Rule-Based Classification

Sohrab Hossain

CSE, EDU

Basics

- learned model is represented as a set of IF-THEN rules.
- Rules are a good way of representing information or bits of knowledge.
- A **rule-based classifier uses a set of IF-THEN rules for classification.**
    - **IF *condition* *THEN* *conclusion***

An example

- *R1: IF age = youth AND student = yes THEN buys computer = yes.*
- LHS: **rule antecedent or precondition.**
- RHS: **rule consequent.**

❑ In the rule antecedent, the condition consists of one or more *attribute tests (e.g., age = youth and student = yes)* that are logically ANDed

- The rule’s consequent contains a class prediction (in this case, we are predicting whether a customer will buy a computer)

$$R1: (age = youth) \wedge (student = yes) \Rightarrow (buys\_computer = yes).$$

If the condition (i.e., all the attribute tests) in a rule antecedent holds true for a given tuple, we say that the rule antecedent is **satisfied** & that the rule **covers the tuple.**

Rule: (Condition) → y

- Condition is a conjunction of attribute tests

  $(A_1 = v_1)$ and $(A_2 = v_2)$ and … and $(A_n = v_n)$

  - y is the class label

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

A Rule Set to Classify the Data

- 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 rules…

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)

Let's check the rules...

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

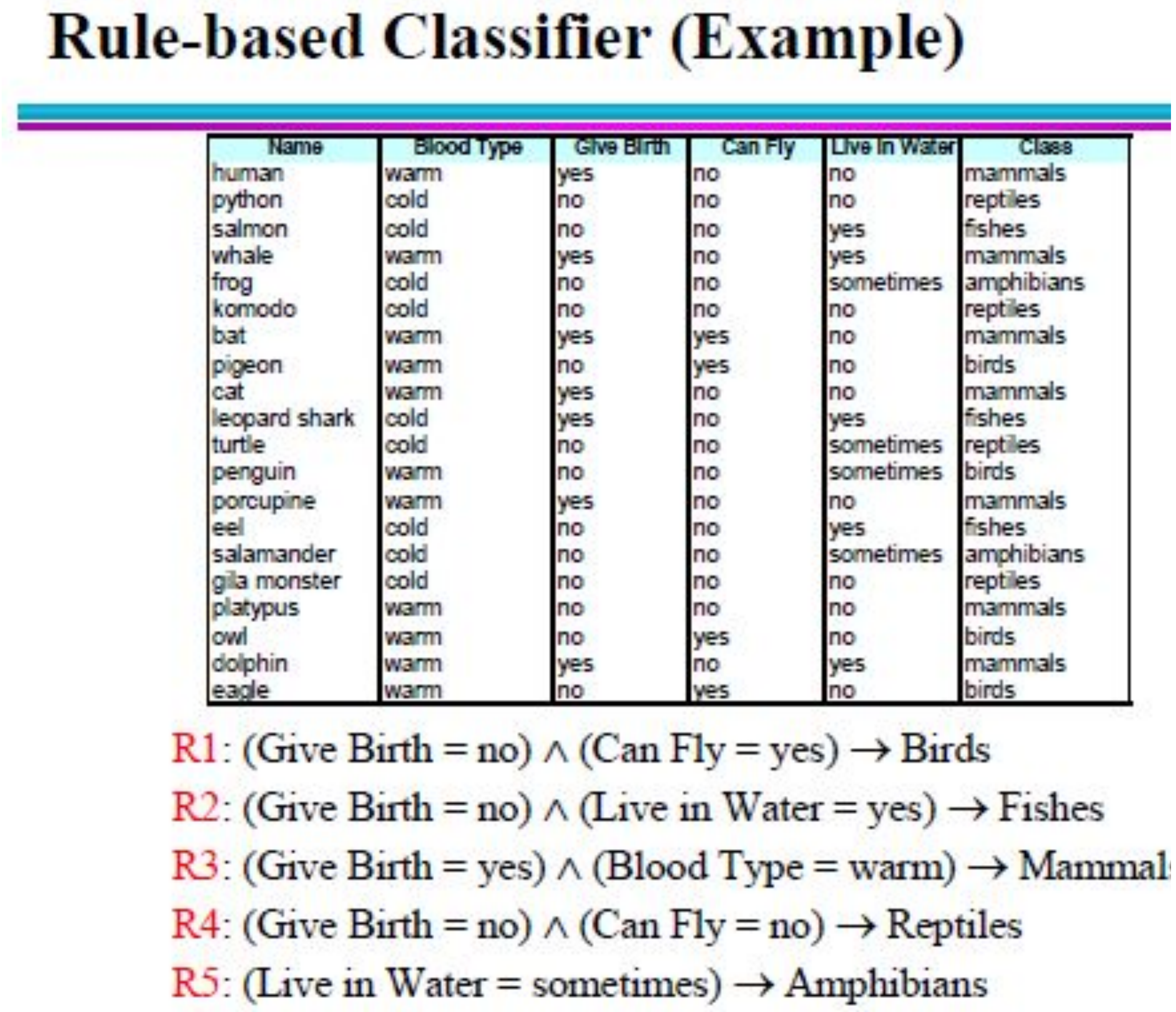
IF (outlook = rainy) and (windy = TRUE) THEN play=no (2.0/0.0)

