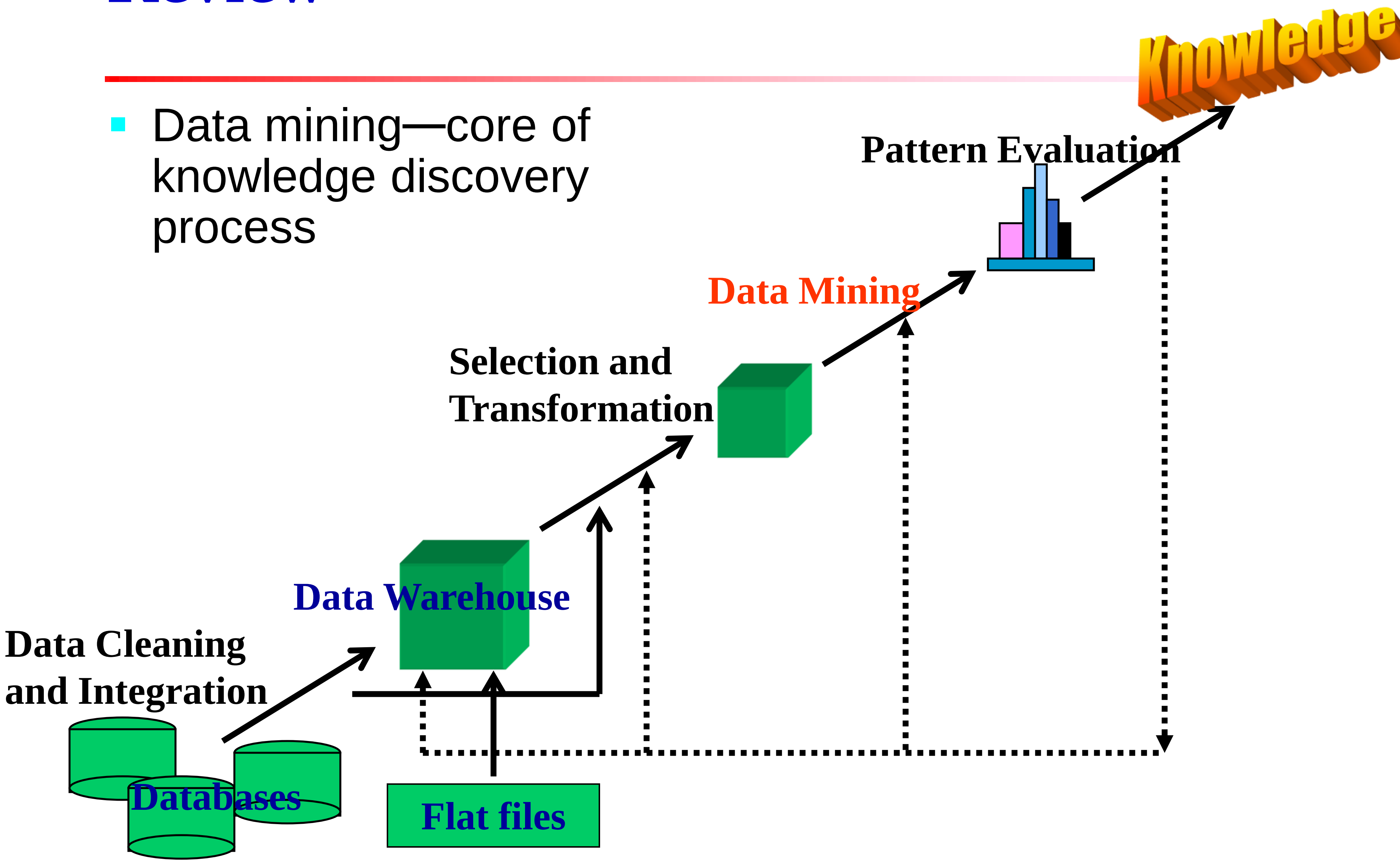
Data Mining

Ying Liu, Prof., Ph.D

School of Computer Science and Technology
University of Chinese Academy of Sciences
Data Mining and High Performance Computing Lab

Review

- Data mining—core of knowledge discovery process



Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Other classification methods
- Prediction
- Accuracy and error measures
- Summary

Classification vs. Prediction

- Classification
  - Predict categorical class labels (discrete or nominal)
  - Classify records (constructs a model) based on the training set and the class labels in a classifying attribute and then use the rules to classify new records
- Prediction
  - Model continuous-valued functions, i.e., predict unknown or missing values
- Typical applications
  - Credit approval
  - Target marketing
  - Medical diagnosis
  - Fraud detection
  - Intrusion detection

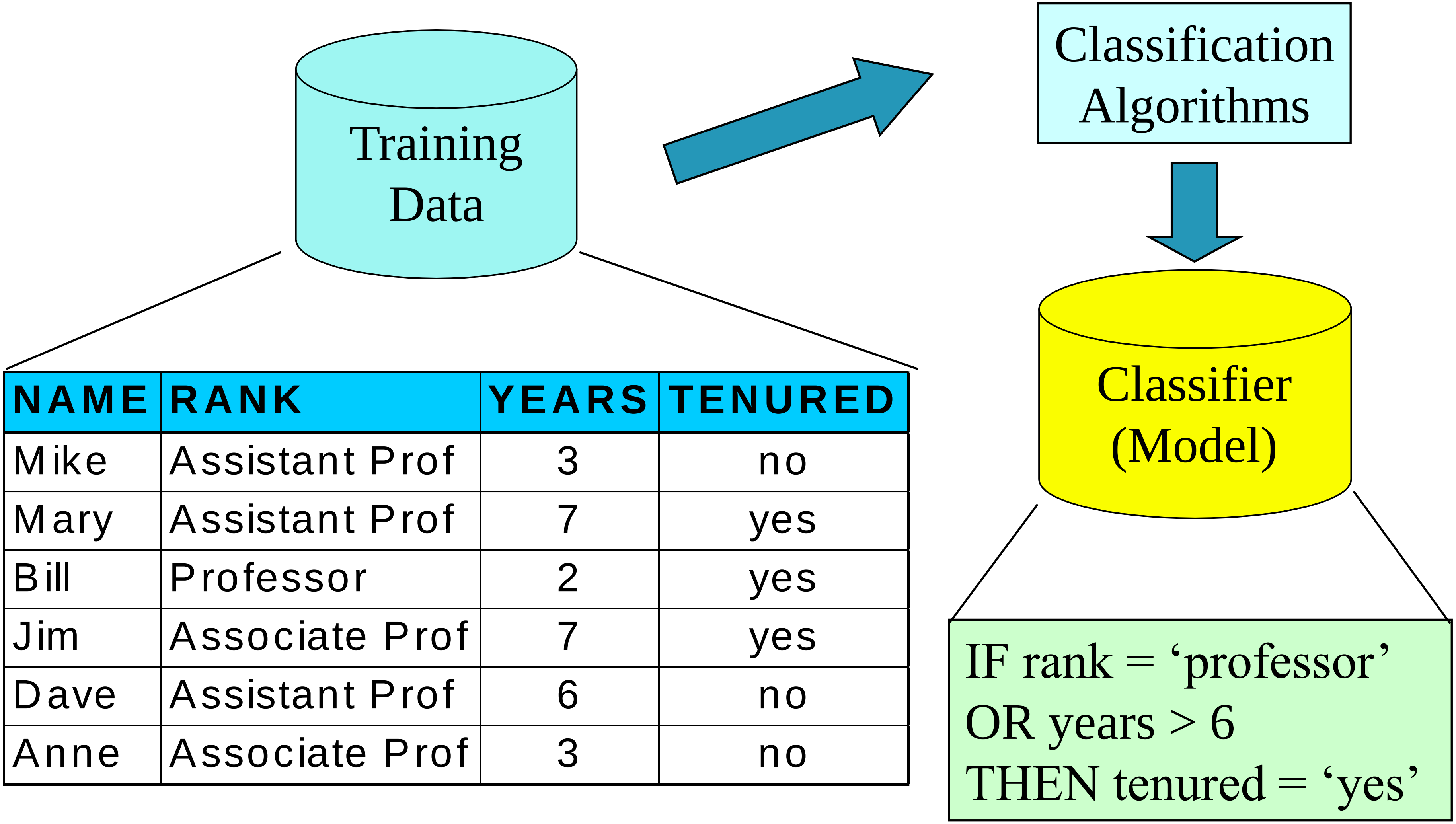
Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae

