Supervised Learning
Rule-based Classification

The Principle

- The model learned in Rule-Based classification is represented as set of **IF-THEN** rules

  IF condition **THEN** conclusion

- Example

  R1: **IF** age=youth AND student=yes **THEN** buys_computer=yes

- Terminology
  - The "IF" part is known as the **rule antecedent** or **precondition**: consists in one or more attributes
  - The "THEN" part is known as **rule consequent:** contains a class prediction
  - If the condition in a rule antecedent holds true we say the condition is **satisfied** or the rule **covers** the tuple

Example

RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

R1: **IF** age=youth AND student=yes **THEN** buys_computer=yes

The condition is satisfied = The rule covers the tuple

How to Assess The Rules

- A rule **R** can be assessed by
  - Coverage
  - Accuracy
- Methodology
  - Class labeled dataset **D** (a set of tuples)

Consider

- $\mathbf{n_{covers}}$: the number of tuples covered by R
- $\mathbf{n_{correct}}$: the number of tuples correctly classified by R
- **|D|**: the total number of tuples in D

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

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

Example

RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

R1: **IF** age=youth AND student=yes **THEN** buys_computer=yes

n_{covers}= ?
$n_{correct}$=?

Example

RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

R1: **IF** age=youth AND student=yes **THEN** buys_computer=yes

$n_{covers} = 3$

$n_{correct} = ?$

Example

RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

R1: **IF** age=youth AND student=yes **THEN** buys_computer=yes

$n_{covers} = 3$
$n_{correct} = 2$

$$\text{coverage}(R1) = \frac{n_{\text{covers}}}{|D|} = \frac{3}{7}$$

$$\text{accuracy}(R1) = \frac{n_{correct}}{n_{\text{covers}}} = \frac{2}{3}$$

How to use Rules for Classification

- Predict the class label for tuple X
  - If a rule **R** is satisfied by X, the rule is said to be **triggered**
  - If a rule **R** is the only one satisfied by X, the rule **fires** by returning the class prediction of X
- **Important**
  - Triggering ≠ Firing
  - More than one rule can be satisfied
- **Problems**
  - What if no rule is satisfied by X?
  - **Solution**: use a default rule that fires, for example, the most frequent class
  - If more than one rule are triggered, what if each rule specifies a different class?

Conflicting Rules

R1: **IF** age=youth AND student=yes **THEN** buys_computer=yes

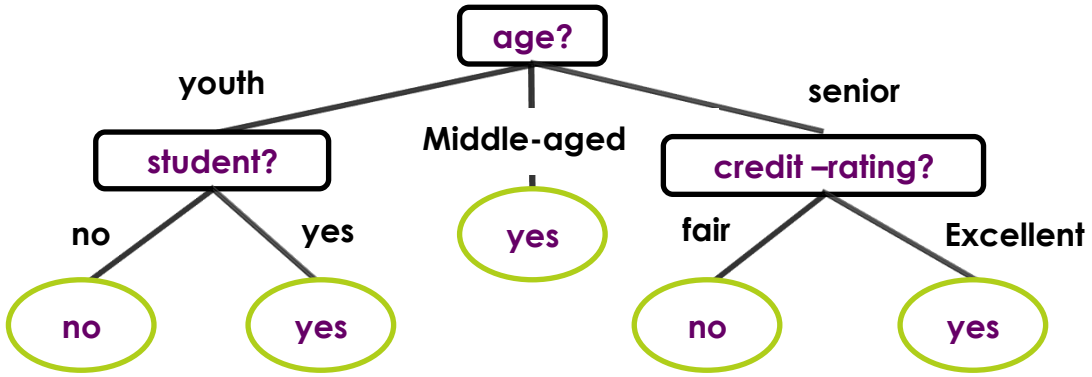
R2: **IF** income=low **THEN** buys_computer=no

- Need a conflict resolution strategy
  - **Size ordering approach**
    - Give priority to the rule having the toughest requirement
    - **Toughness** is measured by the rule antecedent size
    - The triggering rule with the most attribute sets is fired
  - **Rule ordering approach**
    - Prioritize the rules beforehand
    - **Class-based** ordering
    - **Rule-based** ordering

Rule Extraction From a Decision Tree

- One rule is created for each path from the root to a leave node
- Each splitting criterion along a given path is logically ANDed to form the rule antecedent (IF part)
- The leaf node holds the class prediction (the rule consequent)

R1: **IF** age=youth **AND** student=no **THEN** buys_cpmputer=no
R2: **IF** age=youth **AND** student=yes **THEN** buys_cpmputer=yes
R3: **IF** age=middle-aged **THEN** buys_cpmputer=yes
R4: **IF** age=senior **AND** credit_rating=excellent **THEN** buys_cpmputer=yes
R5: **IF** age=senior **AND** credit_rating=fair **THEN** buys_cpmputer=no



Characteristics of Decision Tree Rules

- Decision tree rules are mutually **exclusive** and **exhaustive**
- **Exclusive**
  - No rule conflict, no two rules triggered for the same tuple
  - One rule per leaf and any tuple is mapped to only one leaf
- **Exhaustive**
  - One rule for each attribute-value combination
  - The set of rules does not require a default rule

Note: The order of rules does not matter when extracted from a decision tree

- Pruning Rules
  - Any rule that does not improve accuracy can be pruned
  - Pruning may generates non Mutually exclusive and non exhaustive rules: C4.5 uses class-based ordering

Sequential Covering Algorithm

- IF-THEN rules are **directly** extracted from training data
- Rules are learned sequentially (one at a time) [**Note**: In decision trees rules are learned simultaneously
- Each rule for a given class ideally covers many tuples of that class and hopefully no tuples from other classes
- When a rule is learned, the tuples covered by the rule are removed (need of accurate rules but not necessarily high coverage)
- The process repeats on the remaining tuples until a stopping condition:
  - No tuples left
  - The quality measure of a rule is below a threshold

How Are Rules learned?

