Classification: Basic Classifiers

Lecturer: Dr. Nguyen Ngoc Thao
Department of Computer Science, FIT, HCMUS

Outline

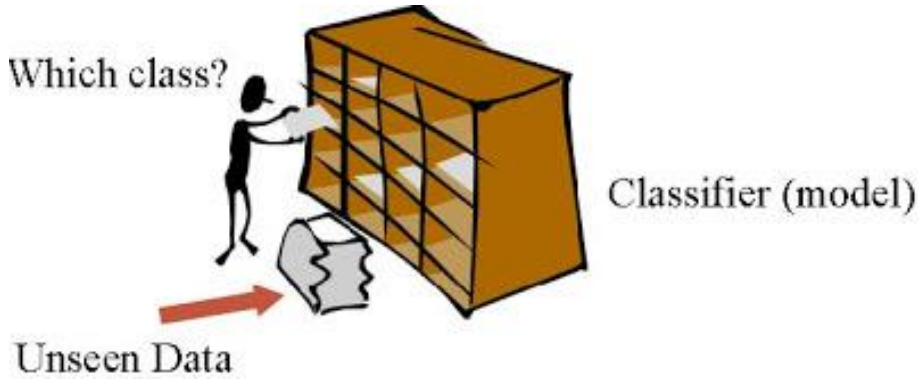
- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification
- Rule-Based Classification
- Lazy Learners (or Learning from Your Neighbors)
- Model Evaluation
- Summary

Outline

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification
- Rule-Based Classification
- Lazy Learners (or Learning from Your Neighbors)
- Model Evaluation
- Summary

Classification: Definition

- Given a collection of records (training set)
  - Each record contains a set of attributes, one of the attributes is the class
- Classification finds a model for class attribute as a function of the values of other attributes
- Goal: previously unseen records (test set) should be assigned a class as accurately as possible.



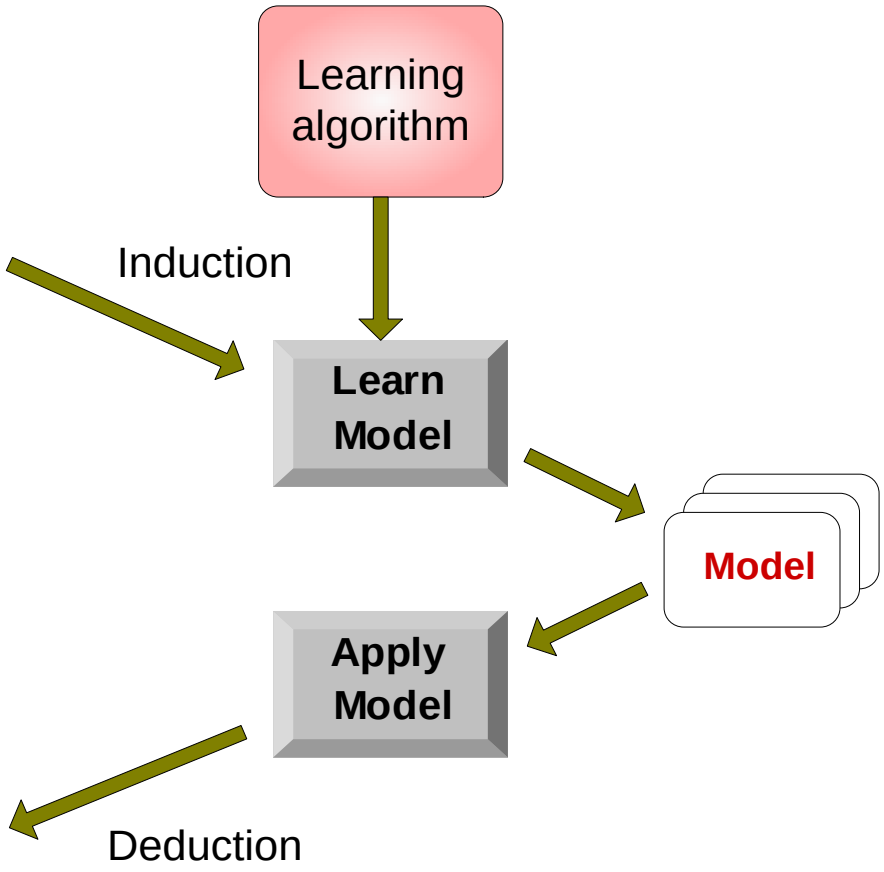
Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

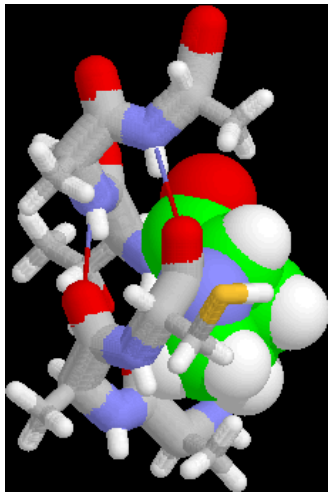


Examples of Classification Task

- Predicting tumor cells as benign or malignant

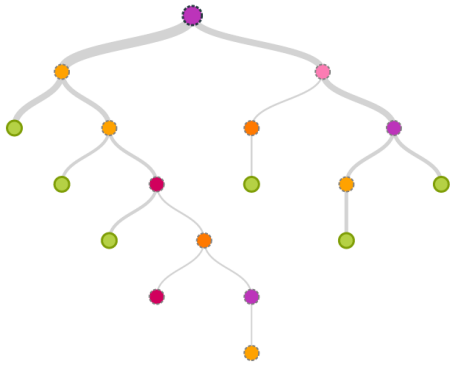
- Classifying credit card transactions as legitimate or fraudulent

- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil



- Categorizing news stories as finance, weather, entertainment, sports, etc

Classification Techniques

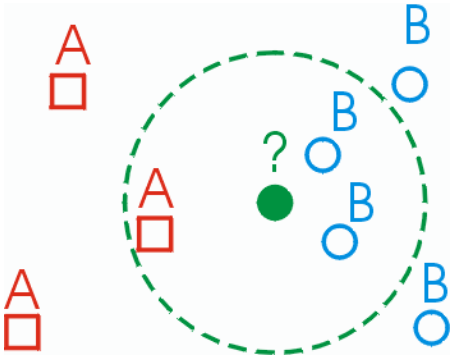


Decision Tree Induction
(ID3, C4.5,…)

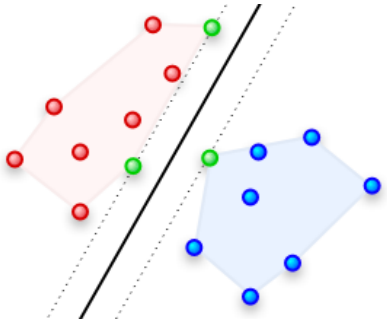
IF… THEN…

Rule-based Methods
(ILA, Learn-one-rule,…)

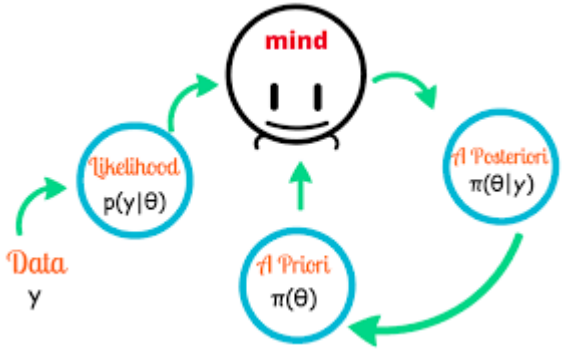
Neural Network



Lazy Learners
(k-NN, case-based,…)



Support Vector Machines



Neural Network

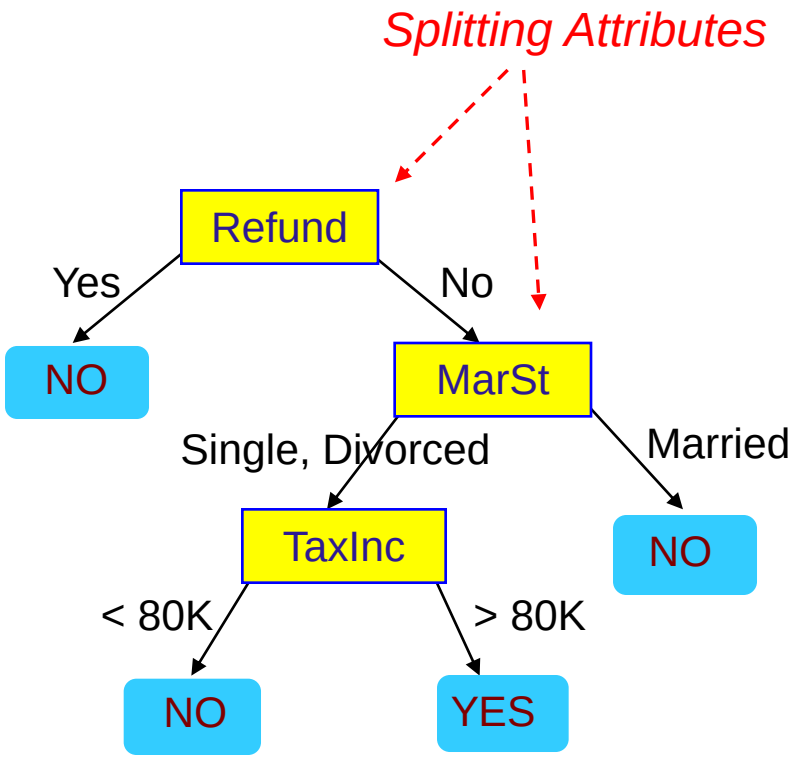
Outline

- Classification: Basic Concepts
- **Decision Tree Induction**
- Bayes Classification
- Lazy Learners (or Learning from Your Neighbors)
- Rule-Based Classification
- Model Evaluation
- Summary

Example of a Decision Tree

Tid	Refund	Marital Status	Taxable Income	Cheat
	categorical	categorical	continuous	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