Classification—A Two-Step Process

- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

Process (1): Model Construction



Training Data

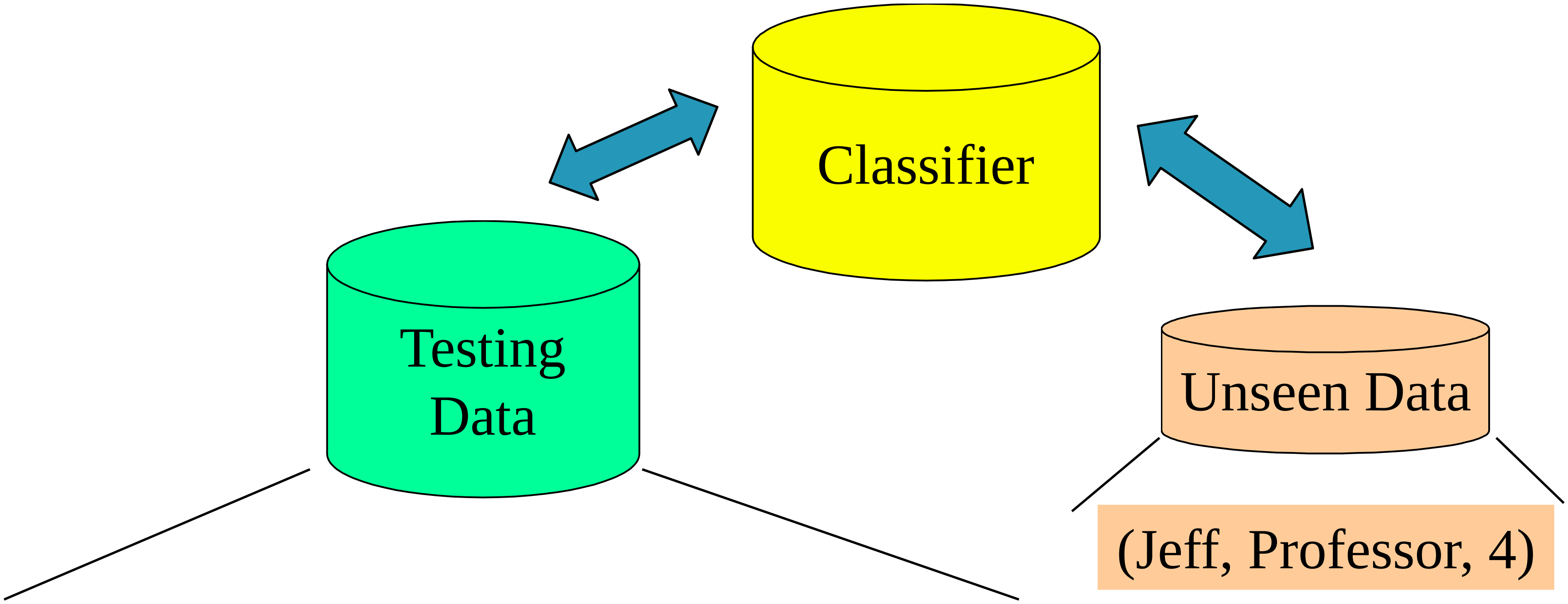
Classification Algorithms

Classifier (Model)

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Process (2): Using the Model in Classification



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Tenured?

Yes

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, establish classes or clusters in the data

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Other classification methods
- Prediction
- Accuracy and error measures
- Summary

Issues: Data Preparation

- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
- Data transformation
  - Generalize and/or normalize data

Issues: Evaluating Classification Methods

- Accuracy
  - classifier accuracy: predicting class label
  - predictor accuracy: guessing value of predicted attributes
- Speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Other classification methods
- Prediction
- Accuracy and error measures
- Summary

Classification by Decision Tree Induction

- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a splitting test on an attribute
  - Branch represents an outcome of the test
  - Leaf node represents class distribution
- Decision tree generation -- two phases
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample

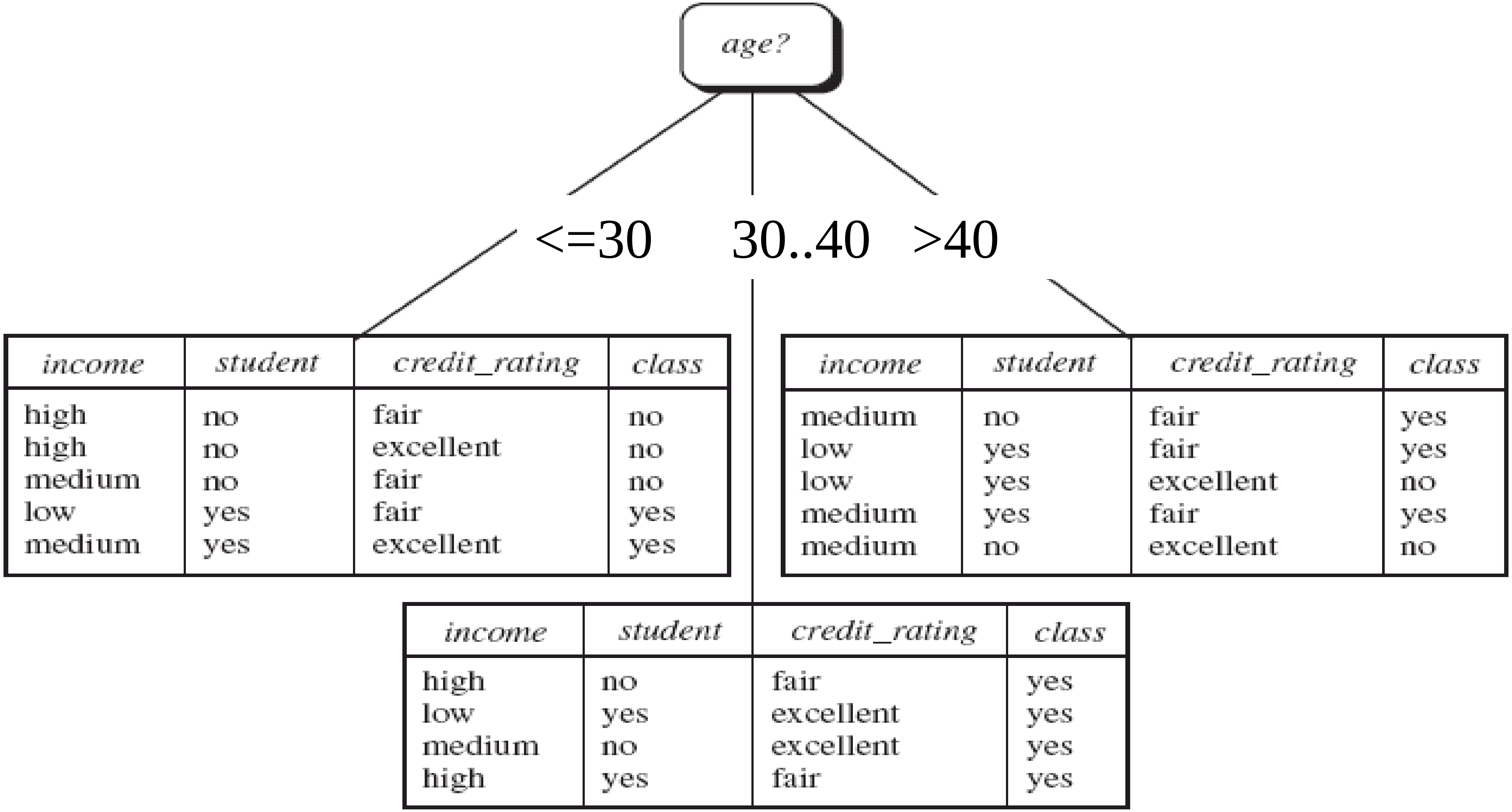
Classification by Decision Tree Induction

```
Generate_decision_tree ($D$, $attribute\_list$)
create a node $N$;
if tuples in $D$ are all of the same class, $C$ then
    return $N$ as a leaf node labeled with the class $C$;
if $attribute\_list$ is empty then
    return $N$ as a leaf node labeled with the majority class in $D$; // majority voting
apply Attribute_selection_method($D$, $attribute\_list$) to find the highest information gain;
label node $N$ with $test\text{-}attribute$;
for each value $a_i$ of $test\text{-}attribute$  // partition the tuples and grow subtrees for each partition
    Grow a branch from node $N$ for $test\text{-}attribute = a_i$;        // a partition
    Let $s_i$ be the set of samples in $D$ for which test-attribute = $a_i$ ;
    if $s_i$ is empty then
        attach a leaf labeled with the majority class in $D$ to node $N$;
    else attach the node returned by Generate_decision_tree($s_i$ , $attribute\_list$) to node $N$;
end for
```

Decision Tree Induction: Training Dataset

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

Decision Tree



age?