Let’s check the rules…

Outlook	Temp	Humidity	Windy	Play
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Overcast	Cool	Normal	True	Yes
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
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
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

OTHERWISE play=yes (9.0/0.0)

Another Example



Rule-based Classifier (Example)

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

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammal

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

What are 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

Assessment of Rules

- Assessment of a rule: coverage and accuracy
  - ncovers = # of tuples covered by R
  - ncorrect = # of tuples correctly classified by R
  - coverage(R) = ncovers /|training data set|
  - accuracy(R) = ncorrect / ncovers

- a rule's coverage is the percentage of tuples that are covered by the rule (i.e., their attribute values hold true for the rule's antecedent).
- For a rule's accuracy, we look at the tuples that it covers and see what percentage of them the rule can correctly classify.

$$coverage(R) = \frac{n_{covers}}{|D|}$$

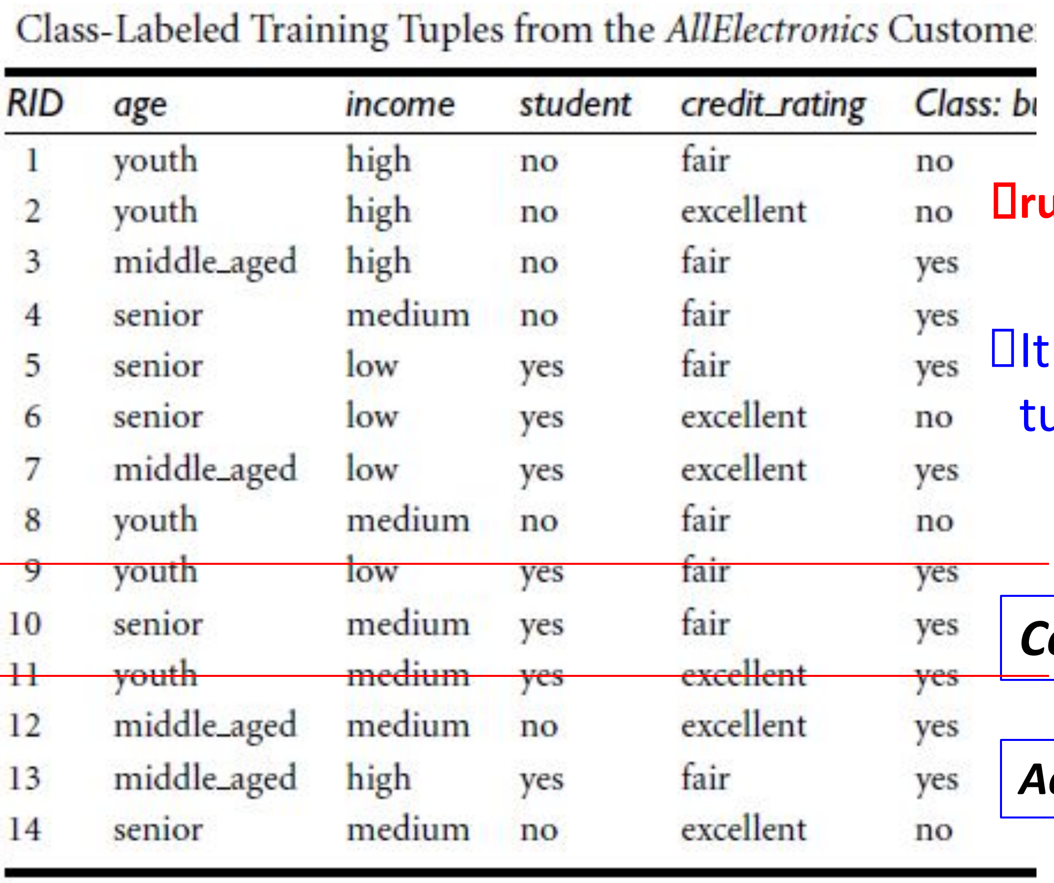
$$accuracy(R) = \frac{n_{correct}}{n_{covers}}.$$

Example

R1: $(age = youth) \wedge (student = yes) \Rightarrow (buys\_computer = yes)$.