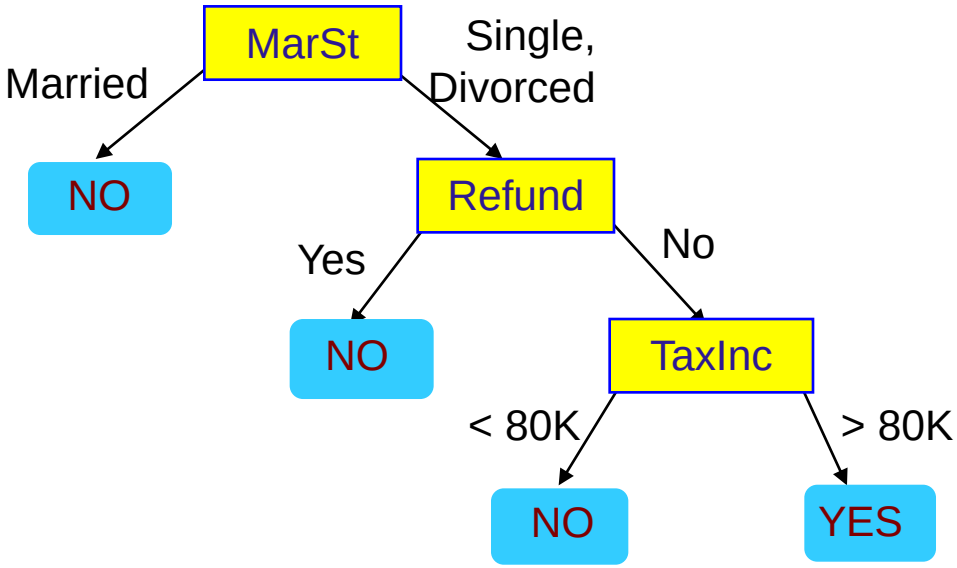
Training Data



Model: Decision Tree

Example of a Decision Tree

Tid	Refund	Marital Status	Taxable Income	Cheat
	categorical	*categorical*	*continuous*	*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



There could be more than one tree that fits the same data!

According to Occam's Razor, the more compact tree is the better one.

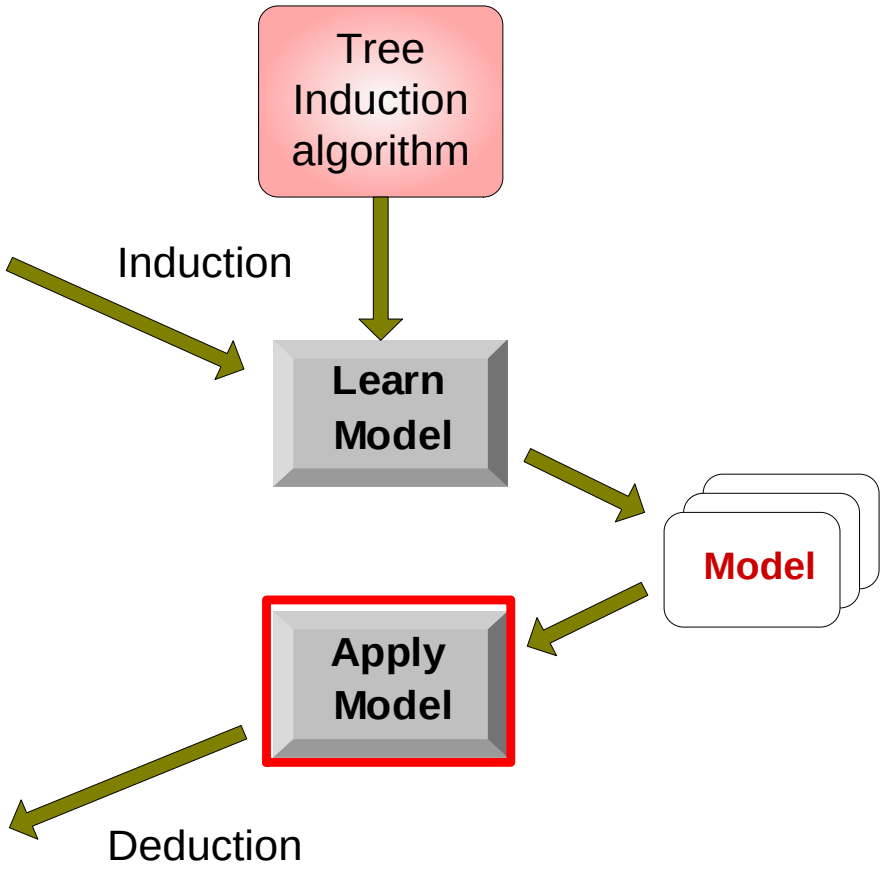
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

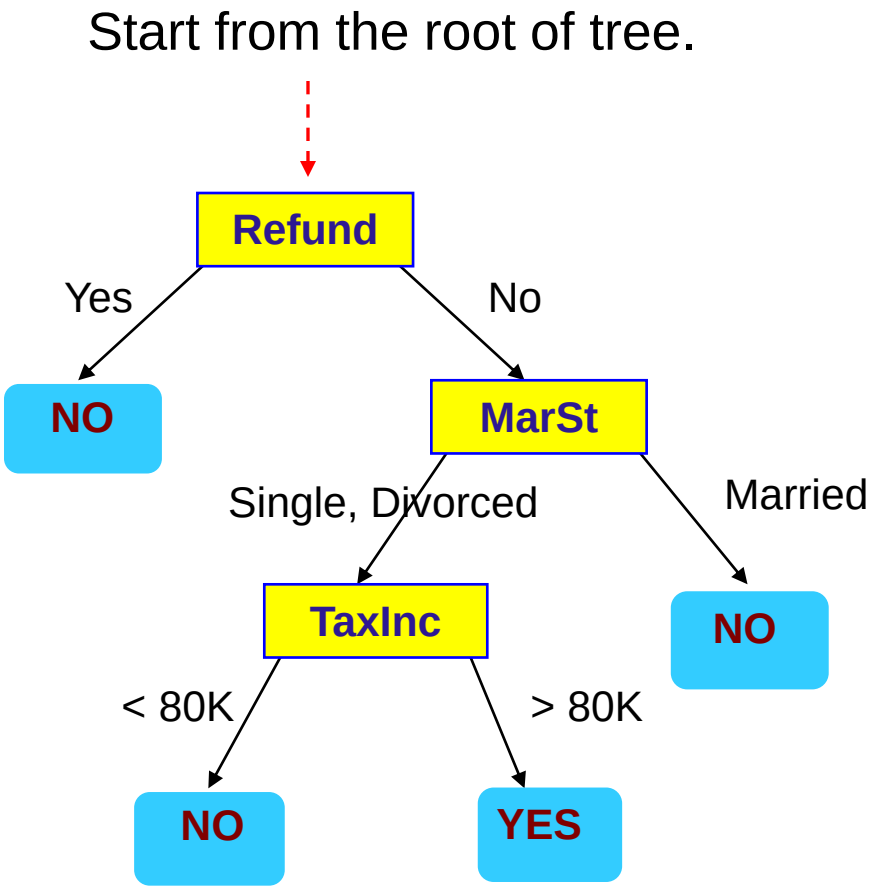
Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



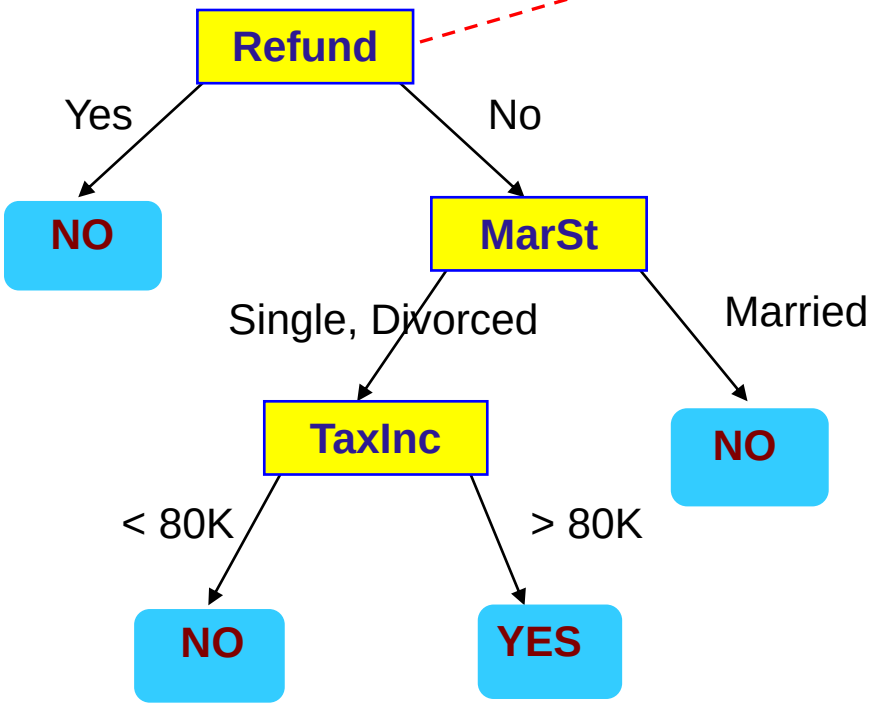
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

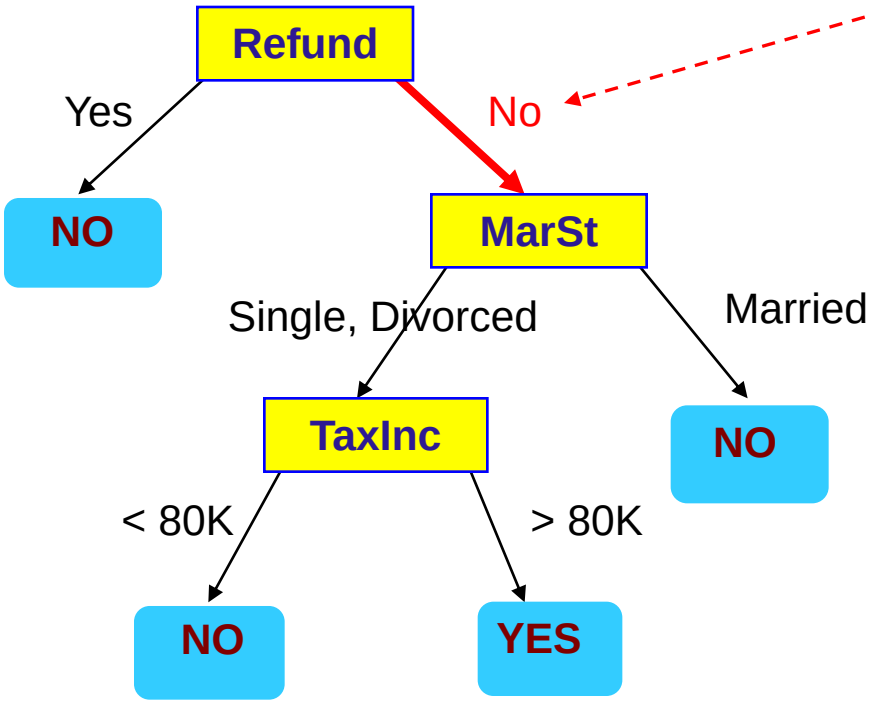
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