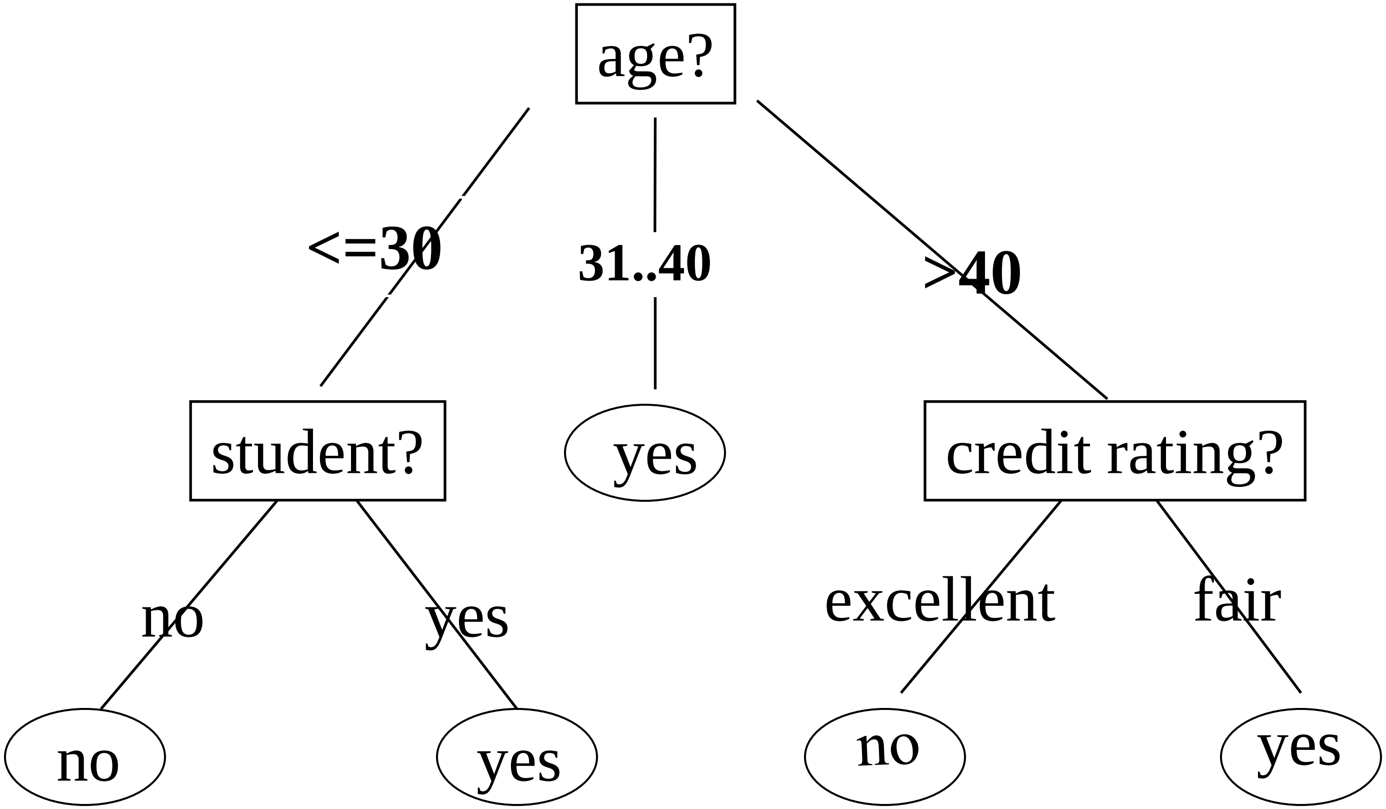
<=30 30..40 >40

income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	*student*	*credit_rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

Output: A Decision Tree for "*buys_computer*"



Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are 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, Gini index)
- 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

Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Assume there are two classes, *P* and *N*
  - Let the set of examples *S* contain *p* elements of class *P* and *n* elements of class *N*
  - The amount of information, needed to classify sample

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Information Gain in Decision Tree Induction

- Assume that attribute A have v distinct values, $\{a_1, a_2, \ldots, a_v\}$
- Training set S will be partitioned into sets $\{S_1, S_2, \ldots, S_v\}$
  - If S_i contains p_i examples of P and n_i examples of N, the entropy, or the expected information based on the partitioning into subsets by attribute A is

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- Information gain of A

$$Gain(A) = I(p,n) - E(A)$$

Attribute Selection by Information Gain Computation

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”
- I(p, n) = I(9, 5) =0.940
- Compute the entropy for *age*:

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
30…40	4	0	0
>40	3	2	0.971

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0)$$

$$+ \frac{5}{14} I(3,2) = 0.694$$

Hence

$$Gain(age) = I(p,n) - E(age)$$

$$= 0.246$$

Exercise

1. Please calculate the information gain of *income*, *student*, and *credit_rating*, respectively.

- Gain(income) = 0.029
- Gain(Student) = 0.151
- Gain(credit_rating) = 0.048

Gain Ratio for Attribute Selection (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)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

$$SplitInfo_A(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 0.926$$

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex.
  - gain_ratio(income) = 0.029/0.926 = 0.031
- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini Index (CART, IBM Intelligent Miner)

- If a data set *T* contains examples from *n* classes, gini index, *gini*(*T*) is defined as

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

where p_j is the relative frequency of class *j* in *T.*

- If a data set *T* is split into two subsets T_1 and T_2 with sizes N_1 and N_2 respectively, the *gini* index of the split data contains examples from *n* classes, the *gini* index of the split is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute provides the smallest $gini_{split}(T)$ is chosen to split the node (*need to enumerate all possible splitting points for each attribute*).

Gini index (CART, IBM IntelligentMiner)

- The lowest is the best
- All attributes are assumed continuous-valued
- Can be modified for categorical attributes
- Ex. D has 9 tuples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1: {medium, high} and 4 in D_2

$$gini_{income \in \{medium, high\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$= \frac{10}{14}(1 - (\frac{6}{10})^2 - (\frac{4}{10})^2) + \frac{4}{14}(1 - (\frac{1}{4})^2 - (\frac{3}{4})^2)$$

$$= 0.450$$

Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- 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 node holds the class prediction
- Rules are easier for humans to understand
- Example

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* = “<=30” AND *credit_rating* = “*fair*” THEN *buys_computer* = “yes”

Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”

Evaluating Classifier Accuracy

- Holdout
  - Train on 2/3
  - Test on 1/3
- Cross validation: *k*-fold cross validation
  - Partition data set into *k* parts
  - Train on random (*k*-1) parts, test on 1 part
  - Repeat *k* times
  - Average accuracy

Comments on Decision Tree Induction

- Relatively faster learning speed (than other classification methods)
- Convertible to simple and easy to understand classification rules
- Comparable classification accuracy with other methods
- Comparably scalable to large database

Enhancements to Basic Decision Tree Induction

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
- Attribute construction
  - Create new attributes based on existing ones

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Other classification methods
- Prediction
- Accuracy and error measures
- Summary

Bayesian Classification

- A statistical classifier

  Perform *probabilistic prediction, i.e.,* predict class membership probabilities

- Foundation

  Based on Bayes' Theorem

- Assumption

  The effect of an attribute on a given class is independent of other attributes

- Performance

  A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers

Bayesian Theorem: Basics

- Let *X* be a data sample, class label is unknown
- Let *H* be a *hypothesis, e.g. X* belongs to class C
- Classification is to determine P(*H*|*X*), the probability that the hypothesis holds given the observed data sample *X*
- P(*H*), the initial probability
  - E.g., *X* will buy computer, regardless of age, income, …
- P(*X*): probability that sample data is observed
- P(*X*|*H*), the probability of observing the sample *X*, given that the hypothesis holds
  - E.g., Given that *X* will buy computer, what is the prob. that *X* is 31..40?

Bayesian Theorem

- Given training data *X*, probability of a hypothesis *H,* P(*H*|*X*)*,* follows the Bayes Theorem

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

- Predict *X* belongs to C_i iff the probability $P(C_i|X)$ is the highest among all the $P(C_k|X)$ for all the *k* classes
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Naïve Bayesian Classifier

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector *X* = (x_1, x_2, …, x_n)
- Suppose there are *m* classes C_1, C_2, …, C_m
- Classification is to derive the maximum posteriori, i.e., the maximal P(C_i|*X*)
- This can be derived from Bayes Theorem

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

- Since P(*X*) is constant for all classes, only

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

  needs to be maximized

- P(C_i) can be obtained from training data set s_i/s

Derivation of Naïve Bayes Classifier

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

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

- This greatly reduces the computation cost: Only counts the class distribution
- If A$_k$ is categorical, P(x$_k$|C$_i$) = s_{ik}/s_i, count the distribution
- If A$_k$ is continuous-valued, P(x$_k$|C$_i$) is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and P(x$_k$|C$_i$) is

$$P(X_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Exercise

Predict what class does the data sample
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair) belong to?

