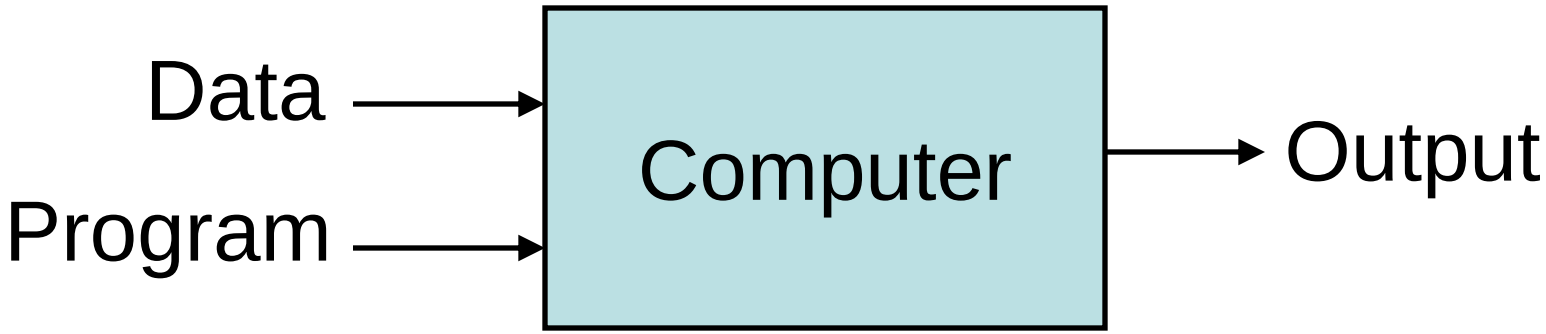
Machine Learning

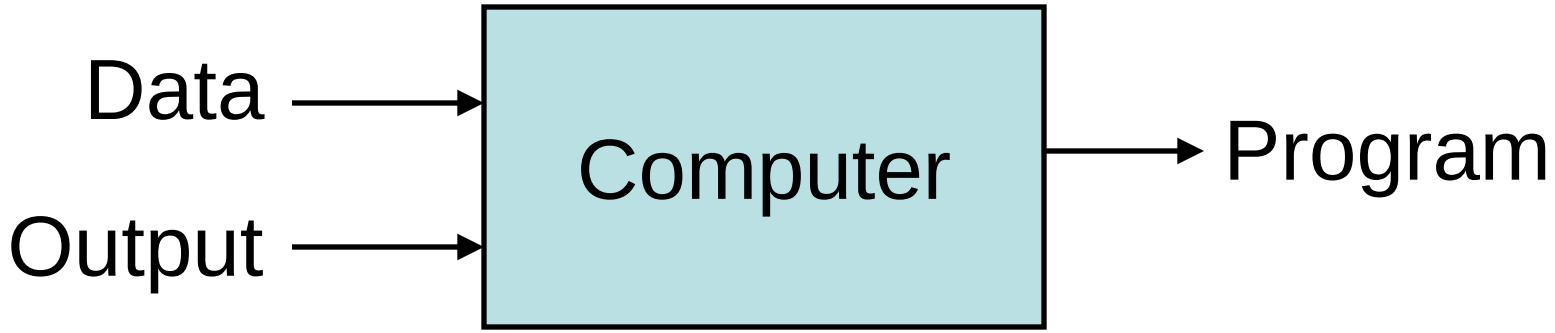
So, What Is Machine Learning?

- Automating automation
- Getting computers to program themselves
- Writing software is the bottleneck
- Let the data do the work instead!

Traditional Programming



Machine Learning



Sample Applications

- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks
- Debugging
- …[Your favorite area]

What is Machine Learning?

- Machine Learning
  - Study of algorithms that
  - improve their performance
  - at some task
  - with experience
- Optimize a performance criterion using example data or past experience.
- Role of Statistics: Inference from a sample
- Role of Computer science: Efficient algorithms to
  - Solve the optimization problem
  - Representing and evaluating the model for inference

Growth of Machine Learning

- Machine learning is preferred approach to
  - Speech recognition, Natural language processing
  - Computer vision
  - Medical outcomes analysis
  - Robot control
  - Computational biology
- This trend is accelerating
  - Improved machine learning algorithms
  - Improved data capture, networking, faster computers
  - Software too complex to write by hand
  - New sensors / IO devices
  - Demand for self-customization to user, environment
  - It turns out to be difficult to extract knowledge from human experts→*failure of expert systems in the 1980's.*

ML in a Nutshell

- Tens of thousands of machine learning algorithms
- Hundreds new every year
- Every machine learning algorithm has three components:
  - **Representation Model**
  - **Evaluation**
  - **Optimization**

Representation

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.

Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- Etc.