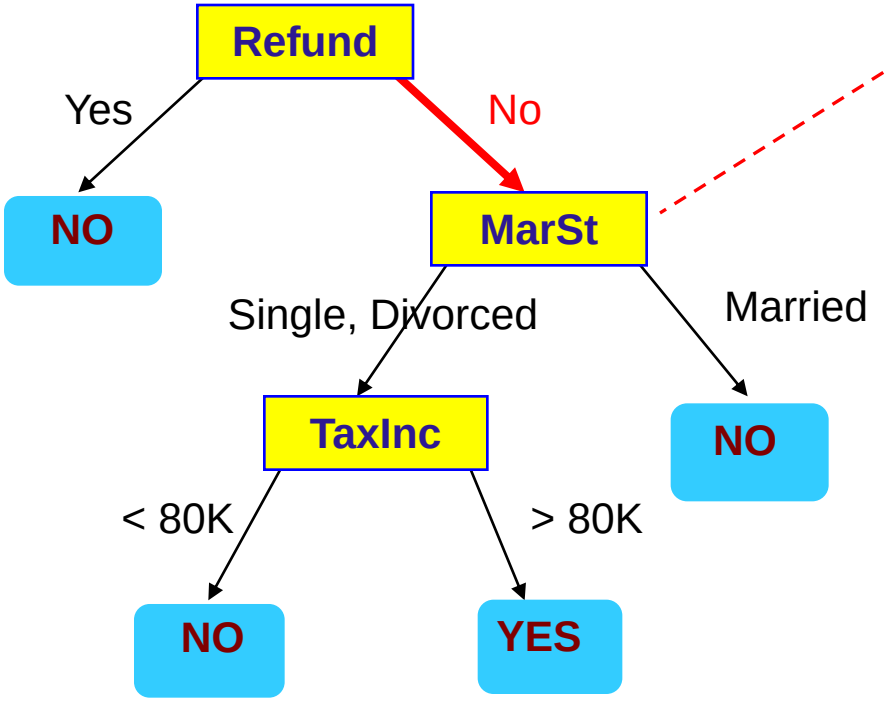
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

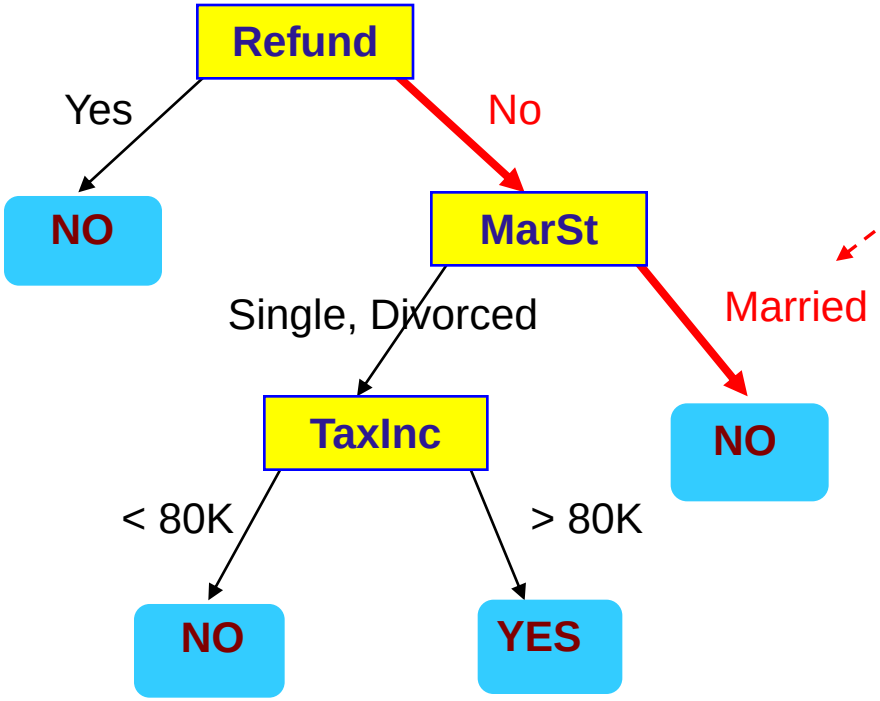
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

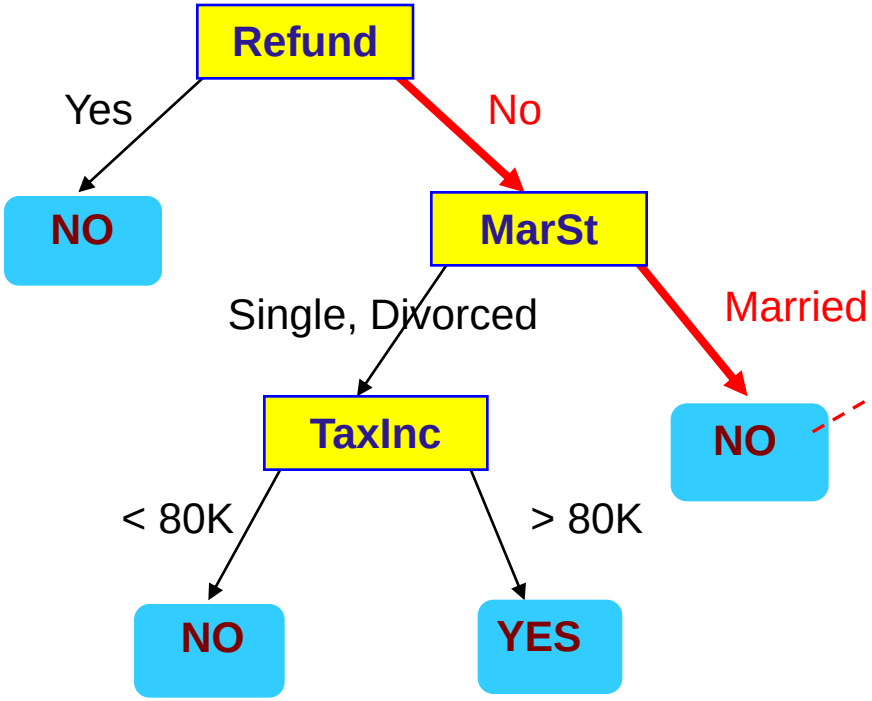
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to “No”

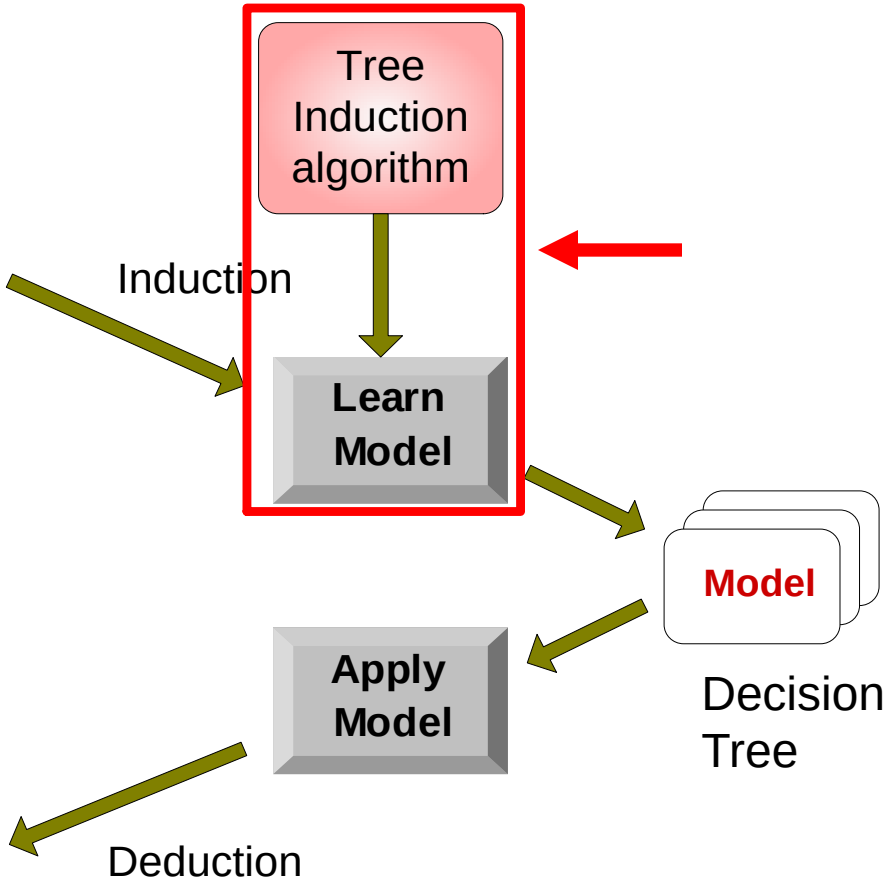
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is built in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical
    - Continuous attributes need to be discretized in advance
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

Algorithm for Decision Tree Induction

- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

- There are several decision tree algorithms, such as ID3 (1986), C4.5 (1993), CART (1984), SLIQ (1996), SPRINT (1996)

Key Issues in Tree Induction

- Greedy strategy: Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

ID3 Decision Tree

- ID3 (Iterative Dichotomiser 3) is invented by Ross Quinlan used to generate a decision tree from a dataset
- It is the precursor to the C4.5 algorithm
- It is typically used in the machine learning and natural language processing domains

