blonde	short	light	yes	none

- Can you set up a Decision Tree to predict if some one will get a sunburn?
- Can you prune the resulting tree into a few simple rules?

Direct Methods

One Rule

Inferring Rudimentary Rules

- OneRule (1R) learns a simple rule involving one attribute
  - Assumes nominal attributes
  - The rule evaluates 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
- ..

Pseudo-Code for OneRule


```
For each attribute:
        For each value of the attribute:

                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
```

Evaluating the Weather Attributes

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

OneRule and Numerical Attributes

- 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

OneRule and Numerical Attributes

- 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 → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temperature	≤ 77.5 → Yes	3/10	5/14
	> 77.5 → No*	2/4	
Humidity	≤ 82.5 → Yes	1/7	3/14
	> 82.5 and ≤ 95.5 → No	2/6	
	> 95.5 → Yes	0/1	
Windy	False → Yes	2/8	5/14
	True → No*	3/6	

Sequential Covering Algorithm

Sequential Covering Algorithms

Sequential Covering Algorithm learns a set of highly accurate rules for one class … not just the simple ones from 1R!

1. Consider the set E of positive (or negative) examples
2. Repeat Until all the examples are covered
    1. Learn one rule with high accuracy, any coverage
    2. Remove positive examples covered by this rule

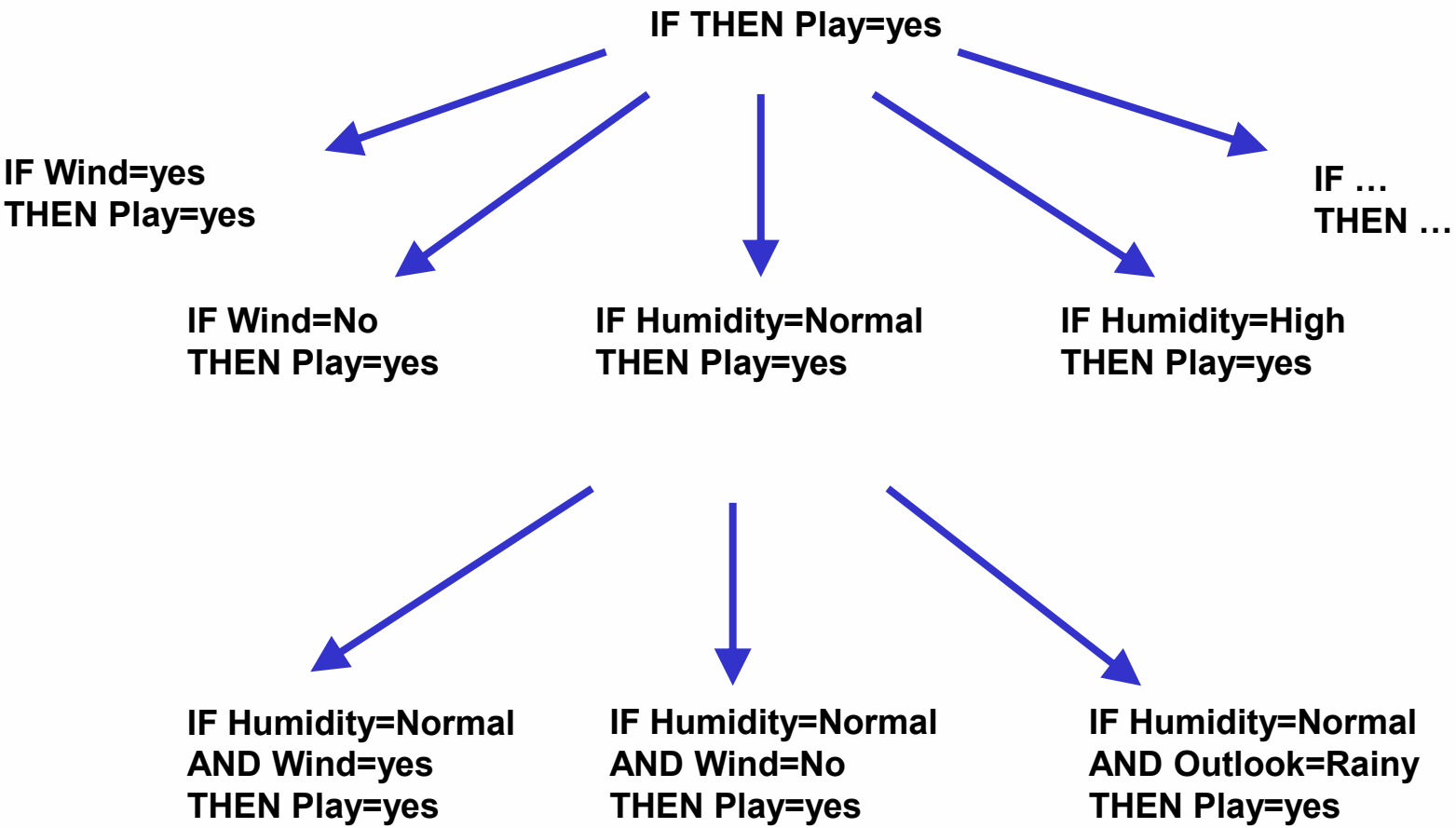
Basic Sequential Covering Algorithm


```
Sequential_Covering(class, attributes, examples, threshold)
{
        LearnedRules:={};
        Rule:=Learn_One_Rule(class,attribute,examples);
        while(Performance(Rule,examples)>threshold)
        {
                LearnedRules:=LearnedRules+Rule;
                examples:=examples-CoveredExamples(examples,Rule);
                Rule:=Learn_One_Rule(class,attribute,examples);
        }
        Sort(LearnedRules,Performance);
        return LearnedRules;
}
```