$$coverage(R) = \frac{n_{covers}}{|D|}$$

$$accuracy(R) = \frac{n_{correct}}{n_{covers}}.$$



Class-Labeled Training Tuples from the *AllElectronics* Custome

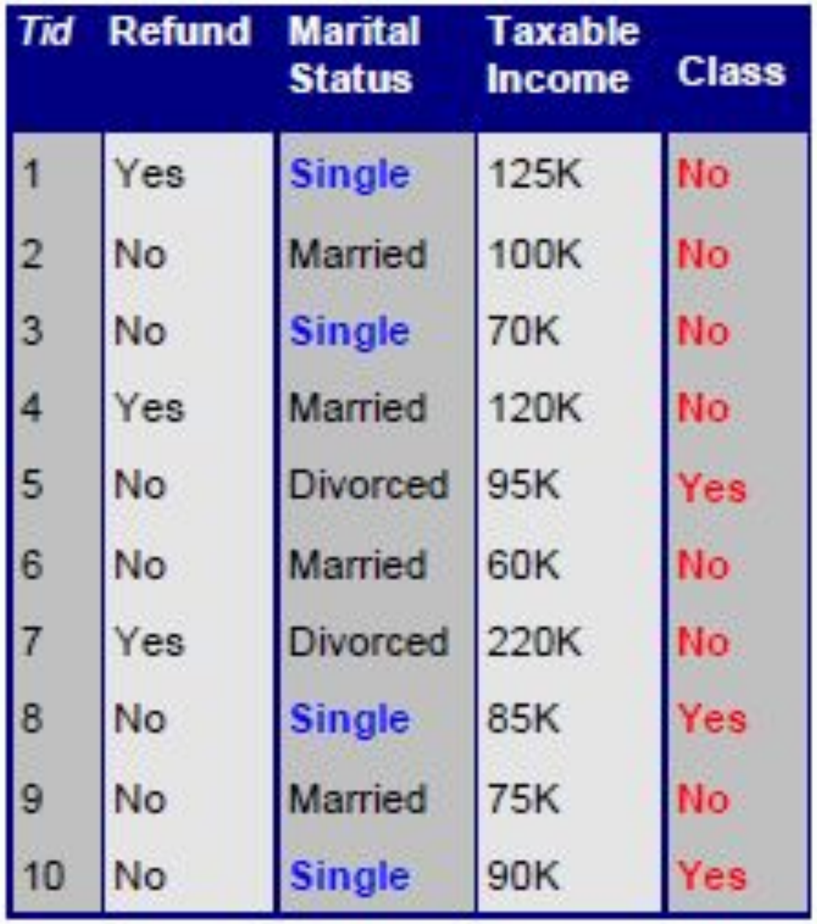
RID	age	income	student	credit_rating	Class: bu
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
~~9~~	~~youth~~	~~low~~	~~yes~~	~~fair~~	~~yes~~
10	senior	medium	yes	fair	yes
~~11~~	~~youth~~	~~medium~~	~~yes~~	~~excellent~~	~~yes~~
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

- rule ***R1, covers 2 of the 14 tuples***

- It can correctly classify both tuples

Coverage(R1)=2/14 = 14.28%

Accuracy(R1) =2/2 = 100%.



Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Coverage = ?

Accuracy = ?