ID3 Decision Tree: Algorithm

- ID3 begins with the original set S as the root node
- On each iteration, the algorithm
  - Iterates through every unused attribute of the set S and calculates the entropy $H(S)$ of that attribute
  - Selects the attribute which has the smallest entropy value
  - The set S is then split by the selected attribute to produce subsets of the data
    - E.g. age is less than 50, age is between 50 and 100, age is greater than 100
  - Recursion is performed on each subset, considering only attributes never selected before

ID3 Decision Tree: Algorithm

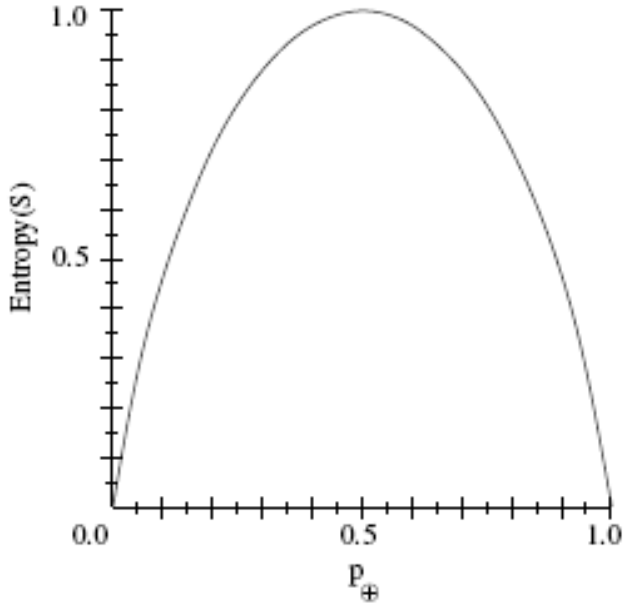
- Non-terminal nodes represent the selected attributes on which the data was split, and terminal nodes represent the class label of the final subset of this branch
- Recursion on a subset may stop in one of these cases:
  - Every element in the subset belongs to the same class (+ or -),
  - There are no more attributes to be selected, but the examples still do not belong to the same class (some are + and some are -),
    - Label with the most common class of the examples in the subset
  - There are no examples in the subset since no example in the parent set was found to be matching a specific value of the selected attribute
    - Label with the most common class of the examples in the parent set

Brief Review of Entropy

- A measure of uncertainty associated with a random variable
- For a discrete random variable A taking v distinct values $\{a_1, a_2, \ldots, a_v\}$

$$H(A) = -\sum_{i=1}^{v} p_i \log(p_i)$$

  - Where $p_i = P(A = a_i)$ and log is usually the natural logarithm (ln)
- Interpretation: the higher the entropy is, the more uncertainty it is



ID3 with Entropy Measure

- The average entropy (AE) of an attribute is computed from the entropy values of child nodes

$$Average\ Entropy\ (A) = \sum_{v \in Value(A)} p_v H_{A=v}$$

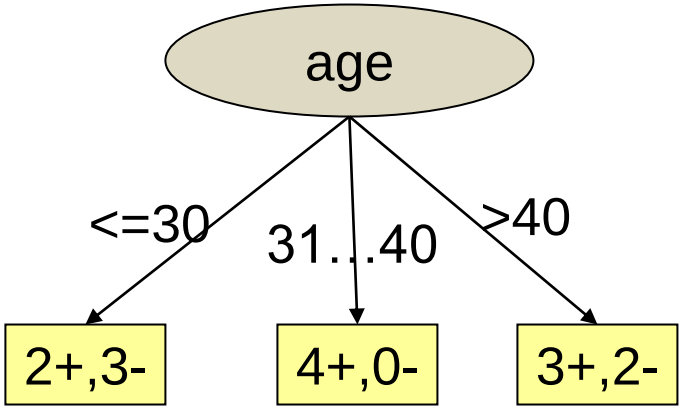
- ID3 selects the best attribute that minimize the AE value

ID3 Decision Tree: An Example

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	**no**
<=30	high	no	excellent	**no**
31…40	high	no	fair	**yes**
>40	medium	no	fair	**yes**
>40	low	yes	fair	**yes**
>40	low	yes	excellent	**no**
31…40	low	yes	excellent	**yes**
<=30	medium	no	fair	**no**
<=30	low	yes	fair	**yes**
>40	medium	yes	fair	**yes**
<=30	medium	yes	excellent	**yes**
31…40	medium	no	excellent	**yes**
31…40	high	yes	fair	**yes**
>40	medium	no	excellent	**no**

ID3 Decision Tree: An Example

- Select attribute for the tree root

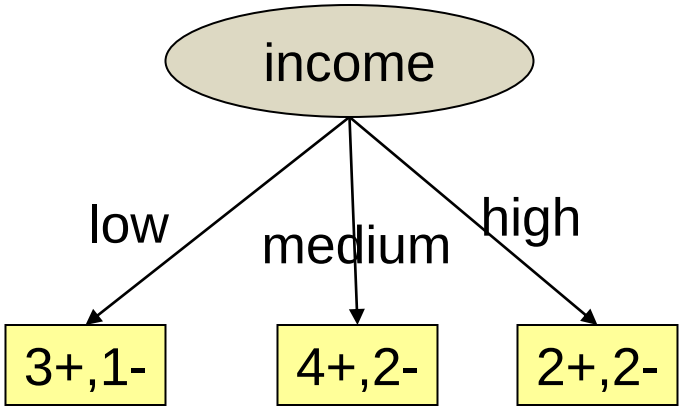


$H_{<=30} = -2/5*\log_2 2/5 - 3/5*\log_2 3/5 = 0.971$

$H_{31…40} = -4/4*\log_2 4/4 - 0/4*\log_2 0/4 = 0$

$H_{>40} = -3/5*\log_2 3/5 - 2/5*\log_2 2/5 = 0.971$

$AE = 5/14*0.971 + 4/14*0 + 5/14*0.971 = 0.694$



$H_{low} = -3/4*\log_2 3/4 - 1/4*\log_2 1/4 = 0.811$

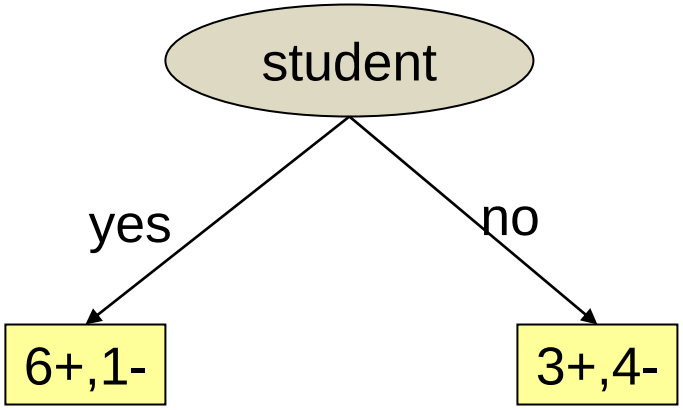
$H_{medium} = -4/6*\log_2 4/6 - 2/6*\log_2 2/6 = 0.918$

$H_{high} = -2/4*\log_2 2/4 - 2/4*\log_2 2/4 = 1$

$AE = 4/14*0.811 + 6/14*0.918 + 4/14*1 = 0.911$

ID3 Decision Tree: An Example

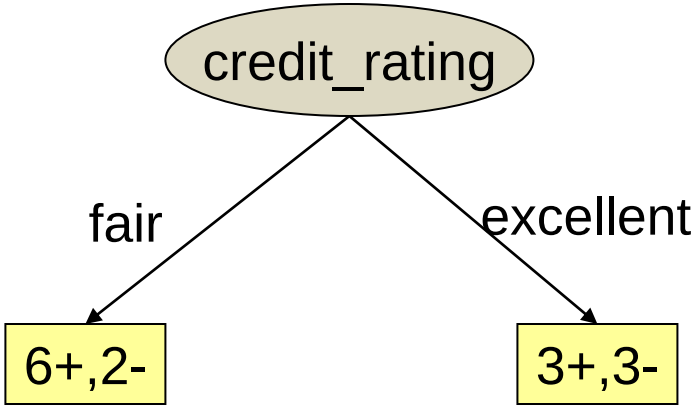
- Select attribute for the tree root



H_{yes}= – 6/7*$\log_2$6/7 – 1/7*$\log_2$1/7 = 0.592

H_{no} = – 3/7*$\log_2$3/7 – 4/7*$\log_2$4/7 = 0.985

AE = 7/14*0.592 + 7/14*0.985 = 0.789



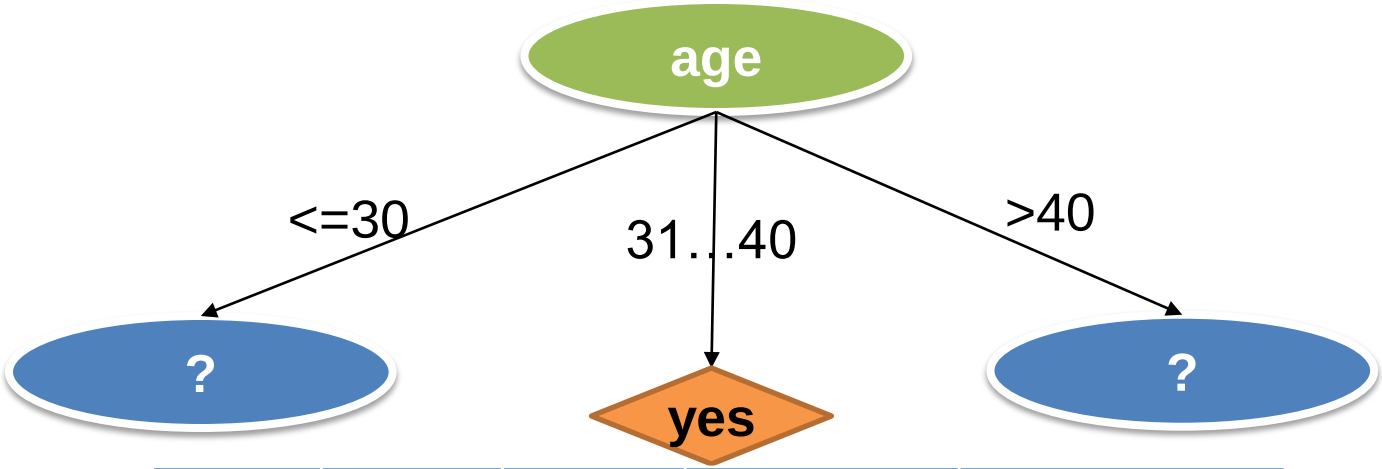
H_{yes}= – 6/8*$\log_2$6/8 – 2/8*$\log_2$2/8 = 0.811