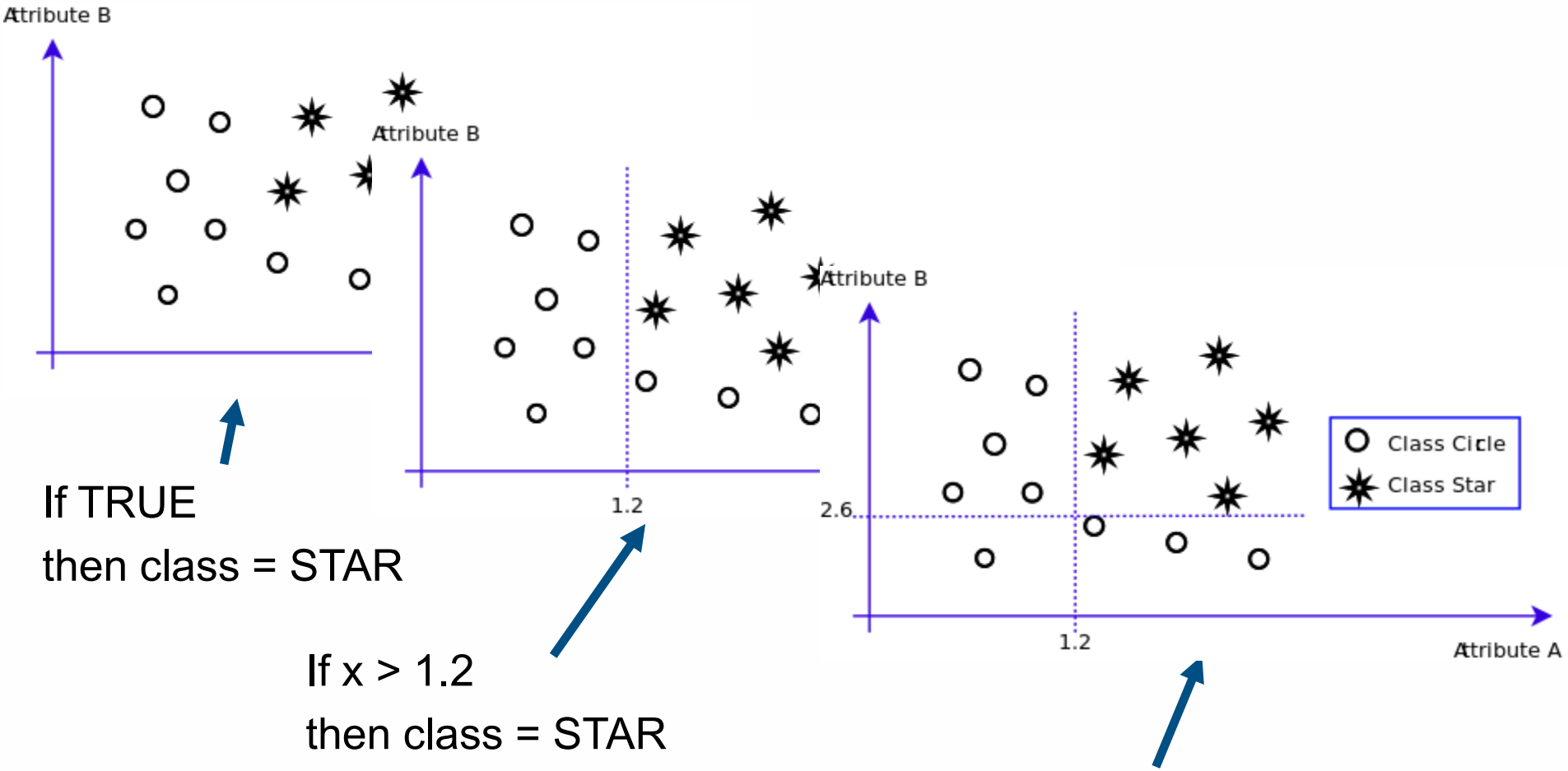
Learning One Rule


```
Learn_One_Rule(class, attribute, examples);
{
        NewRule:= Most_General_Rule();
        NegativeExamples:= Evalute_Negative_Examples(NewRule);
        while (NegativeExamples)
        {
                Cadidates:=Generate(attribute,NegativeExamples);
                BestLiteral:= argmax (L in Candidates) of
                                Performance(Specialize_Rule(NewRule,L));
                NewRule:=Specialize_Rule(NewRule,BestLiteral);
        NegativeExamples:= Evalue_Negative_Examples(NewRule);
        }
        return NewRule;
}
```