Coverage = 40%,

Accuracy = 50%

Rule: (Status = Single) $\rightarrow$ No

IF-THEN Rules for Classification

- 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

Characteristics of Rule-Based Classifier

- 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

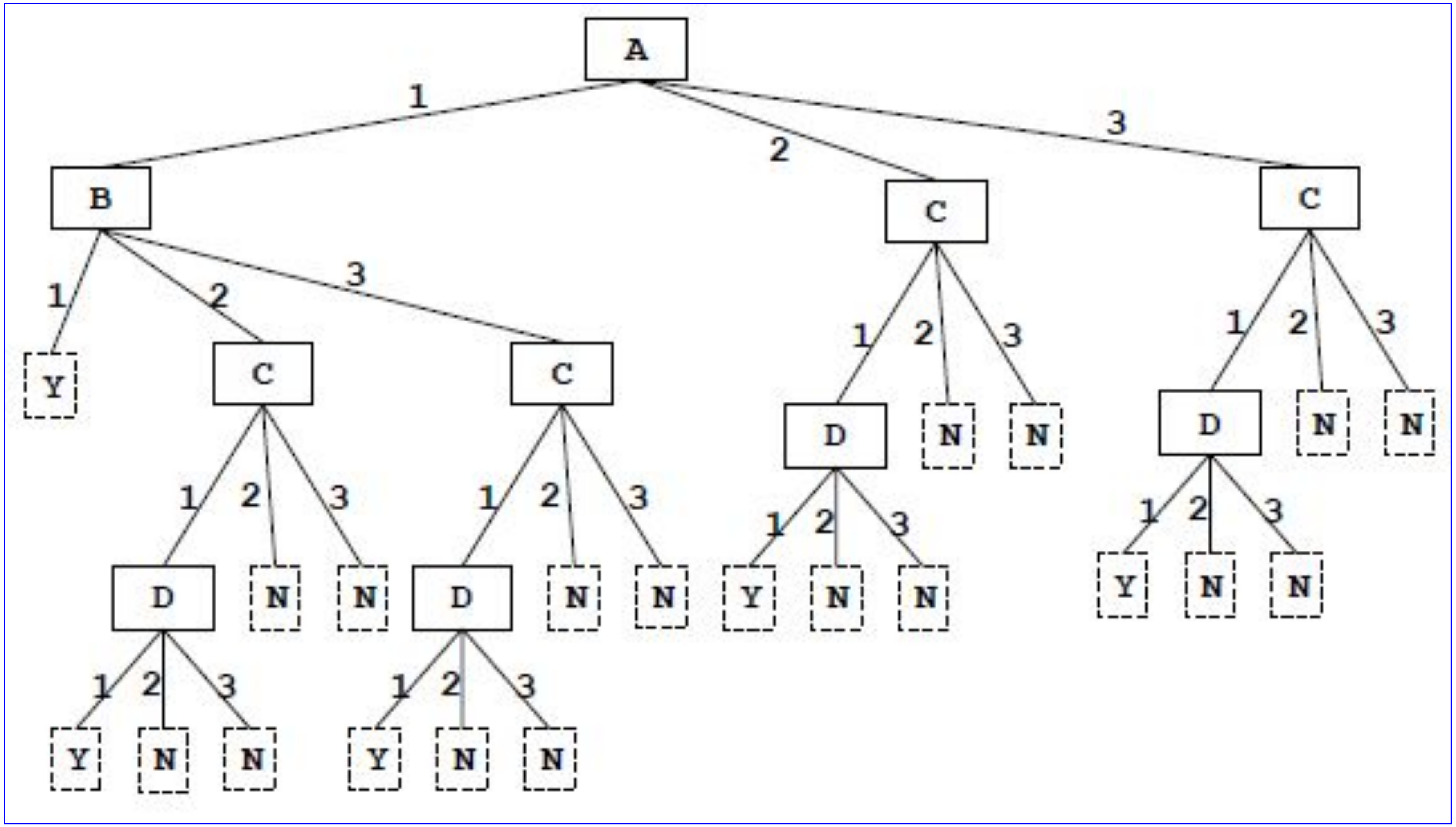
Rule to Tree

- Consider the rule set
  - Attributes A, B, C, and D can have values 1, 2, and 3

$A = 1 \wedge B = 1 \rightarrow \text{Class} = Y$
$C = 1 \wedge D = 1 \rightarrow \text{Class} = Y$
Otherwise, Class = N

- How to represent it as a decision tree?
  - The rules need a common attribute

$A = 1 \wedge B = 1 \rightarrow \text{Class} = Y$
$A = 1 \wedge B = 2 \wedge C = 1 \wedge D = 1 \rightarrow \text{Class} = Y$
$A = 1 \wedge B = 3 \wedge C = 1 \wedge D = 1 \rightarrow \text{Class} = Y$
$A = 2 \wedge C = 1 \wedge D = 1 \rightarrow \text{Class} = Y$
$A = 3 \wedge C = 1 \wedge D = 1 \rightarrow \text{Class} = Y$
Otherwise, Class = N


A
1
2
3
B
C
C
1
2
3
Y
C
C
D
N
N
D
N
N
Y
N
N
Y
N
N
Y
N
N
Y
N
N

Application of Rule-Based Classifier

- A rule *r* **covers** an instance **x** if the attributes of the instance satisfy the condition (LHS) of the rule

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk $\Rightarrow$ Class = *Bird*

The rule R3 covers the grizzly bear $\Rightarrow$ Class = *Mammal*

Building Classification Rules

- Direct Methods
  - Directly learn the rules from the training data

RIPPER, Holte's 1R (OneR)

- Indirect Methods
  - Learn decision tree, then convert to rules
  - Learn neural networks, then extract rules

C4.5rules

A Direct Method: 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

Algorithm: Sequential covering. Learn a set of IF-THEN rules for classification.