- In a **general-to-specific** manner
- **Example**
  - In loan-application data, costumers have (age, income, education level, residence, credit-rating, and term of the loan)
  - Two classes: loan_decision=accept and loan_decision=reject
- Start with a general rule for class accept:

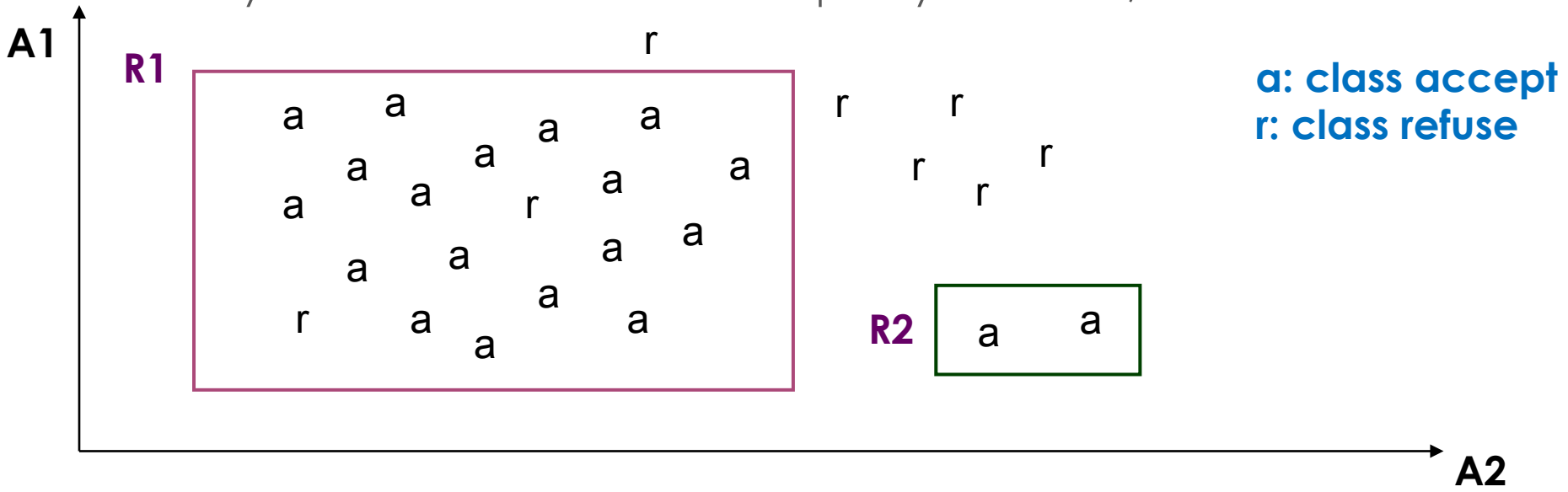
IF **THEN** loan_decision=accept

  - Consider each possible attribute test that may be added to the rule
  - Adopt a greedy depth-first strategy choosing the rule with high quality (use beam search where the k best attributes are maintained)
  - Repeat the process till the rule reached an acceptable quality level

IF income=high AND credit_rating=execellent
THEN loan_decision=accept

Rule Quality Measure

- Accuracy seems to be natural as a quality measure, but



  - R1: correctly classifies 18 tuples out of 20 (accuracy=90%)
  - R2: correctly classifies 2 tuples out of 2 (accuracy=100%)
- Accuracy alone is not enough
- Coverage alone is not enough (cover many tuples of ≠ classes)
- Use **Entropy**

Rule Quality Measure

- Using **Entropy** (Information Gain)

R: IF condition THEN class=c

- If logically ANDing a given attribute test to **condition** we obtain **condition'**

R': IF condition' THEN class=c

  - Test the potential rule R' using entropy
  - Compute the **entropy** based on probabilities p_i, where p_i is the probability of a class C_i in **D**
  - **D** is the set of tuples covered by **R'**
  - Entropy prefers conditions that cover a large number of tuples of a single class and few tuples of other classes

Rule Quality Measures

- Using **FOIL_Gain (First Order Inductive Learner- Gain)**

R: IF condition THEN class=c

- If logically ANDing a given attribute test to **condition** we obtain **condition'**

R': IF condition' THEN class=c

- The FOIL Gain is computed by

$$FOIL\_Gain = pos' \times \left(\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg} \right)$$

  - **pos**,**pos'**: the number of positive tuples covered by R and R'
  - **neg**, **neg'**: the number of negative tuples covered by R and R'

- It favors rules that have high accuracy and cover many positive tuples

Summary

- Rule-based classification builds a model that is a **set of rules**
- Rules can be **extracted** from a **decision tree** or directly from **training data**
- Rule **quality measures** are important to **assess** the rules and to define orders for **conflict resolution**