Class:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

age	income	student	credit_rating	comp
<=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

Solution

- $P(C_i)$: P(buys_computer = “yes”) = 9/14 = 0.643
  P(buys_computer = “no”) = 5/14= 0.357

- Compute $P(X|C_i)$ for each class
  P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222
  P(age = “<= 30” | buys_computer = “no”) = 3/5 = 0.6
  P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444
  P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4
  P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667
  P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2
  P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667
  P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = “yes”) = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = “no”) = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: 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”)

Naïve Bayesian Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies do 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 Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

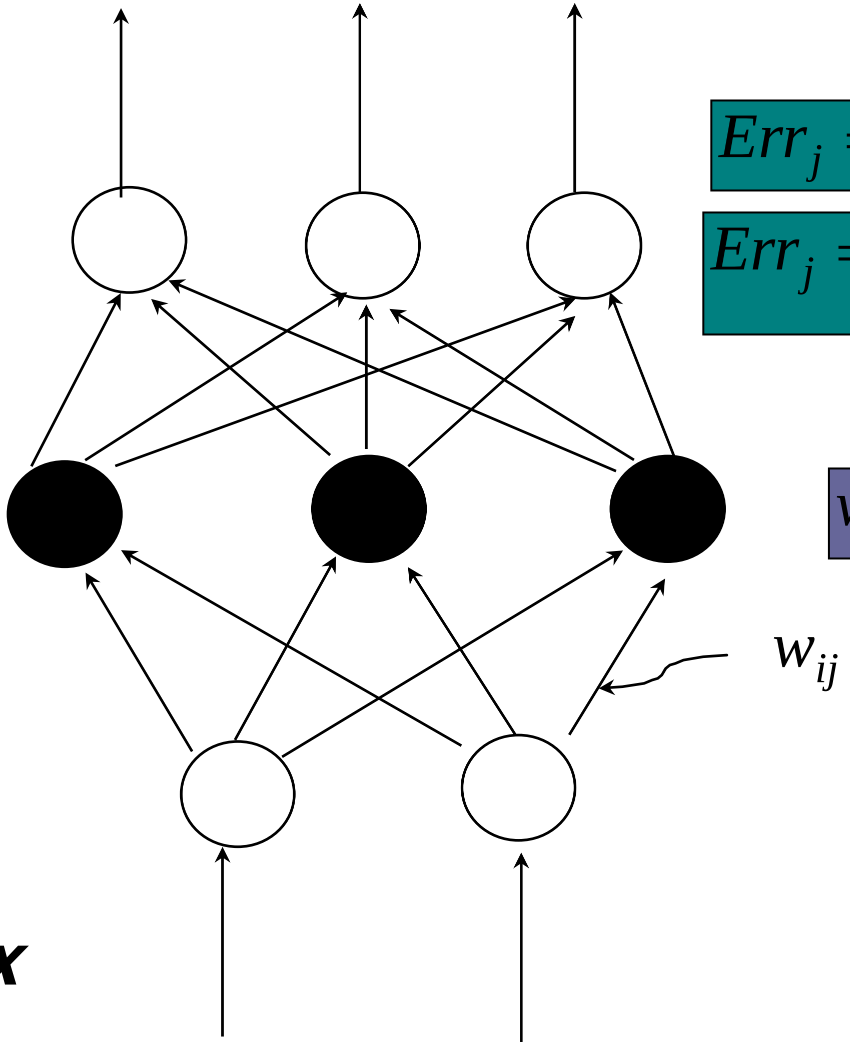
Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Other classification methods
- Prediction
- Accuracy and error measures
- Summary

Classification by Backpropagation

- Backpropagation: A **neural network** learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where each connection has a **weight** associated with it
- During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class label of the input tuples

A Multi-Layer Feed-Forward Neural Network



Output vector

Output layer

Hidden layer

Input layer

Input vector: ***X***

$$Err_j = O_j(1-O_j)(T_j - O_j)$$

$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$$

$$\theta_j = \theta_j + (l)Err_j$$

$$w_{ij} = w_{ij} + (l)Err_j O_i$$

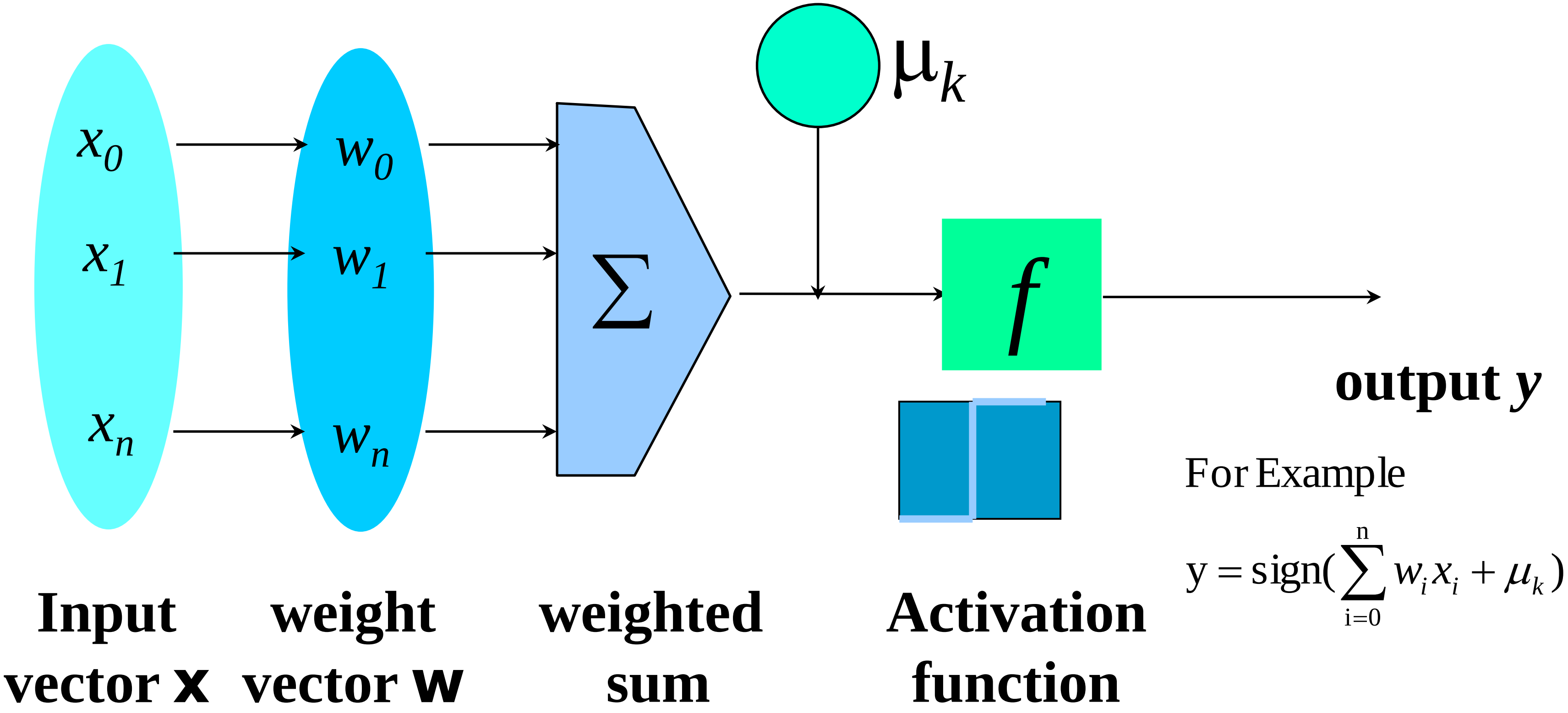
$$O_j = \frac{1}{1+e^{-I_j}}$$

$$I_j = \sum_i w_{ij} O_i + \theta_j$$

Defining a Network Topology

- First decide the **network topology:** # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer*
- Normalizing the input values for each attribute measured in the training tuples to [0.0—1.0]
- For categorical attribute, one **input** unit per domain value, each initialized to 0
- **Output**, if for classification and more than two classes, one output unit per class is used
- Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a *different network topology* or a *different set of initial weights*

A Neuron (= a perceptron)



For Example

$$y = \text{sign}(\sum_{i=0}^{n} w_i x_i + \mu_k)$$

- The *n*-dimensional input vector **x** is mapped into variable y by means of the scalar product and a nonlinear function mapping