$H_{excellent}$ = 1

AE = 8/14*0.811 + 6/14*1 = 0.892

ID3 Decision Tree: An Example

- Select attribute for the tree root

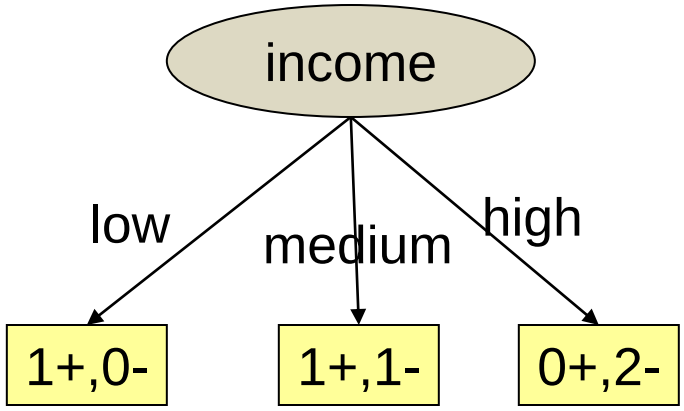


age	income	student	credit_rating	buys_computer
31…40	high	no	fair	**yes**
31…40	low	yes	excellent	**yes**
31…40	medium	no	excellent	**yes**
31…40	high	yes	fair	**yes**

ID3 Decision Tree: An Example

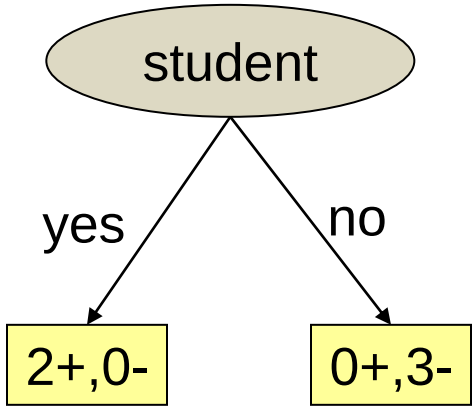
- Select attribute for the branch age = <=30

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	**no**
<=30	high	no	excellent	**no**
<=30	medium	no	fair	**no**
<=30	low	yes	fair	**yes**
<=30	medium	yes	excellent	**yes**



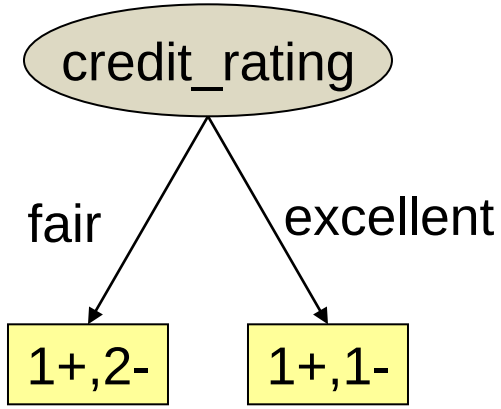
$H_{low} = 0$ $H_{medium} = 1$ $H_{high} = 0$

AE = 1/5*0 + 2/5*1 + 2/5*0 = 0.4



$H_{yes} = 0$ $H_{no} = 0$

AE = 0



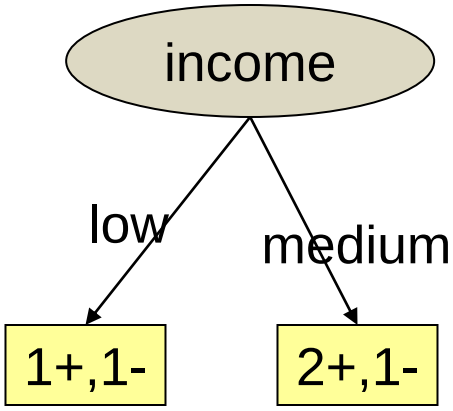
$H_{fair} = 0.918$ $H_{excellent} = 1$

AE = 0.951

ID3 Decision Tree: An Example

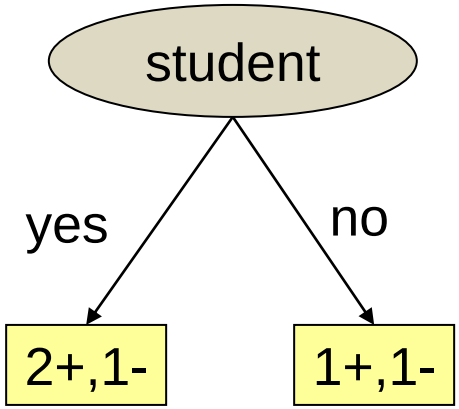
- Select attribute for the branch age = >40

age	income	student	credit_rating	buys_computer
>40	medium	no	fair	**yes**
>40	low	yes	fair	**yes**
>40	low	yes	excellent	**no**
>40	medium	yes	fair	**yes**
>40	medium	no	excellent	**no**



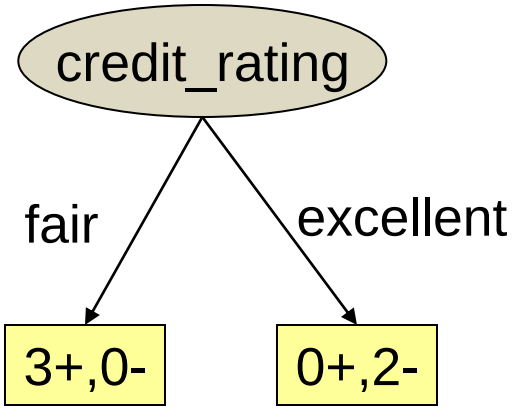
$H_{low} = 1$ $H_{medium} = 0.918$

AE = 0.951



$H_{yes} = 0.918$ $H_{no} = 1$

AE = 0.951

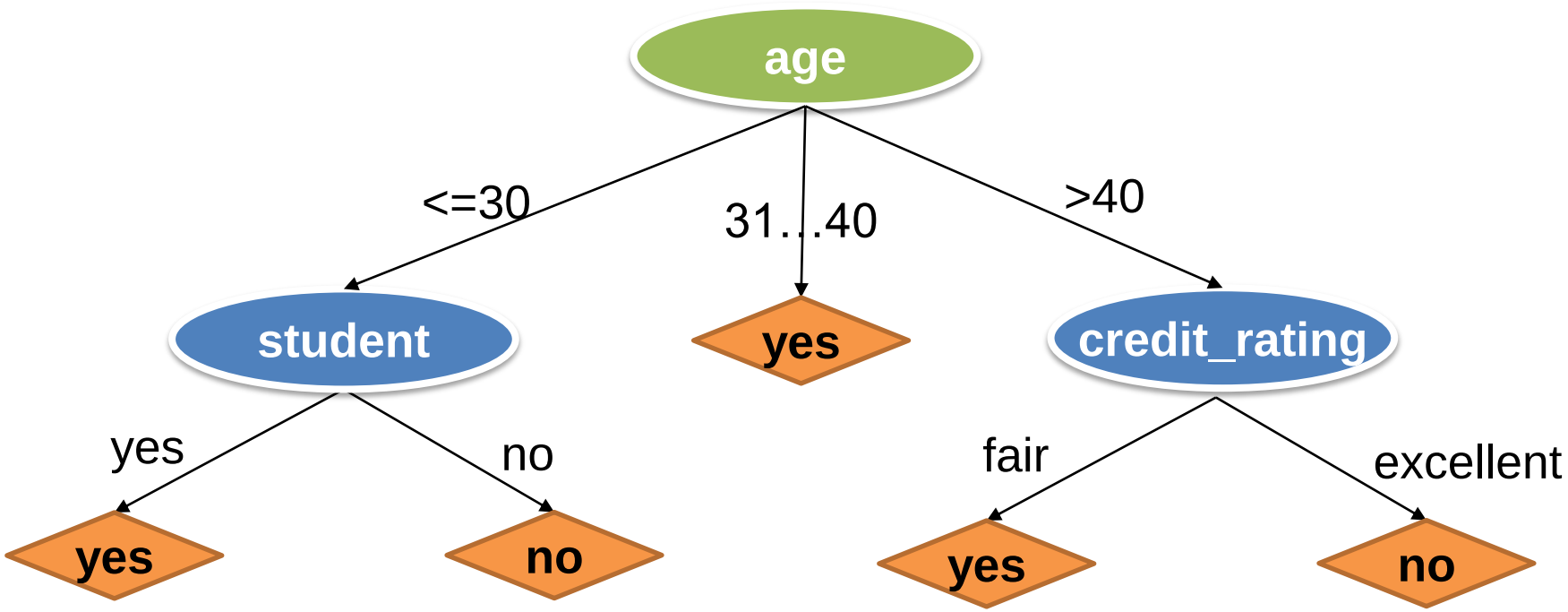


$H_{fair} = 0$ $H_{excellent} = 0$

AE = 0

ID3 Decision Tree: An Example

- The final ID3 tree



Information Gain (ID3/C4.5)

- Let an attribute $A = \{a_1, a_2, \dots, a_v\}$, which divides the training set D into partitions $\{D_1, D_2, \dots, D_v\}$
- Information gain by branching on attribute A

$$IG(A) = Info(D) - Info_A(D)$$

  - Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{v} p_i \log(p_i)$$

  - Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} Info(D_j)$$

- Select the attribute with the highest information gain

Information Gain Ratio (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$IGRatio(A) = IG(A)/SplitInfo_A\ (D)$$

  - $SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times log(\frac{|D_j|}{|D|})$

- Select the attribute with the maximum gain ratio

Gini Index (CART, IBM IntelligentMiner)

- Let an attribute $A = \{a_1, a_2, \dots, a_v\}$, which divides the training set D into partitions $\{D_1, D_2, \dots, D_v\}$
- Gini index of D :