Input:

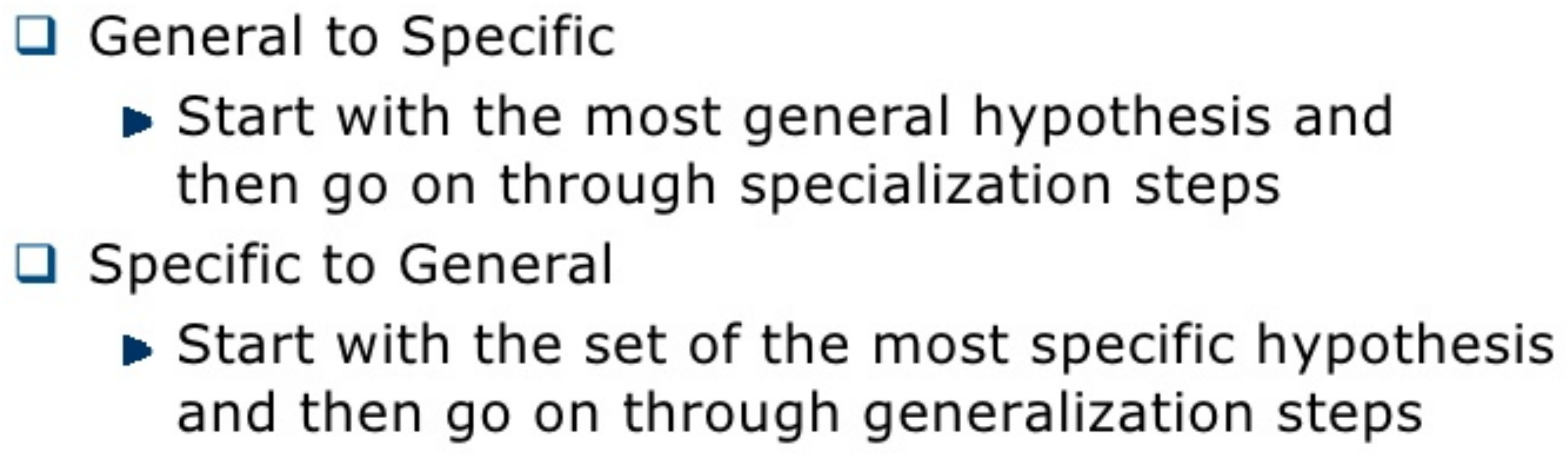
- *D*, a data set of class-labeled tuples;
- *Att_vals*, the set of all attributes and their possible values.

Output: A set of IF-THEN rules.

Method:

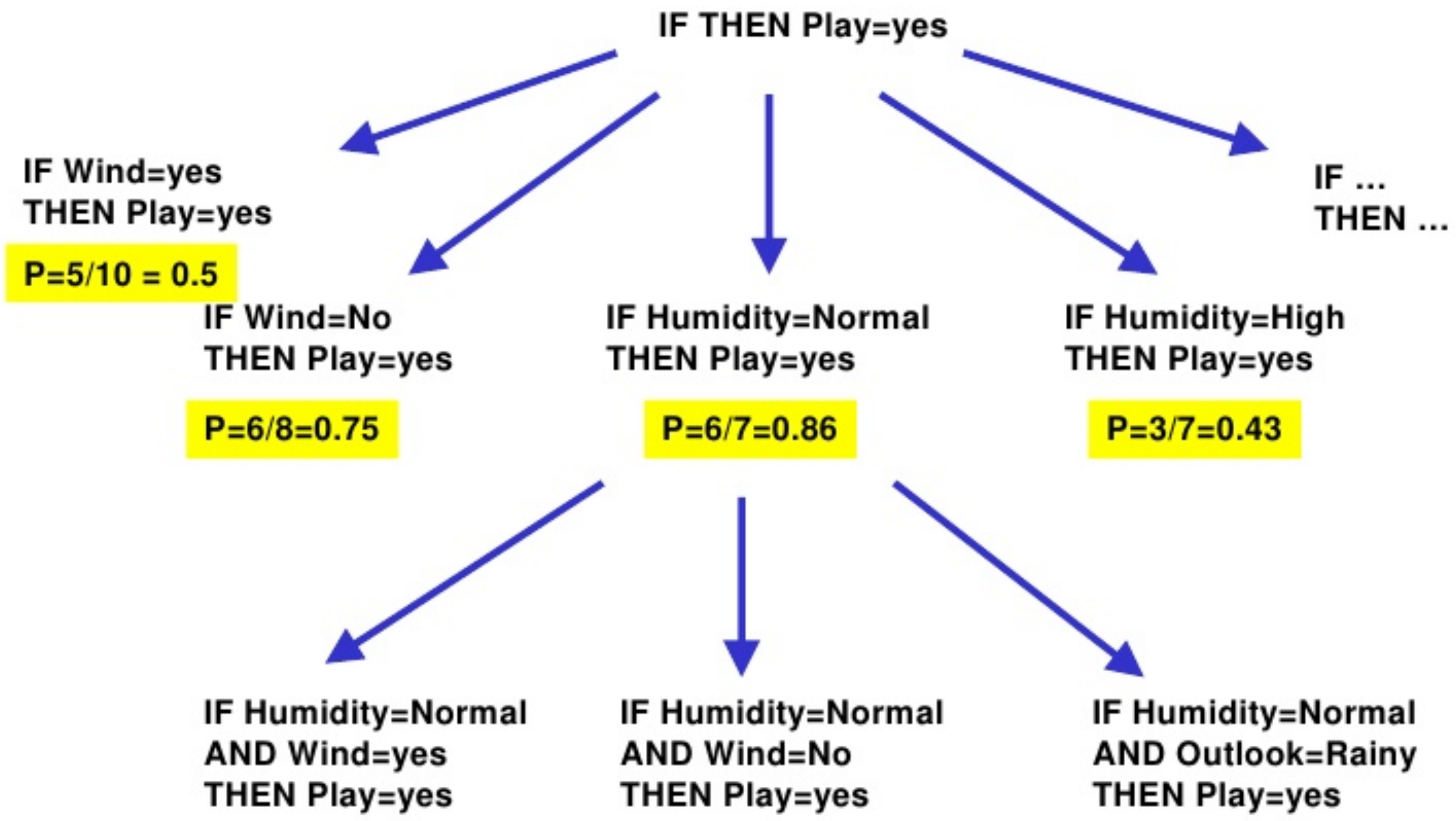
```
(1) Rule_set = {}; // initial set of rules learned is empty
(2) for each class c do
(3)     repeat
(4)         Rule = Learn_One_Rule(D, Att_vals, c);
(5)         remove tuples covered by Rule from D;
(6)         Rule_set = Rule_set + Rule; // add new rule to rule set
(7)     until terminating condition;
(8) endfor
(9) return Rule_Set;
```