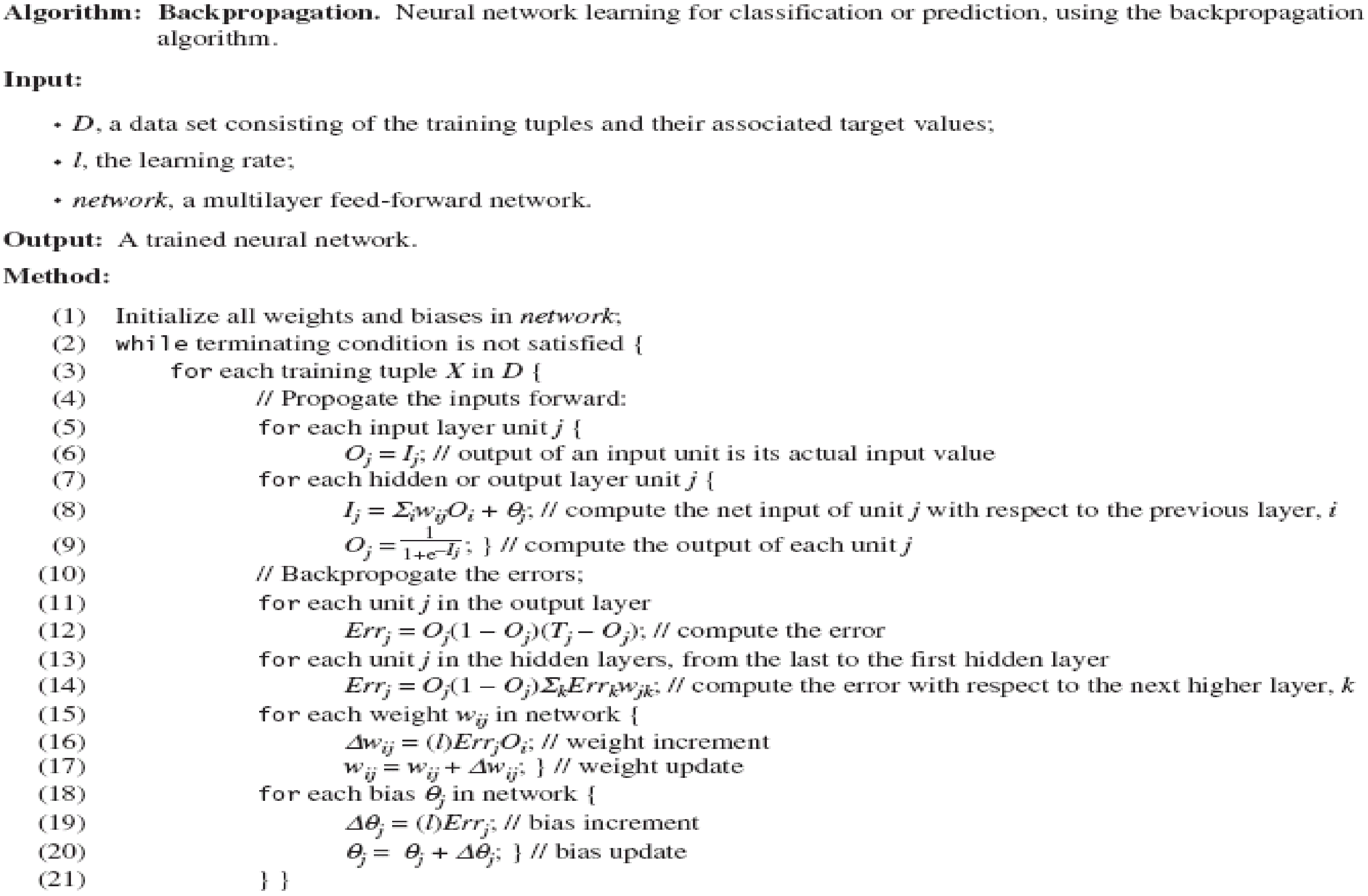
How A Multi-Layer Neural Network Works?

- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction
- The network is **feed-forward** in that none of the weights cycles back to an input unit or to an output unit of a previous layer
- From a statistical point of view, networks perform **nonlinear regression**: Given enough hidden units and enough training samples, they can closely approximate any function

Backpropagation

- Initialize weights and biases as random numbers
- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to minimize the mean squared error between the network's prediction and the actual target value
- Modifications are made in the “backwards” direction: from the output layer, through each hidden layer down to the first hidden layer, hence “backpropagation”

Backpropagation



Algorithm: Backpropagation. Neural network learning for classification or prediction, using the backpropagation algorithm.

Input:

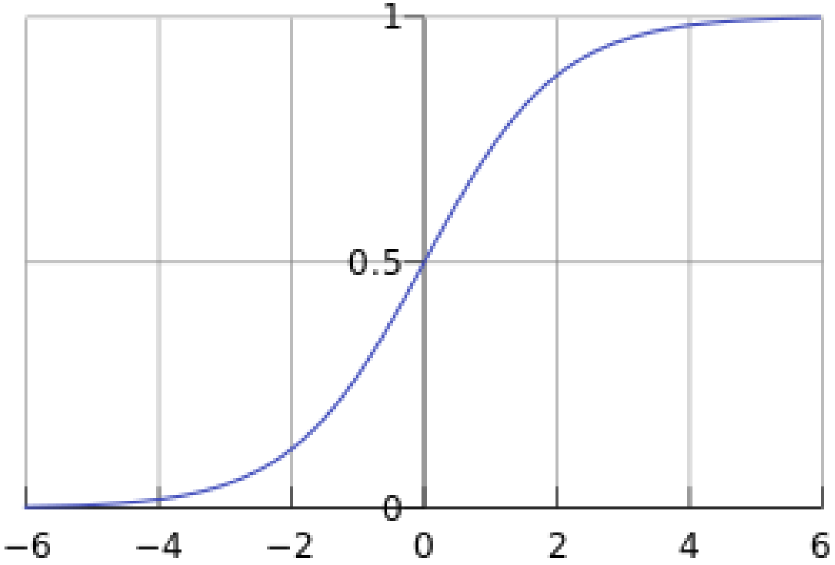
- *D*, a data set consisting of the training tuples and their associated target values;
- *l*, the learning rate;
- *network*, a multilayer feed-forward network.

Output: A trained neural network.

Method:

```
Initialize all weights and biases in network;
while terminating condition is not satisfied {
    for each training tuple $X$ in $D$ {
        // Propogate the inputs forward:
        for each input layer unit $j$ {
            $O_j = I_j$; // output of an input unit is its actual input value
        for each hidden or output layer unit $j$ {
            $I_j = \Sigma_i w_{ij} O_i + \theta_j$; // compute the net input of unit $j$ with respect to the previous layer, $i$
            $O_j = \frac{1}{1+e^{-I_j}}$; } // compute the output of each unit $j$
        // Backpropogate the errors;
        for each unit $j$ in the output layer
            $Err_j = O_j(1 - O_j)(T_j - O_j)$; // compute the error
        for each unit $j$ in the hidden layers, from the last to the first hidden layer
            $Err_j = O_j(1 - O_j)\Sigma_k Err_k w_{jk}$; // compute the error with respect to the next higher layer, $k$
        for each weight $w_{ij}$ in network {
            $\Delta w_{ij} = (l) Err_j O_i$; // weight increment
            $w_{ij} = w_{ij} + \Delta w_{ij}$; } // weight update
        for each bias $\theta_j$ in network {
            $\Delta\theta_j = (l) Err_j$; // bias increment
            $\theta_j = \theta_j + \Delta\theta_j$; } // bias update
        } }
```