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

  - where p_j is the relative frequency of class j in D
- Gini index of D by branching on attribute A

$$gini_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{D} gini(D_j)$$

- Select the attribute with the minimum gini index

Exercise 1

- Build the ID3 decision tree using Information gain measure

No.	snow	weather	season	physical condition	go skiing
1	sticky	foggy	low	rested	no
2	fresh	sunny	low	injured	no
3	fresh	sunny	low	rested	yes
4	fresh	sunny	high	rested	yes
5	fresh	sunny	mid	rested	yes
6	frosted	windy	high	tired	no
7	sticky	sunny	low	rested	yes
8	frosted	foggy	mid	rested	no
9	fresh	windy	low	rested	yes
10	fresh	windy	low	rested	yes
11	fresh	foggy	low	rested	yes
12	fresh	foggy	low	rested	yes
13	sticky	sunny	mid	rested	yes
14	frosted	foggy	low	injured	no

Exercise 2

- Build the ID3 decision tree using Gini Index measure

No.	Refund	Marital Status	Taxable Income	Cheat
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	No

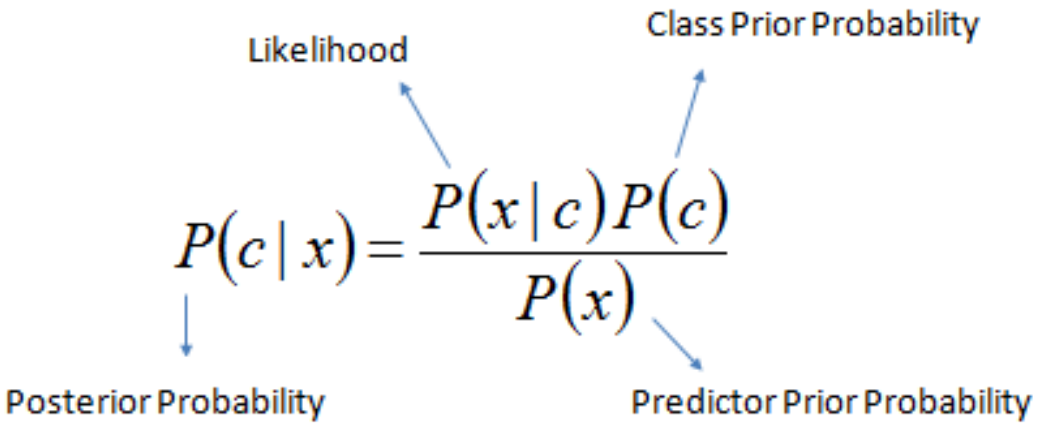
Taxable Income attribute values will be splitted into <80K and >=80K

Outline

- Classification: Basic Concepts
- Decision Tree Induction
- **Bayes Classification**
- Rule-Based Classification
- Lazy Learners (or Learning from Your Neighbors)
- Model Evaluation and Selection
- Summary

Bayesian Classification: Why?

- A statistical classifier performs probabilistic prediction, i.e., predicts class membership probabilities
- Foundation: Based on Bayes' Theorem



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Bayesian Classification: Why?

- Performance: A simple Bayesian classifier, such as naïve Bayesian classifier, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayes' Theorem: Basics

- Total probability Theorem: $P(B) = \sum_{i=1}^{M} P(B|A_i)P(A_i)$

- Bayes' Theorem: $P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{P(\mathbf{X})}$
  - Let $\mathbf{X}$ be a data sample ("evidence"): class label is unknown
  - Let H be a hypothesis that $\mathbf{X}$ belongs to class C
  - Classification is to determine the posteriori probability $P(H \mid \mathbf{X})$, i.e. the probability that the hypothesis holds given the observed data sample $\mathbf{X}$

Bayes' Theorem: Basics

- Total probability Theorem: $P(B) = \sum_{i=1}^{m} P(B|A_i)P(A_i)$

- Bayes' Theorem: $P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{P(\mathbf{X})}$
  - $P(H)$ (prior probability): the initial probability
    - E.g., $\mathbf{X}$ will buy computer, regardless of age, income, …
  - $P(\mathbf{X})$: probability that sample data is observed
  - $P(\mathbf{X} \mid H)$ (likelihood): the probability of observing the sample $\mathbf{X}$, given that the hypothesis holds
    - E.g., Given that $\mathbf{X}$ will buy computer, the prob. that $\mathbf{X}$ is 31..40, medium income

Prediction Based on Bayes' Theorem

- Given training data $\mathbf{X}$, *posteriori probability of a hypothesis* H, $P(H \mid \mathbf{X})$, follows the Bayes' theorem

$$P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{P(\mathbf{X})}$$

- Informally, this can be viewed as

  posteriori = likelihood $\times$ prior / evidence

- Predicts $\mathbf{X}$ belongs to C_i iff the probability $P(C_i \mid \mathbf{X})$ is the highest among all the $P(C_k \mid \mathbf{X})$ for all the k classes
- Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

Classification: Derive the Maximum Posteriori

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector $\mathbf{X} = (x_1, x_2, \ldots, xn)$
- Suppose there are m classes $C_1, C_2, \ldots, Cm$
- Classification is to derive the maximum posteriori $P(C_i | \mathbf{X})$
- This can be derived from Bayes' theorem

$$P(C_i \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, only $P(\mathbf{X} \mid C_i)P(C_i)$ needs to be maximized

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X} \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) = P(x_1 \mid C_i) \times P(x_2 \mid C_i) \times \cdots \times P(x_n \mid C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution

Naïve Bayes Classifier

$$P(\mathbf{X}|\, C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) = P(x_1 \mid C_i) \times P(x_2 \mid C_i) \times \cdots \times P(x_n \mid C_i)$$

- If A_k is categorical, $P(x_k \mid C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)
- If A_k is continous-valued, $P(x_k \mid C_i)$ is usually computed based on Gaussian distribution $g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ and $P(x_k \mid C_i)$ is $P(\mathbf{X}|\, C_i) = g\left(x_k, \mu_{C_i}, \sigma_{C_i}\right)$

Naïve Bayes Classifier: An Example

- $P(C_i)$:
  - P(buys_computer = “yes”) = 9/14
  - P(buys_computer = “no”) = 5/14
- $P(\mathbf{X}|C_i)$ for each class
  - P(age = “<=30” | buys_computer = “yes”) = 2/9
  - P(age = “<= 30” | buys_computer = “no”) = 3/5
  - P(income = “medium” | buys_computer = “yes”) = 4/9
  - P(income = “medium” | buys_computer = “no”) = 2/5
  - P(student = “yes” | buys_computer = “yes) = 6/9
  - P(student = “yes” | buys_computer = “no”) = 1/5
  - P(credit_rating = “fair” | buys_computer = “yes”) = 6/9
  - P(credit_rating = “fair” | buys_computer = “no”) = 2/5
  - …

Naïve Bayes Classifier: An Example

P(buys_computer = “yes”)	9/14
P(buys_computer = “no”)	5/14

	buys_computer = “yes”	buys_computer = “no”
age = “<=30”	2/9	3/5
age = “31…40”	4/9	0/5
age = “>40”	3/9	2/5
income = “low”	3/9	1/5
income = “medium”	4/9	2/5
income = “high”	2/9	2/5
student = “yes”	6/9	1/5
student = “no”	3/9	4/5
credit_rating = “fair”	6/9	2/5
credit_rating = “excellent”	3/9	3/5

Naïve Bayes Classifier: An Example

age	income	student	credit_rating	buys_computer
<=30	medium	yes	fair	**?**

- $P(\mathrm{X} \mid C_i)$: P(X | buys_computer = “yes”) = 2/9 * 4/9 * 6/9 * 6/9 = 0.044
  P(X | buys_computer = “no”) = 3/5 * 2/5 * 1/5 * 2/5 = 0.019

- $P(\mathrm{X} \mid C_i) * P(C_i)$:
  P(X | buys_computer = “yes”) * P(buys_computer = “yes”) = 0.028
  P(X | buys_computer = “no”) * P(buys_computer = “no”) = 0.007

  Therefore, X belongs to class (“buys_computer = yes”)

Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional probability be **non-zero**.
  - Otherwise, the predicted probability will be zero

$$P(\mathbf{X} \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

  - For example,

age	income	student	credit_rating	buys_computer
31…40	medium	yes	fair	**?**

    - P(X | buys_computer = “yes”) = 4/9 * 4/9 * 6/9 * 6/9 > 0
    - P(X | buys_computer = “no”) = 0 * 2/5 * 1/5 * 2/5 = 0
    - Therefore, the conclusion is always **yes** regardless the value of P(X | buys_computer = “yes”)

Avoiding the Zero-Probability Problem

- Use **Laplacian correction** (or Laplacian estimator)
  - $P(C_i) = \frac{|C_i|+\mathbf{1}}{|D|+m}$
    - where m is the number of classes
  - $P(x_k \mid C_i) = \frac{|x_k \cup C_i|+1}{|C_i|+r}$
    - where $|x_k \cup C_i|$ denotes the number of tuples contains both $A_k = x_k$ and C_i, and r is the number of values of attribute A_k
  - The "corrected" probability estimates are close to their "uncorrected" counterparts

Naïve Bayes Classifier: An Example

P(buys_computer = “yes”)	10/16
P(buys_computer = “no”)	6/16

	buys_computer = “yes”	buys_computer = “no”
age = “<=30”	3/12	4/8
age = “31…40”	5/12	1/8
age = “>40”	4/12	3/8
income = “low”	4/12	2/8
income = “medium”	5/12	3/8
income = “high”	3/12	3/8
student = “yes”	7/11	2/7
student = “no”	4/11	5/7
credit_rating = “fair”	7/11	3/7
credit_rating = “excellent”	4/11	4/7

Naïve Bayes Classifier: An Example

age	income	student	credit_rating	buys_computer
<=30	medium	yes	fair	**?**

- $P(\mathrm{X} \mid C_i)$: P(X | buys_computer = “yes”) = 3/12 * 5/12 * 7/11 * 7/11 = 0.042
  P(X | buys_computer = “no”) = 4/8 * 3/8 * 2/7 * 3/7 = 0.023

- $P(\mathrm{X} \mid C_i) * P(C_i)$:
  P(X | buys_computer = “yes”) * P(buys_computer = “yes”) = 0.026
  P(X | buys_computer = “no”) * P(buys_computer = “no”) = 0.009

Therefore, X belongs to class (“buys_computer = yes”)

Handling Missing Values

- If the values of some attributes are missing, these attributes are omitted from the product of probabilities
  - As a result, the estimation is less accurate

age	income	student	credit_rating	buys_computer
?	medium	yes	fair	**?**

- $P(\mathrm{X} \mid Ci)$: P(X | buys_computer = “yes”) = 5/12 * 7/11 * 7/11 = 0.169

  P(X | buys_computer = “no”) = 3/8 * 2/7 * 3/7 = 0.046
- $P(\mathrm{X} \mid Ci) * P(Ci)$:

  P(X | buys_computer = “yes”) * P(buys_computer = “yes”) = 0.105

  P(X | buys_computer = “no”) * P(buys_computer = “no”) = 0.017

Naïve Bayes Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence $\rightarrow$ loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.