Optimization

- Combinatorial optimization
  - E.g.: Greedy search
- Convex optimization
  - E.g.: Gradient descent
- Constrained optimization
  - E.g.: Linear programming

Types of Learning

- **Association Analysis**
- **Supervised (inductive) learning**
  - Training data includes desired outputs
- **Unsupervised learning**
  - Training data does not include desired outputs
- **Semi-supervised learning**
  - Training data includes a few desired outputs
- **Reinforcement learning**
  - Rewards from sequence of actions

Supervised Learning

- **Given** examples of a function *(X, F(X))*
- **Predict** function *F(X)* for new examples *X*
  - Discrete *F(X)*: Classification
  - Continuous *F(X)*: Regression
  - *F(X)* = Probability(*X*): Probability estimation

Supervised Learning: Uses

Example: decision trees tools that create rules

- Prediction of future cases: Use the rule to predict the output for future inputs
- Knowledge extraction: The rule is easy to understand
- Compression: The rule is simpler than the data it explains
- Outlier detection: Exceptions that are not covered by the rule, e.g., fraud

Unsupervised Learning

- Learning “what normally happens”
- No output
- Clustering: Grouping similar instances
- Other applications: Summarization, Association Analysis
- Example applications
  - Customer segmentation in CRM
  - Image compression: Color quantization
  - Bioinformatics: Learning motifs

Reinforcement Learning

- Topics:
  - Policies: what actions should an agent take in a particular situation
  - Utility estimation: how good is a state (→used by policy)
- No supervised output but delayed reward
- Credit assignment problem (what was responsible for the outcome)
- Applications:
  - Game playing
  - Robot in a maze
  - Multiple agents, partial observability, ...

Supervised vs. unsupervised Learning

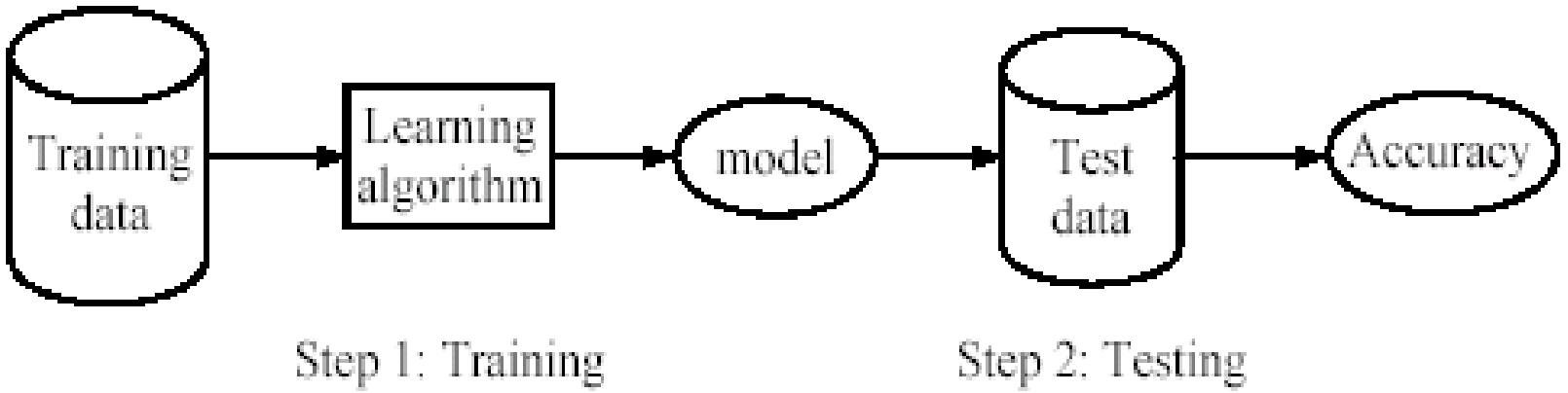
- Supervised learning: classification is seen as supervised learning from examples.
  - Supervision: The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a “teacher” gives the classes (supervision).
  - Test data are classified into these classes too.
- Unsupervised learning (clustering)
  - Class labels of the data are unknown
  - Given a set of data, the task is to establish the existence of classes or clusters in the data

Supervised learning process: two steps

Learning (training): Learn a model using the training data

Testing: Test the model using unseen test data to assess the model accuracy

$$Accuracy = \frac{\text{Number of correct classifica tions}}{\text{Total number of test cases}},$$



Supervised Learning

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, observations, etc. with the aim of establishing the existence of classes or clusters in the data