How to learn a rule for a class C?

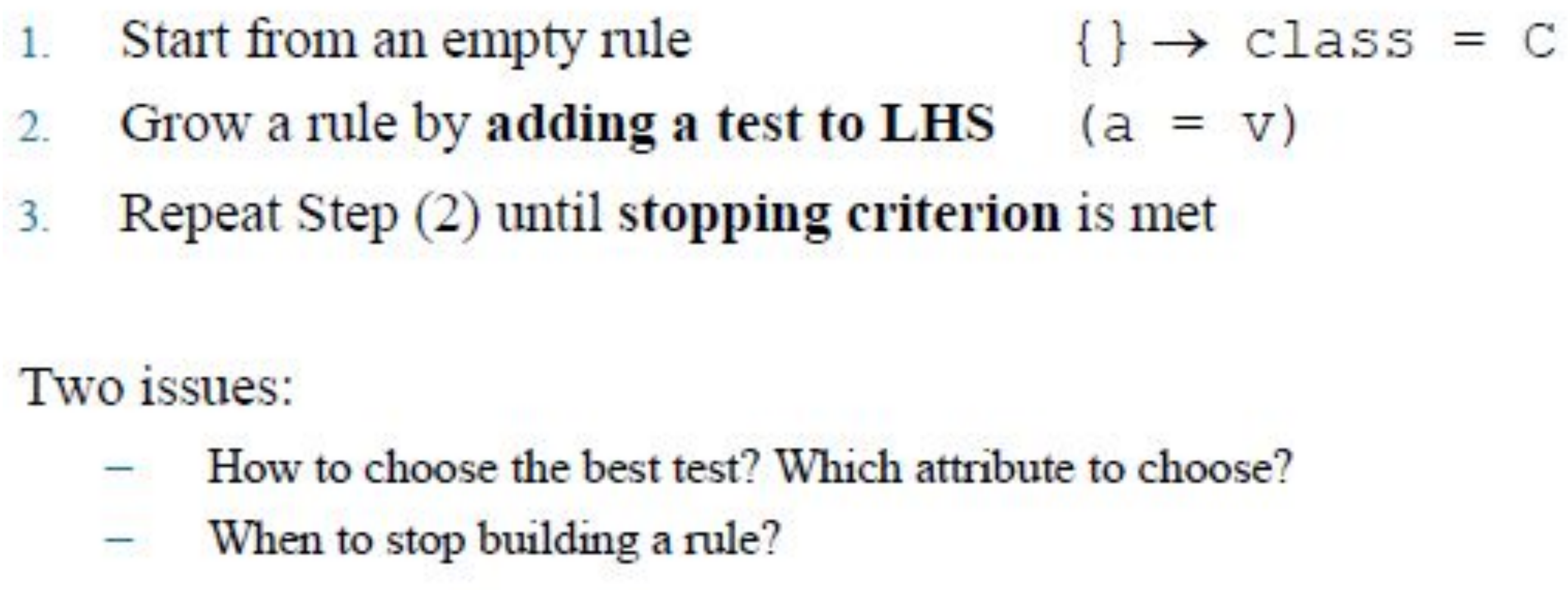


- General to Specific
  - Start with the most general hypothesis and then go on through specialization steps
- Specific to General
  - Start with the set of the most specific hypothesis and then go on through generalization steps