      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes
- How to deal with these dependencies?
  - Bayesian Belief Networks (Chapter 9, Data Mining: Concepts and Techniques)

Outline

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification
- **Rule-Based Classification**
- Lazy Learners (or Learning from Your Neighbors)
- Model Evaluation
- Summary

Using IF-THEN Rules for Classification

- Represent the knowledge in the form of IF-THEN rules

  R: IF age = youth AND student = yes THEN buys_computer = yes

  - Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: coverage and accuracy

$$coverage(R) = \frac{ncovers}{|D|} \qquad accuracy(R) = \frac{ncorrect}{ncovers}$$

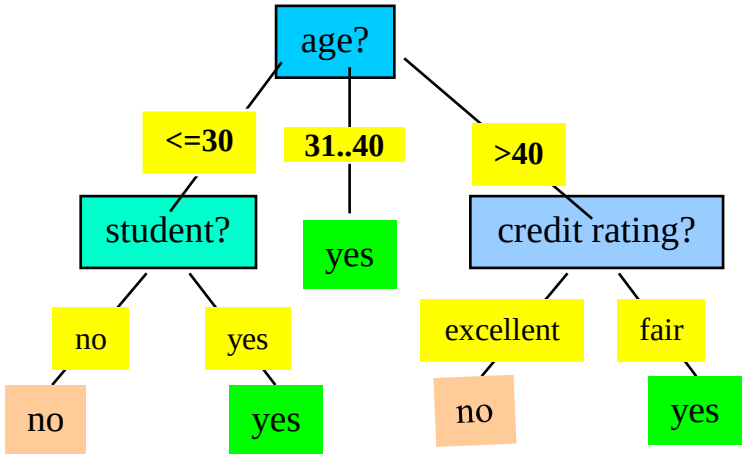
  - D: training data set
  - $ncovers$ = # of tuples covered by R
  - $ncorrect$ = # of tuples correctly classified by R

Using IF-THEN Rules for Classification

- If more than one rule are 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 tests)
  - 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 (e.g., accuracy, coverage, or size) or by experts

Rule Extraction from a Decision Tree

- Rules are easier to understand than large trees
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive



IF *age* = <=30 AND *student* = *no* THEN *buys_computer* = *no*
IF *age* = <=30 AND *student* = *yes* THEN *buys_computer* = *yes*
IF *age* = 31…40 THEN *buys_computer* = *yes*
IF *age* = >40 AND *credit_rating* = *excellent* THEN *buys_computer* = *no*
IF *age* = >40 AND *credit_rating* = *fair* THEN *buys_computer* = *yes*

Rule Induction: Sequential Covering Method

- Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Meanwhile, decision-tree induction learns a set of rules *simultaneously*

Sequential Covering Algorithm

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.

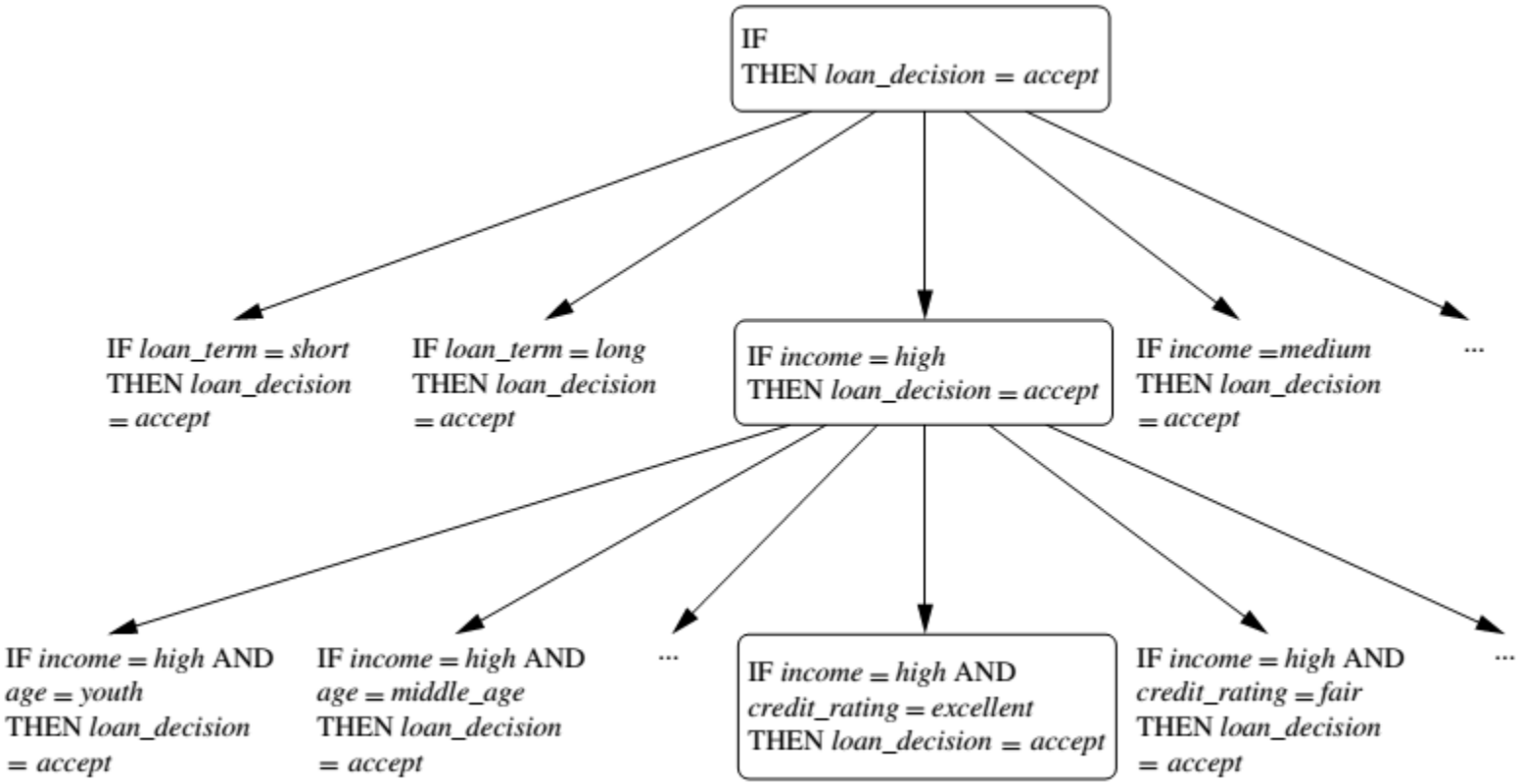
Method:

```
(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;
```

- *Termination condition*: when no more training examples or when the quality of a rule returned is below a user-specified threshold

How to Learn-One-Rule?

- Start with the *most general rule* possible: condition = empty
- *Adding new attributes* with a greedy depth-first strategy
  - Picks the one that most improves the rule quality



Rule Quality Measures

- Let R be the current rule, R' is the new rule when logically ANDing a given attribute test to condition
- Let pos (neg) and pos' (neg') be the number of positive (negative) tuples covered by R and R'
- Foil-gain (in FOIL & RIPPER) assesses info_gain by extending condition

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

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

How to Learn-One-Rule?

- Rule pruning based on an independent set of test tuples

$$FOIL\_Prune(R) = \frac{pos - neg}{pos + neg}$$

- A rule is pruned by removing a conjunct (attribute test)
- If the FOIL Prune value is higher for the pruned version of R, then prune R

Outline

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification
- Rule-Based Classification
- **Lazy Learners (or Learning from Your Neighbors)**
- Model Evaluation
- Summary

Lazy vs. Eager Learning

- Lazy learning (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
- Eager learning (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify

- Lazy learning consumes less time in training but more time in predicting

Lazy Learner: Instance-Based Methods

- Instance-based learning: Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - k-nearest neighbor (k-NN) approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference