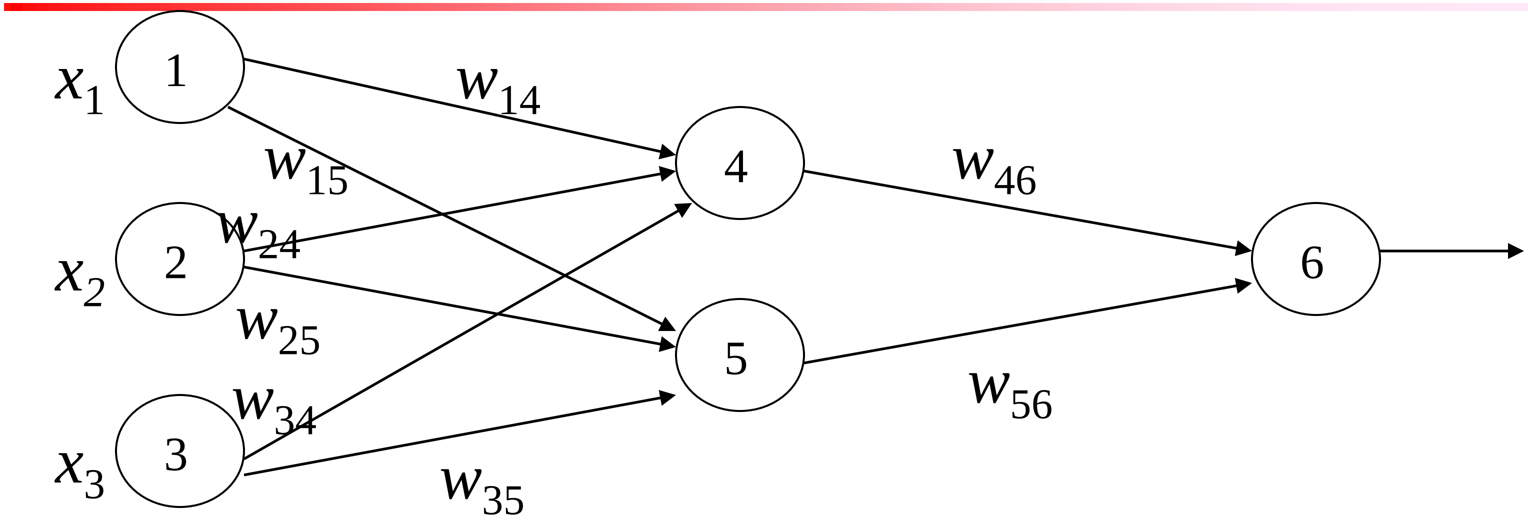
$$g(z) = \frac{1}{1 + e^{-z}}$$

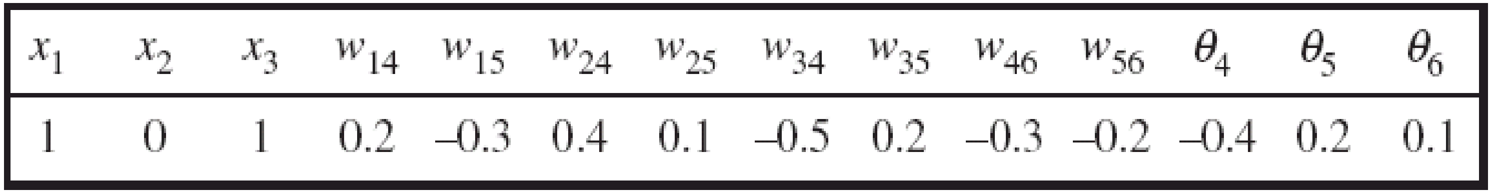


Backpropagation

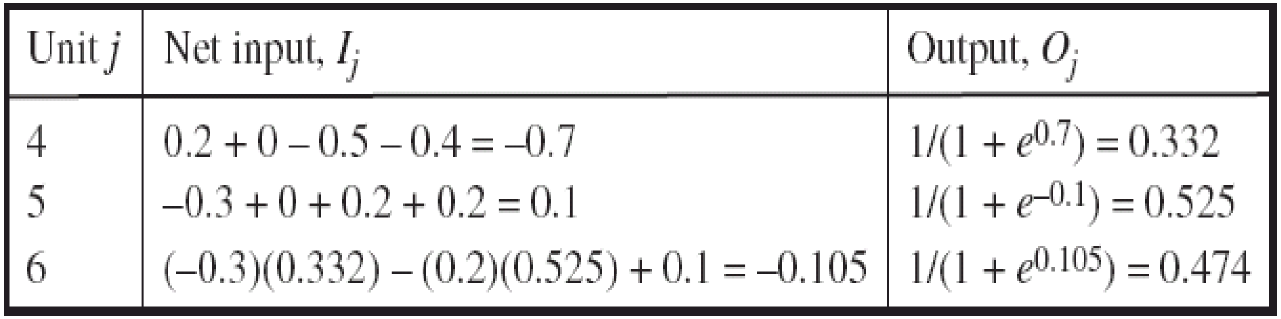
- Steps
  - Initialize weights (to small random #s) and biases in the network
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

Exercise



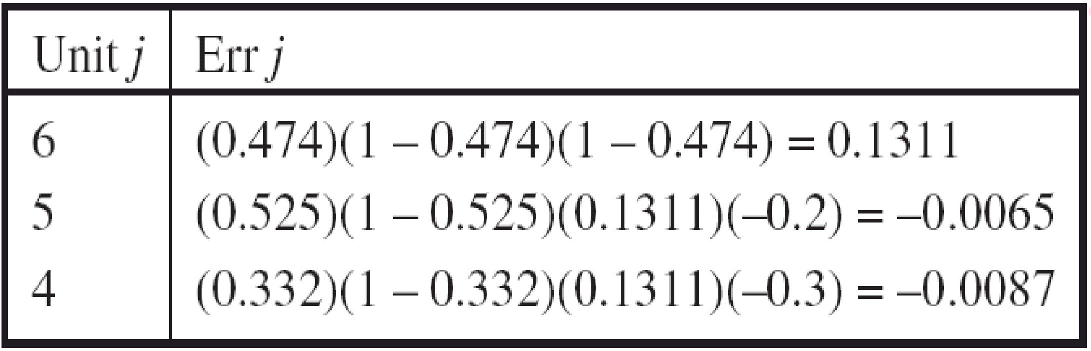


x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	–0.3	0.4	0.1	–0.5	0.2	–0.3	–0.2	–0.4	0.2	0.1

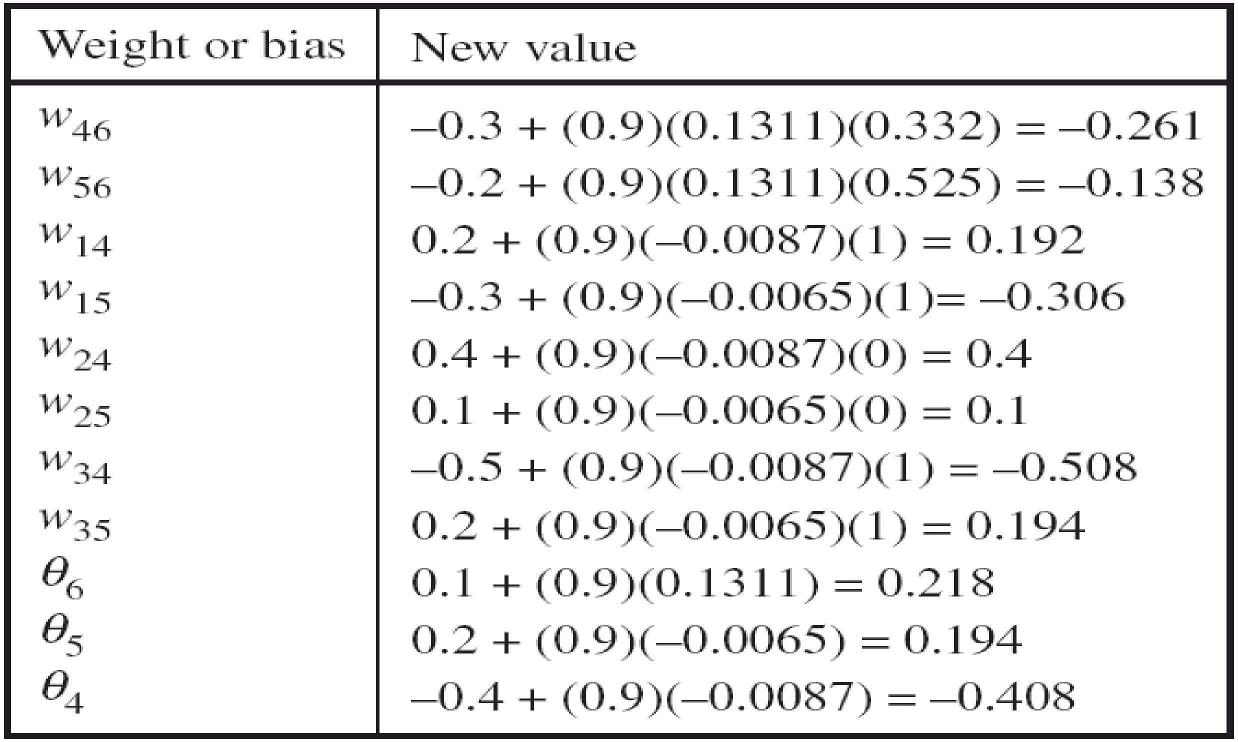


Unit j	Net input, I_j	Output, O_j
4	$0.2 + 0 - 0.5 - 0.4 = -0.7$	$1/(1 + e^{0.7}) = 0.332$
5	$-0.3 + 0 + 0.2 + 0.2 = 0.1$	$1/(1 + e^{-0.1}) = 0.525$
6	$(-0.3)(0.332) - (0.2)(0.525) + 0.1 = -0.105$	$1/(1 + e^{0.105}) = 0.474$

Exercise



Unit *j*	Err *j*
6	(0.474)(1 – 0.474)(1 – 0.474) = 0.1311
5	(0.525)(1 – 0.525)(0.1311)(–0.2) = –0.0065
4	(0.332)(1 – 0.332)(0.1311)(–0.3) = –0.0087



Weight or bias	New value
w_{46}	–0.3 + (0.9)(0.1311)(0.332) = –0.261
w_{56}	–0.2 + (0.9)(0.1311)(0.525) = –0.138
w_{14}	0.2 + (0.9)(–0.0087)(1) = 0.192
w_{15}	–0.3 + (0.9)(–0.0065)(1)= –0.306
w_{24}	0.4 + (0.9)(–0.0087)(0) = 0.4
w_{25}	0.1 + (0.9)(–0.0065)(0) = 0.1
w_{34}	–0.5 + (0.9)(–0.0087)(1) = –0.508
w_{35}	0.2 + (0.9)(–0.0065)(1) = 0.194
θ_6	0.1 + (0.9)(0.1311) = 0.218
θ_5	0.2 + (0.9)(–0.0065) = 0.194
θ_4	–0.4 + (0.9)(–0.0087) = –0.408