Finding the Best Rule Possible



The STAR Dataset

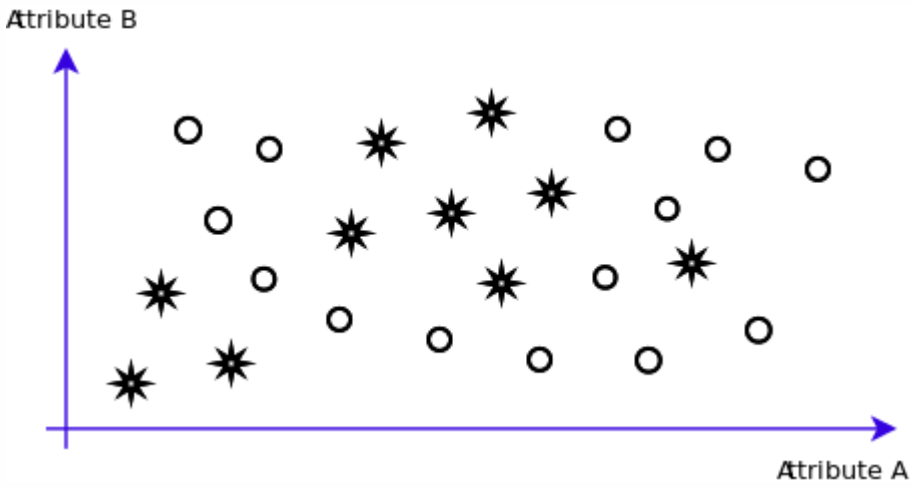


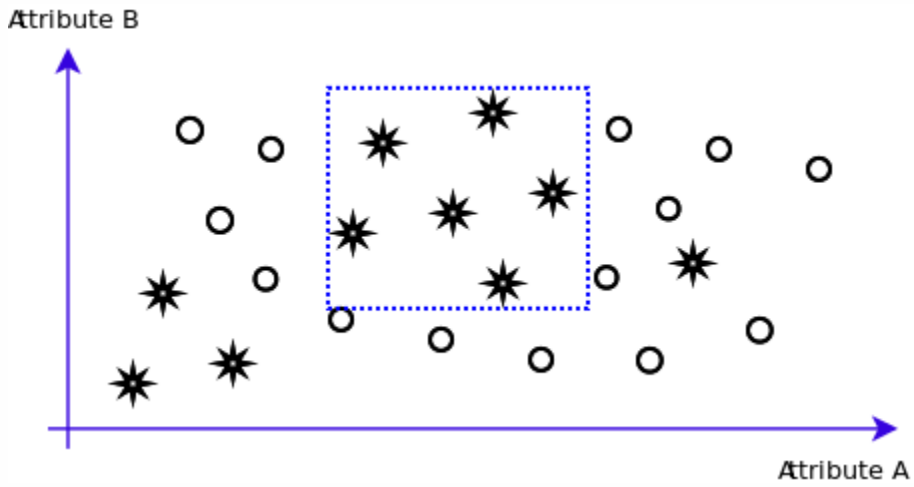
If TRUE
then class = STAR

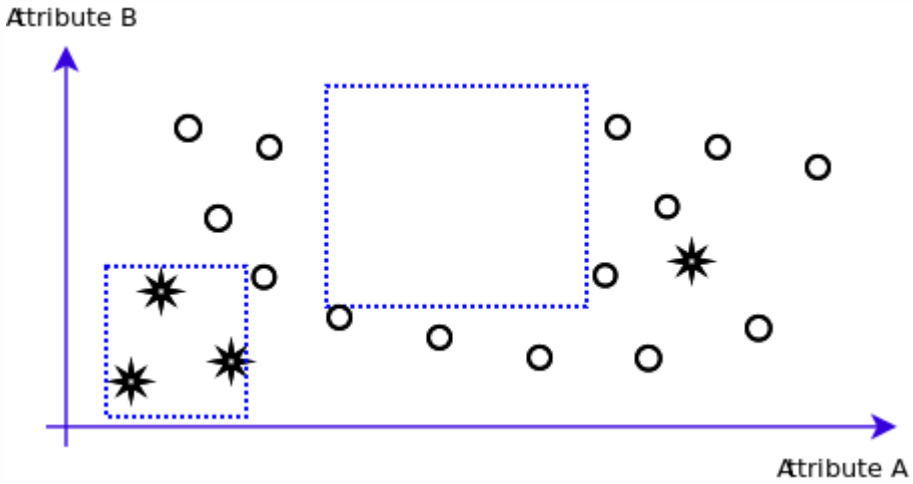
If x > 1.2
then class = STAR

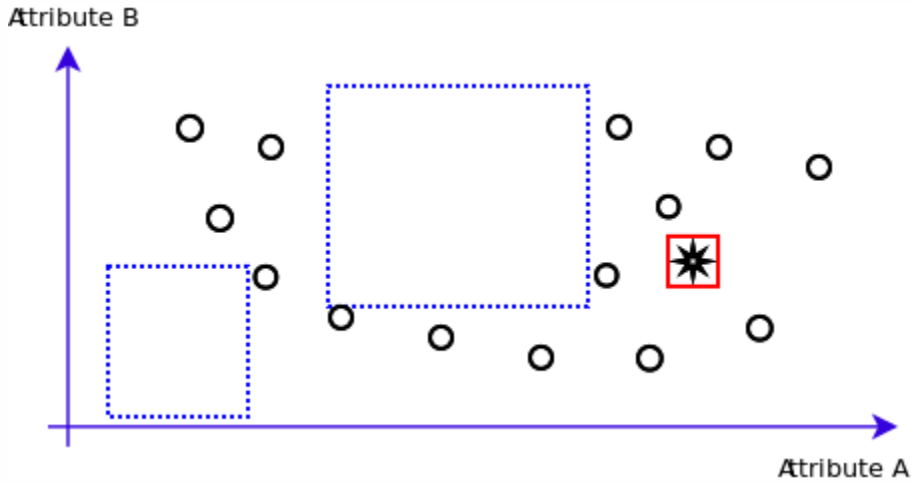
If x > 1.2 and y > 2.6
then class = STAR

Sequential Covering … for STAR









An Example Using Contact Lens Data

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

First Step: the Most General Rule

- 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

Adding the First Clause

- 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

Extending the First Rule

- 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

The Second Rule

- 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

Adding the Third Clause

- 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

The Final Result

- 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

Selecting Attribute-Value Pairs

- Training accuracy on the training example (AQ15): p/t
  - t instances covered by rule, p number of positive
  - Produce rules for positive instances as quickly as possible
  - May produce rules with very small coverage
- Information gain (CN2, PRISM): log2 p/t − log2 P/T
  - P and T are the positive and total number of examples that satisfied the previous rule
  - Equivalent to p/t
- Information gain with coverage (FOIL): p(log2 p/t − log2 P/T)
  - P and T are the positive and total number of examples that satisfied the previous rule
  - *Coverage* is also considered in the evaluation.

Stopping Criterion and Rule Pruning

- The process usually stops when there is no significant improvement by adding the new rule
- Also post-pruning could be applyed 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