Learning One Rule, General to Specific


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

Learning one rule: another viewpoint

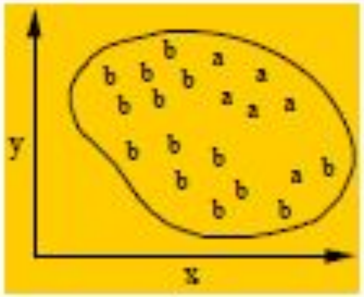


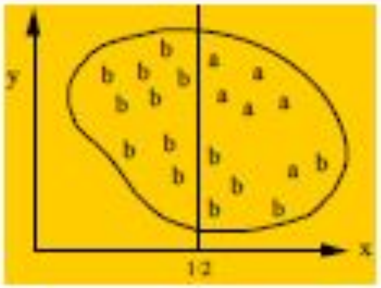
1. Start from an empty rule $\{\} \rightarrow$ `class = C`
2. Grow a rule by **adding a test to LHS** `(a = v)`
3. Repeat Step (2) until **stopping criterion** is met

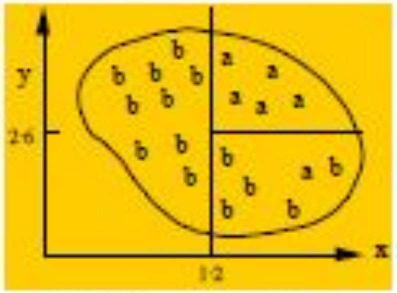
Two issues:

- How to choose the best test? Which attribute to choose?
- When to stop building a rule?

Example: Generating a Rule







```
If {}
   then class = a
```

```
If (x > 1.2)
   then class = a
```

```
If (x > 1.2) and (y > 2.6)
   then class = a
```

- Possible rule set for class "b":

```
If (x ≤ 1.2) then class = b
If (x > 1.2) and (y ≤ 2.6) then class = b
```

- Could add more rules, get "perfect" rule set

Exploring the Hypothesis Space

- 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.

Example: 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

Example: contact lens data

- 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

Modified rule and resulting data

- 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-	Myope	Yes	Reduced	None
presbyopic	Myope	Yes	Normal	Hard
presbyopic	Hypermetrope	Yes	Reduced	None
Pre-	Hypermetrope	Yes	Normal	None
Presbyopic	Myope	Yes	Reduced	None
Presbyopic	Myope	Yes	Normal	Hard
Presbyopic	Hypermetrope	Yes	Reduced	None
Presbyopic	Hypermetrope	Yes	Normal	None

Further refinement

- 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

Modified rule and resulting data

- 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

Further refinement

- 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 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

When to Stop Building a Rule

- When the rule is perfect, i.e. accuracy = 1
- When increase in accuracy gets below a given threshold
- When the training set cannot be split any further

PRISM Algorithm

```
For each class C
  Initialize E to the training 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
```

Learn one rule

Available in *WEKA*

Rule Evaluation in PRISM

$$\text{Accuracy} = \frac{p}{t}$$

t : Number of instances covered by rule

p : Number of instances covered by rule that belong to the positive class

- Produce rules that don't cover *negative* instances, as quickly as possible
- Disadvantage: may produce rules with very small coverage
  - Special cases or noise? (overfitting)

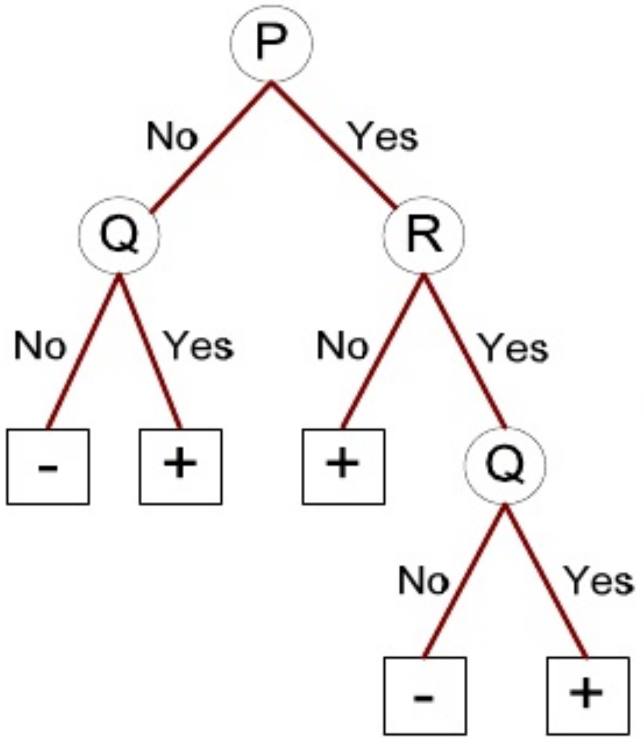
Direct Method: RIPPER

- Learn one rule:
  - Start from empty rule
  - Add conjuncts as long as they improve FOIL's information gain
  - Stop when rule no longer covers negative examples
    - Build rules with accuracy = 1 (if possible)
  - Prune the rule immediately using reduced error pruning
  - Measure for pruning: $W(R) = (p-n)/(p+n)$
    - 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 starts from the last test added to the rule
    - May create rules that cover some negative examples (accuracy < 1)
- A global optimization (pruning) strategy is also applied