Backpropagation and Interpretability

- Rule extraction from networks: network pruning
  - Simplify the network structure by removing weighted links that have the least effect on the trained network
  - The set of input and activation values are studied to derive rules describing the relationship between the input and hidden unit layers
- Sensitivity analysis: assess the impact that a given input variable has on a network output. The knowledge gained from this analysis can be represented in rules

Neural Network as a Classifier

- Strength
  - High tolerance to noisy data
  - Well-suited for continuous-valued inputs and outputs
  - Successful on a wide array of real-world data
  - Techniques have recently been developed for the extraction of rules from trained neural networks

Neural Network as a Classifier

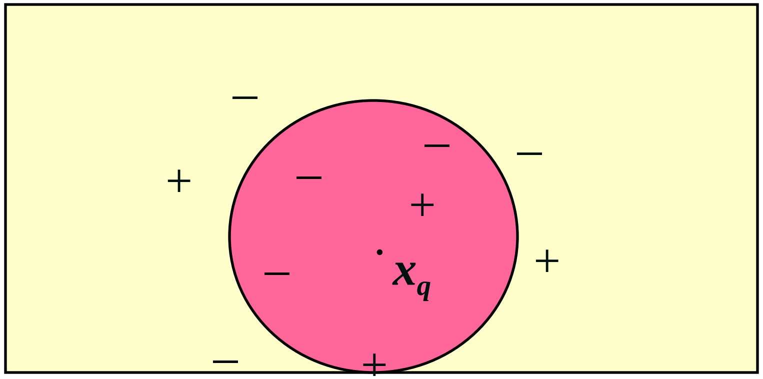
- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology or “structure”
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of “hidden units” in the network

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Other classification methods
- Prediction
- Accuracy and error measures
- Summary

The *k*-Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space
- The nearest neighbor is defined in terms of Euclidean distance, dist(X_1, X_2)
- Target function could be discrete- or real- valued
- For discrete-valued, *k*-NN returns the most common value among the *k* training examples nearest to x_q



Exercise

1. Consider the one-dimensional data set. Please classify the data point x=5.0 according to its 1-, 3-, and 5-nearest neighbors (using majority vote).

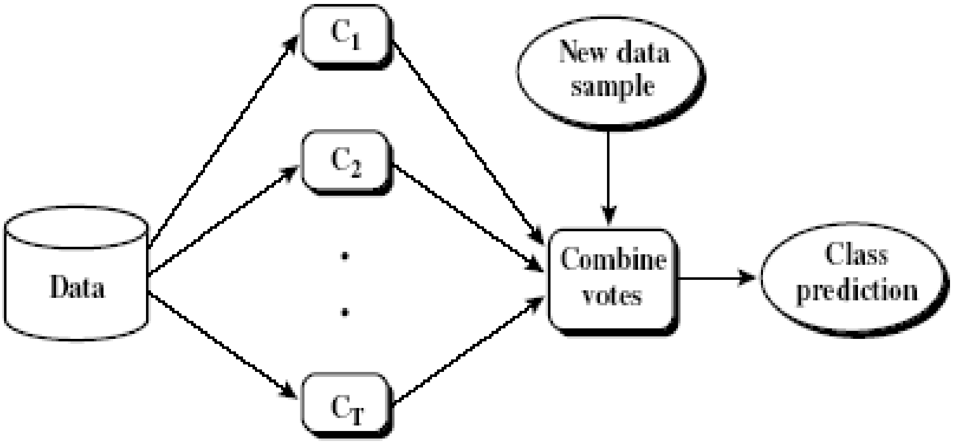
x	0.5	3.0	4.5	4.6	4.9	5.2	5.3	5.5	7.0	9.5
y	-	-	+	+	+	-	-	+	-	-

Discussion on the *k*-NN Algorithm

- k-NN for real-valued prediction for a given unknown tuple
  - Returns the mean values of the *k* nearest neighbors
- Robust to noisy data by averaging k-nearest neighbors
- Distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, eliminate irrelevant attributes
- Lazy-learner
  - Not build a classifier
  - Store all the training samples
  - High computational cost for each new tuple

Ensemble Methods: Increasing the Accuracy

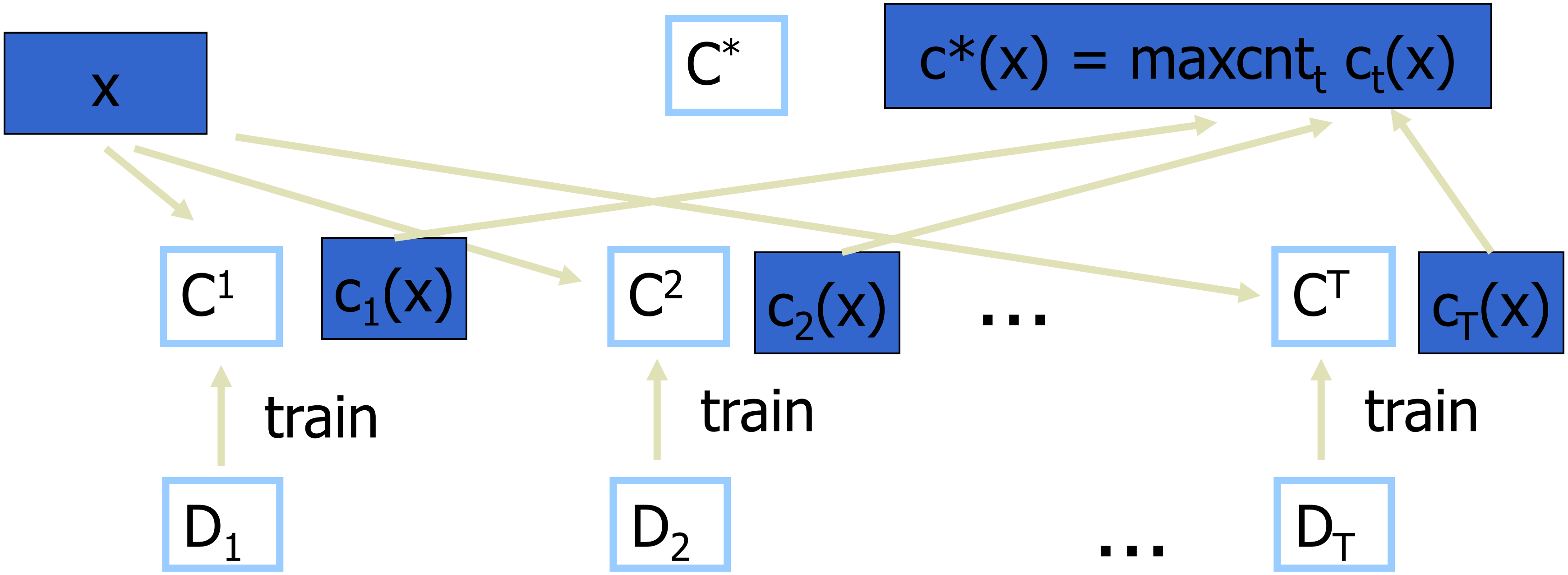
- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging
  - Boosting



Bagging: Boostrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i is sampled with replacement from D
  - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample *X*
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to *X*

Bagging: Boostrap Aggregation



Bagging: Boostrap Aggregation

- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significant better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

Exercise

1. Following is a data set to construct a bagging classifier.

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

Examples chosen for training in each round are shown below:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