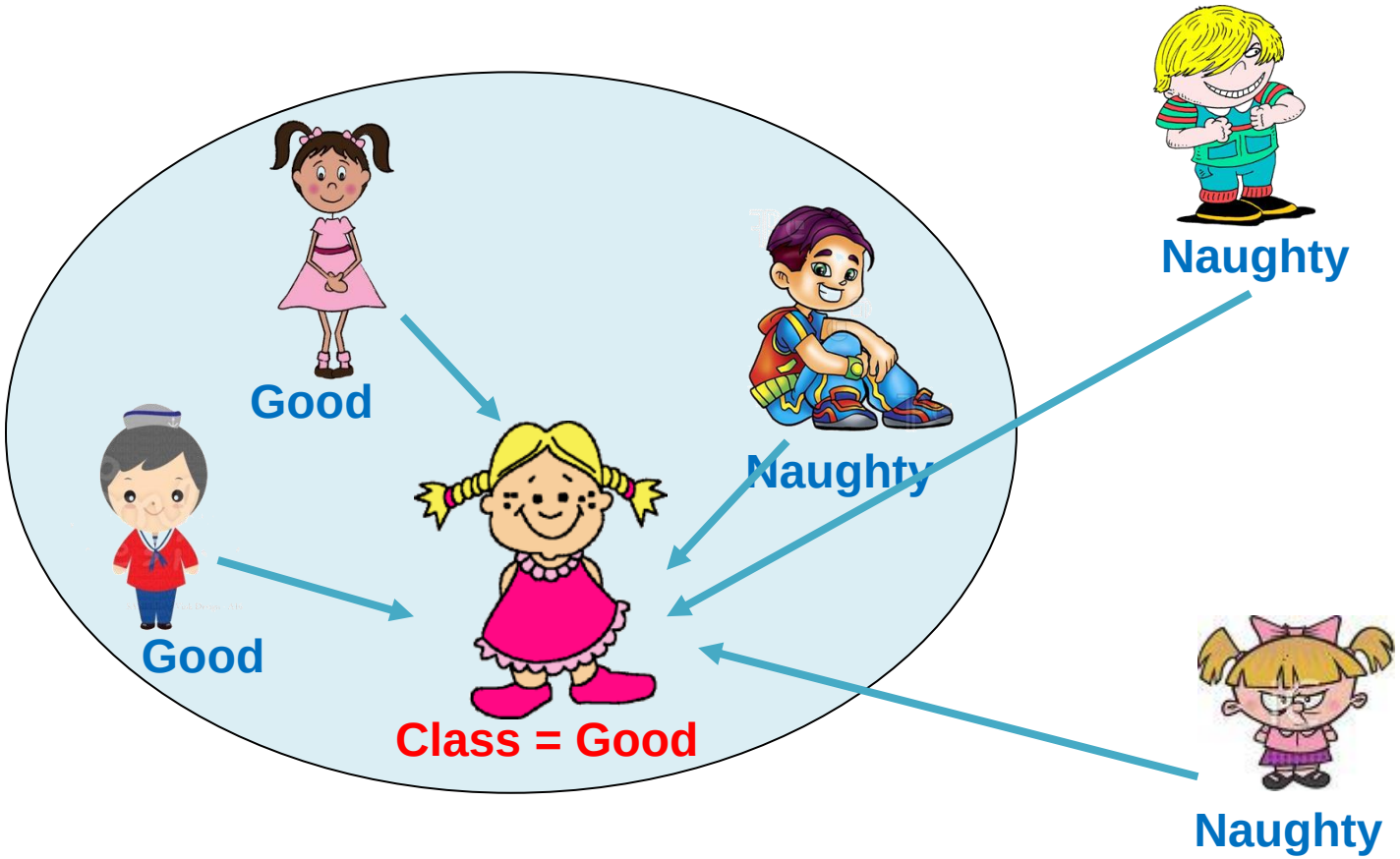
k-NN Classifier

- An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors
  - k is a positive integer, typically small
  - If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.
- The nearest neighbor are defined in terms of some distance metrics
  - E.g., Euclidean, Manhattan, etc.

Show me your friends and I'll tell you who you are

© Alex Bannykh * www.ClipartOf.com/33107

k-NN Classifier



Class = Good or Naughty?

k-NN Classifier: An Example

Age	Income (K)	No. Cards	Response	Euclidean distance to unseen record
35	35	3	Yes	22.14
22	50	2	No	20.9
28	40	1	Yes	21.35
45	100	2	No	44.11
20	30	3	Yes	34.06
34	55	2	No	8.12
63	200	1	No	145.54
55	140	2	No	85.01
59	170	1	No	115.28
25	40	4	Yes	23.37
42	**56**	**3**	**?**	

- When k = 5, there are 3 “Yes” samples and 2 “No” samples
- The unseen record belongs to the class **Yes**

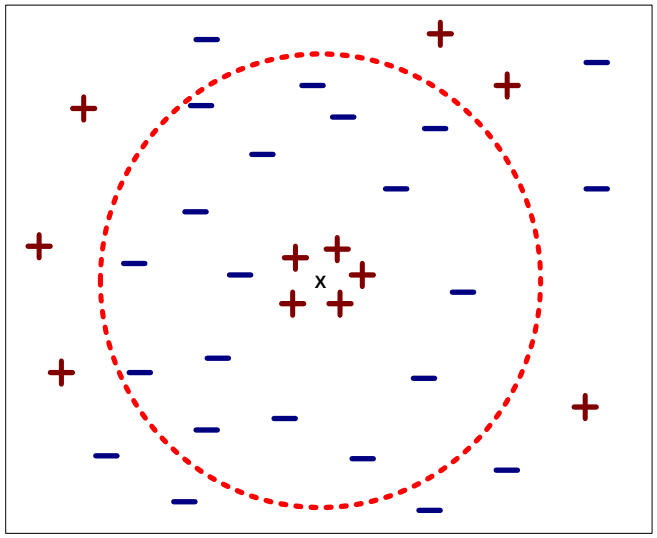
k-NN Classifier: Normalization

- Calculating distance directly on the data that attributes have different measurement scales or there is a mixture of numerical and categorical attributes, is inaccurate
  - E.g., annual income is in dollars, and age is in years then income will have a much higher influence on the distance calculated

Age	Income (K)	No. Cards	Response	Euclidean distance
0.35	0.03	0.67	Yes	0.21
0.05	0.12	0.33	No	0.57
0.19	0.06	0	Yes	0.75
0.58	0.41	0.33	No	0.43
0	0	0.67	Yes	0.53
0.33	0.15	0.33	No	0.38
1	1	0	No	1.19
0.81	0.65	0.33	No	0.67
0.91	0.82	0	No	1.03
0.12	0.06	1	Yes	0.52
0.51	0.15	0.67	**No**	

Discussion on the k-NN Algorithm

- Advantages
  - Easy to use and implement
  - Robust to noisy data by averaging k nearest neighbors



- Disadvantages
  - All samples are stored
  - It is time-consuming to determine the class for an unseen record
  - Heavily depending on the value of k
    - If k is too small, there is insufficient information for making decision
    - If k is too large, noisy values may be included or the neighborhood will violate the areas of other classes

Case-Based Reasoning

- Case-based reasoning (CBR) classifier uses a database of problem solutions to solve new problems
- It stores the tuples or “cases” for problem solving as complex symbolic descriptions
- For example,
  - Customer service help desks: cases describe product-related diagnostic problems
  - Engineering and law: cases are either technical designs or legal rulings, respectively
  - Medical education: patient case histories and treatments are used to help diagnose and treat new patients

Case-Based Reasoning (CBR)

- When given a new case to classify, a case-based reasoner will first check if an identical training case exists
  - If one is found, the accompanying solution to that case is returned
  - If no identical case is found, the CBR will search for training cases having components that are similar to those of the new case.
- CBRs use background knowledge and problem-solving strategies to propose a feasible combined solution

- Challenges
  - Find a good similarity metric
  - Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases

Outline

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification
- Rule-Based Classification
- Lazy Learners (or Learning from Your Neighbors)
- **Model Evaluation**
- Summary

Model Evaluation

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use validation test set of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating the accuracy of a classifier
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap

Classifier Evaluation Metrics: Confusion Matrix

- Confusion Matrix

		Predicted class	
		C_1	$\neg C_1$
Actual Class	C_1	**True Positives (TP)**	**False Negatives (FN)**
	$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

- Given m classes, an entry, $CM_{i,j}$ in a confusion matrix indicates # of tuples in class i that were labeled by the classifier as class j
- May have extra rows/columns to provide totals

Classifier Evaluation Metrics: Confusion Matrix

- Example of Confusion Matrix:

		Predicted class		Total
		buy_computer = yes	buy_computer = no	
Actual Class	buy_computer = yes	**6954**	**46**	7000
	buy_computer = no	**412**	**2588**	3000
Total		7366	2634	10000

Accuracy, Error Rate, Sensitivity

- Classifier accuracy: $Accuracy = (TP + TN)/All$
- Error rate: $Error\ rate = 1 - Accuracy = \frac{FP+FN}{All}$

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

- Class Imbalance Problem:
  - One class may be rare, e.g. fraud, or HIV-positive
  - Significant majority of the negative class and minority of the positive class
  - Sensitivity: True Positive recognition rate $Sensitivity = TP/P$
  - Specificity: True Negative recognition rate $Specificity = TN/N$

Precision and Recall, and F-measure

- Precision: exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- Recall: completeness – what % of positive tuples did the classifier label as positive?

$$recall = \frac{TP}{TP + FN}$$

- Perfect score is 1.0
- Inverse relationship between precision & recall

Precision and Recall, and F-measure

- F-measure (F1 or F-score): harmonic mean of precision and recall

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- F_β: weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

Classifier Evaluation Metrics: An Example

		Predicted class		Total	Recognition (%)
		cancer = yes	cancer = no		
Actual Class	cancer = yes	**90**	**210**	300	30.00 (*sensitivity*)
	cancer = no	**140**	**9560**	9700	98.56 (*specificity*)
Total		230	9770	10000	96.40 (*accuracy*)

- *Precision* = 90/230 = 39.13%
- *Recall* = 90/300 = 30.00%

Evaluating Classifier Accuracy

- Holdout method
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- Cross-validation (k-fold, where k = 10 is most popular)
  - Randomly partition the data into k mutually exclusive subsets, each approximately equal size
  - At ith iteration, use D_i as test set and others as training set
  - Leave-one-out: k folds where k = # of tuples, for small sized data
  - Stratified cross-validation: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

Evaluating Classifier Accuracy

- Bootstrap
  - Works well with small data sets
  - Samples the given training tuples uniformly with replacement
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- .632 boostrap is the most common method
  - A data set with d tuples is sampled d times, with replacement, resulting in a training set of d samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx$ e-1 = 0.368)
  - Repeat the sampling procedure k times, overall accuracy of the model:

$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k}(0.632 \times Acc(M_i)_{test\_set} + 0.632 \times Acc(M_i)_{train\_set})$$

Outline

- Classification: Basic Concepts
- Decision Tree Induction
- Naïve Bayes Classification
- Rule-Based Classification
- Lazy Learners (or Learning from Your Neighbors)
- Model Evaluation
- Summary

Summary

- Basic concepts of classification
- Performing classification tasks using
  - Decision tree induction: ID3
  - Bayes classification: Naïve Bayes
  - Rule-based classification: extracting rules from a decision tree, Sequential covering method
  - Lazy learner (or learning from your neighbor): k-nearest neighbor, case-based classification
- Measures of impurity: entropy, information gain, information gain ratio and Gini index
- Classifier evaluation metrics: confusion matrix, accuracy, error Rate, sensitivity, precision-recall, F-measure
- Evaluating classifier accuracy: hold-out, cross-validation, bootstrap