Indirect Method: C4.5rules

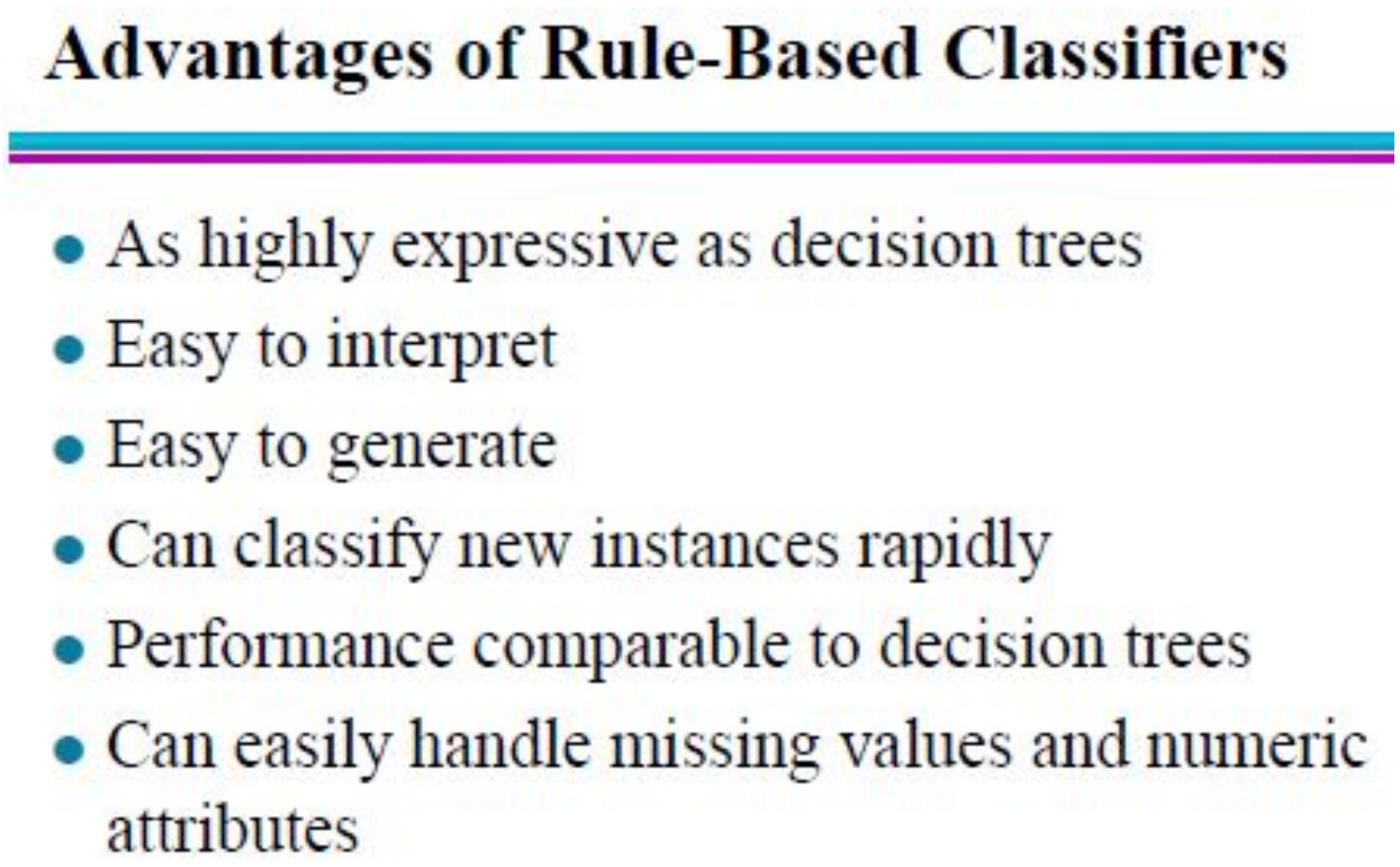
- Extract rules from an unpruned decision tree
- For each rule, r: RHS $\rightarrow$ c, consider pruning the rule
- Use class ordering
  - Each subset is a collection of rules with the same rule consequent (class)
  - Classes described by simpler sets of rules tend to appear first

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) ==> +



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
- Can easily handle missing values and numeric attributes

Available in *WEKA: Prism, Ripper, PART, OneR*