$0.35 < x \leq 0.75 \Rightarrow y=-1$

x	0.1	0.2	0.3	0.5	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

$0.4 < x \leq 0.65 \Rightarrow y=-1$

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

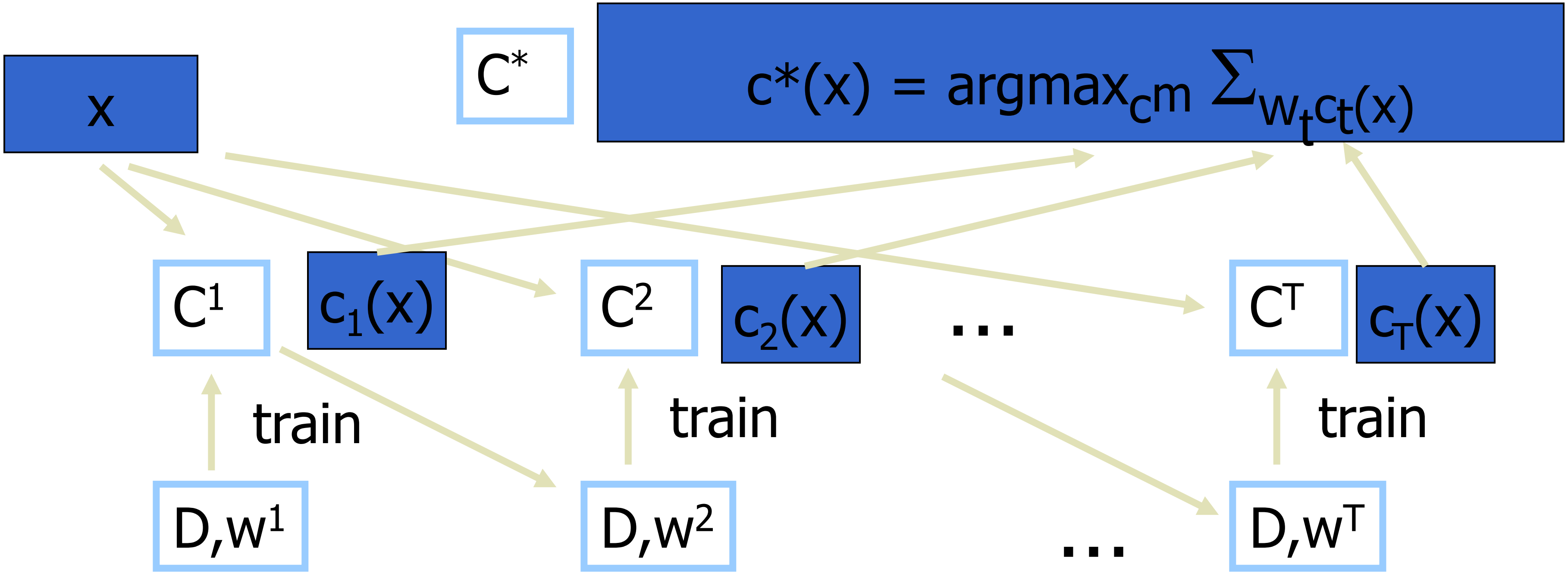
$0.35< x \leq 0.75 \Rightarrow y=-1$

Please predict the class label for the record x=0.38.

Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses — weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, pay more attention to the training tuples that were misclassified by M_i
  - A series of k classifiers is iteratively learned
  - The final M* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy

Boosting



x

C*

$c^*(x) = \text{argmax}_{cm} \sum w_t c_t(x)$

C^1 $c_1(x)$

C^2 $c_2(x)$

…

C^T $c_T(x)$

train

train

train

D,w^1

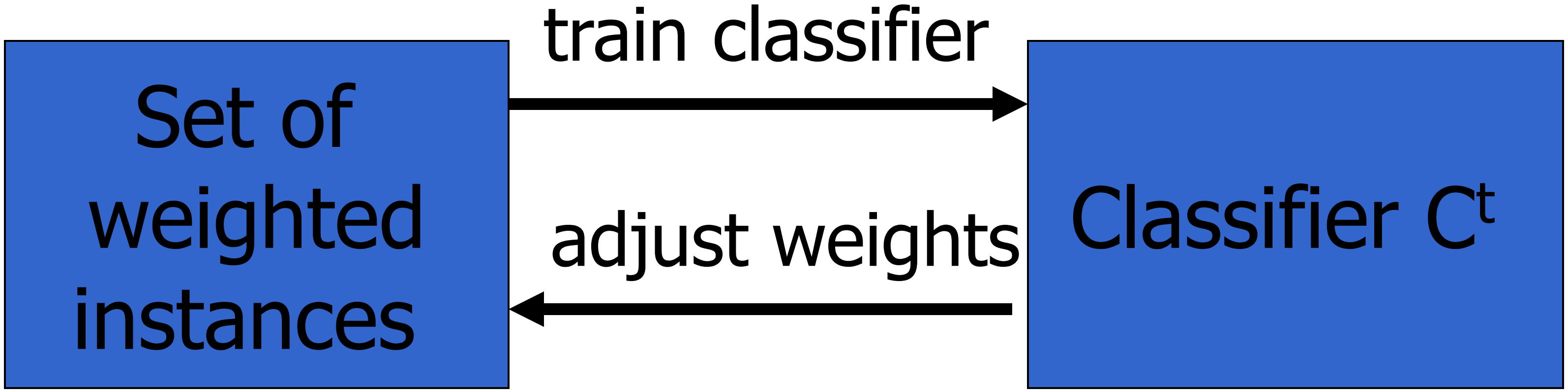
D,w^2

…

D,w^T

Boosting

- The boosting algorithm can be extended for the prediction of continuous values
- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data



train classifier

Set of weighted instances

Classifier C^t

adjust weights

Bagging vs. Boosting

- Model training:
  - Bagging: random sampling, independent classifiers
  - Boosting: subsequent classifier, M_{i+1}, pay more attention to the training tuples that were misclassified by M_i
- Model usage:
  - Bagging: equal weight
  - Boosting: different weight assigned

Ensemble Methods

- Text mining
- Video pattern recognition
- Audio pattern recognition

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Other classification methods
- Prediction
- Accuracy and error measures
- Summary

What Is Prediction?

- (Numerical) prediction is similar to classification
  - construct a model
  - use model to predict continuous or ordered value for a given input
- Prediction is different from classification
  - Classification refers to predict categorical class label
  - Prediction models continuous-valued functions
- Major method for prediction: regression
  - model the relationship between one or more *independent* or **predictor** variables and a *dependent* or **response** variable
- Regression analysis
  - Linear and multiple regression
  - Non-linear regression
  - Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees, logistic regression

Linear Regression

- Linear regression: a response variable y and a single predictor variable x

  $y = w_0 + w_1 x$

  where w_0 (intercept) and w_1 (slope) are regression coefficient

- Method of least squares: estimates the best-fitting straight line

$$w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})^2} \qquad w_0 = \bar{y} - w_1 \bar{x}$$

- Multiple linear regression: more than one predictor variable
  - Training data is of the form ($\mathbf{X}_1$, y_1), ($\mathbf{X}_2$, y_2),…, ($\mathbf{X}_{|D|}$, $y_{|D|}$)
  - Ex. For 2-D data, we may have: $y = w_0 + w_1 x_1 + w_2 x_2$
  - Solvable by extension of least square method or using SAS, S-Plus, R, Matlab
  - Many nonlinear functions can be transformed into the above

Nonlinear Regression

- A polynomial regression model can be transformed into linear regression model. For example,

  $$y = w_0 + w_1\ x + w_2\ x^2 + w_3\ x^3$$

  convertible to linear with new variables: $x_2 = x^2$, $x_3 = x^3$

  $$y = w_0 + w_1\ x + w_2\ x_2 + w_3\ x_3$$

- Some models are intractable nonlinear (e.g., sum of exponential terms)
  - possible to obtain least square estimates through extensive calculation on more complex formulae

Other Regression-Based Models

- Generalized linear model:
  - Foundation on which linear regression can be applied to modeling categorical response variables
  - Logistic regression: models the prob. of some event occurring as a linear function of a set of predictor variables

    $\text{Log}(p/1\text{-}p) = w_0 + w_1\, x_1 + w_2\, x_2 + \ldots + w_n\, x_n$, p is probability Y=1
  - Poisson regression: models the data that exhibit a Poisson distribution
- Log-linear models: (for categorical data)
  - Approximate discrete multidimensional prob. distributions
  - Also useful for data compression and smoothing

$$\log(y) = w_0 + w_1\, x + w_2\, x_2 + \ldots + w_n\, x_n$$

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Other classification methods
- Prediction
- Accuracy and error measures
- Summary

Classifier Accuracy Measures

- Accuracy of a classifier M, acc(M): percentage of test set tuples that are correctly classified by the model M
  - Given *m* classes, $CM_{i,j}$, an entry in a confusion matrix, indicates # of tuples in class *i* that are labeled by the classifier as class *j*
  - Accuracy = (t-pos + t-neg)/ (pos + neg)
  - Error rate (misclassification rate) of M = 1 – acc(M)

Predicted class

Actual class

	C_1	C_2	Total
C_1	True positive	False negative	pos
C_2	False positive	True negative	neg
Total	t-pos+f-pos	t-neg+f-neg	pos+neg

Classifier Accuracy Measures

- Alternative accuracy measures

  sensitivity = t-pos/pos /* true positive recognition rate */

  specificity = t-neg/neg /* true negative recognition rate */

  precision = t-pos/(t-pos + f-pos)

  accuracy = sensitivity * pos/(pos + neg) + specificity * neg/(pos + neg)

  = (t-pos + t-neg)/ (pos + neg)

Exercise

1. Please compute the sensitivity, specificity, precision and accuracy of the classifier.

classes	buy_computer = yes	buy_computer = no	total	recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
total	7366	2634	10000	95.42

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Other classification methods
- Prediction
- Accuracy and error measures
- Summary

Summary (I)

- Classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends.
- Effective and scalable methods have been developed for decision trees induction, Naive Bayesian classification, Backpropagation, k-nearest neighbor classifiers.

Summary (II)

- Linear, nonlinear, and generalized linear models of regression can be used for prediction. Many nonlinear problems can be converted to linear problems by performing transformations on the predictor variables.
- k-fold cross-validation is a recommended method for accuracy estimation.
- Bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models.
- No single method has been found to be superior over all others for all data sets.
- Issues such as accuracy, training time, robustness, interpretability, and scalability must be considered.