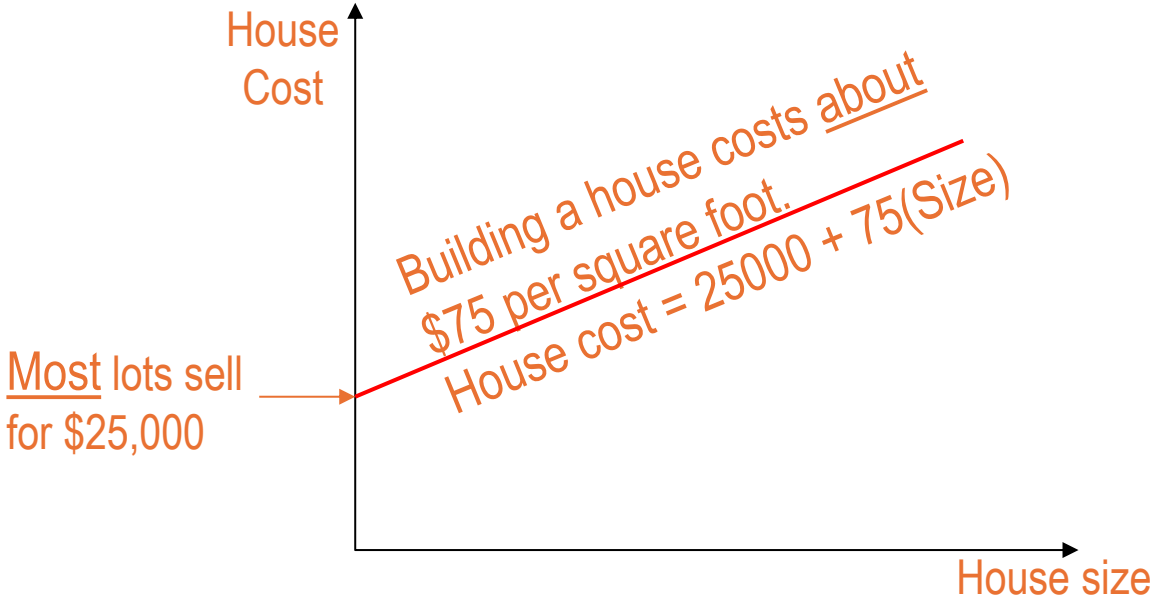
Prediction: Classification vs. Numeric Prediction

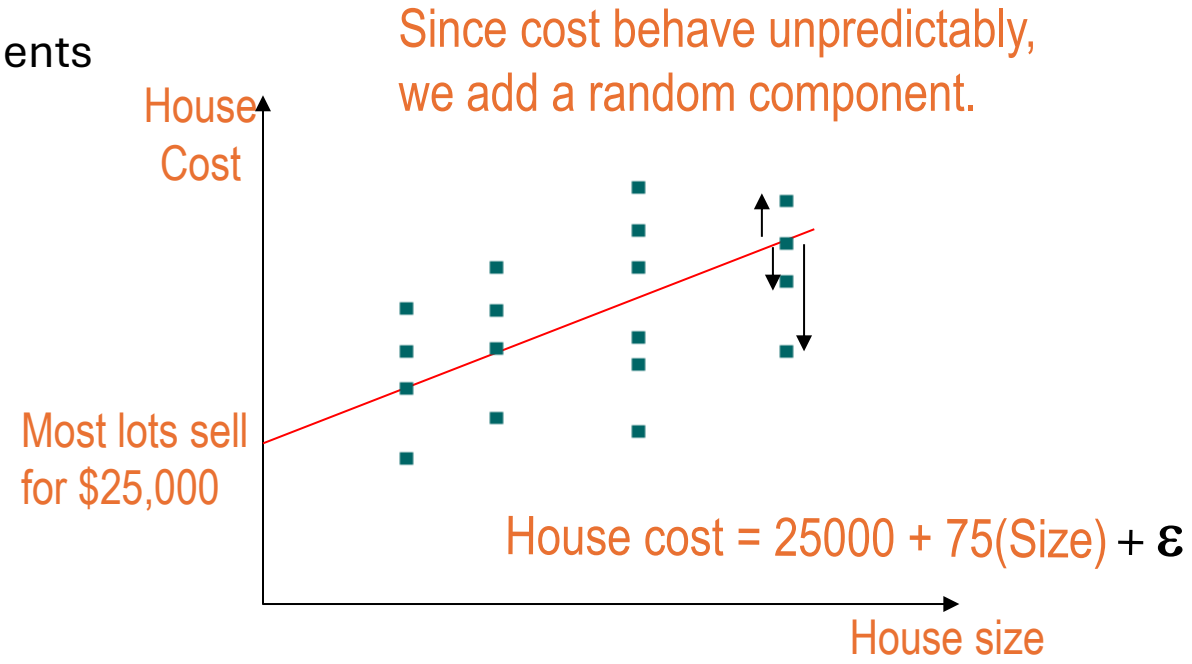
- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
  - E.g. Credit/loan approval, Medical diagnosis: if a tumor is cancerous or benign, Fraud detection: if a transaction is fraudulent or not, etc.
  - **Algorithms:** Decision trees, support vector machines (SVMs), Naive Bayes.
- Numeric Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
  - E.g. Predicting house prices, Forecasting stock market values, Estimating temperature, etc.
  - **Algorithms:** Linear regression, polynomial regression, support vector regression (SVR).

Simple linear regression

It is statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables:

- One variable, denoted *x*, is regarded as the **predictor**, **explanatory**, or **independent** variable.
- The other variable, denoted *y*, is regarded as the **response**, **outcome**, or **dependent** variable.
- We will examine the relationship between quantitative variables x and y via a mathematical equation.
- The model has a deterministic and a statistical components





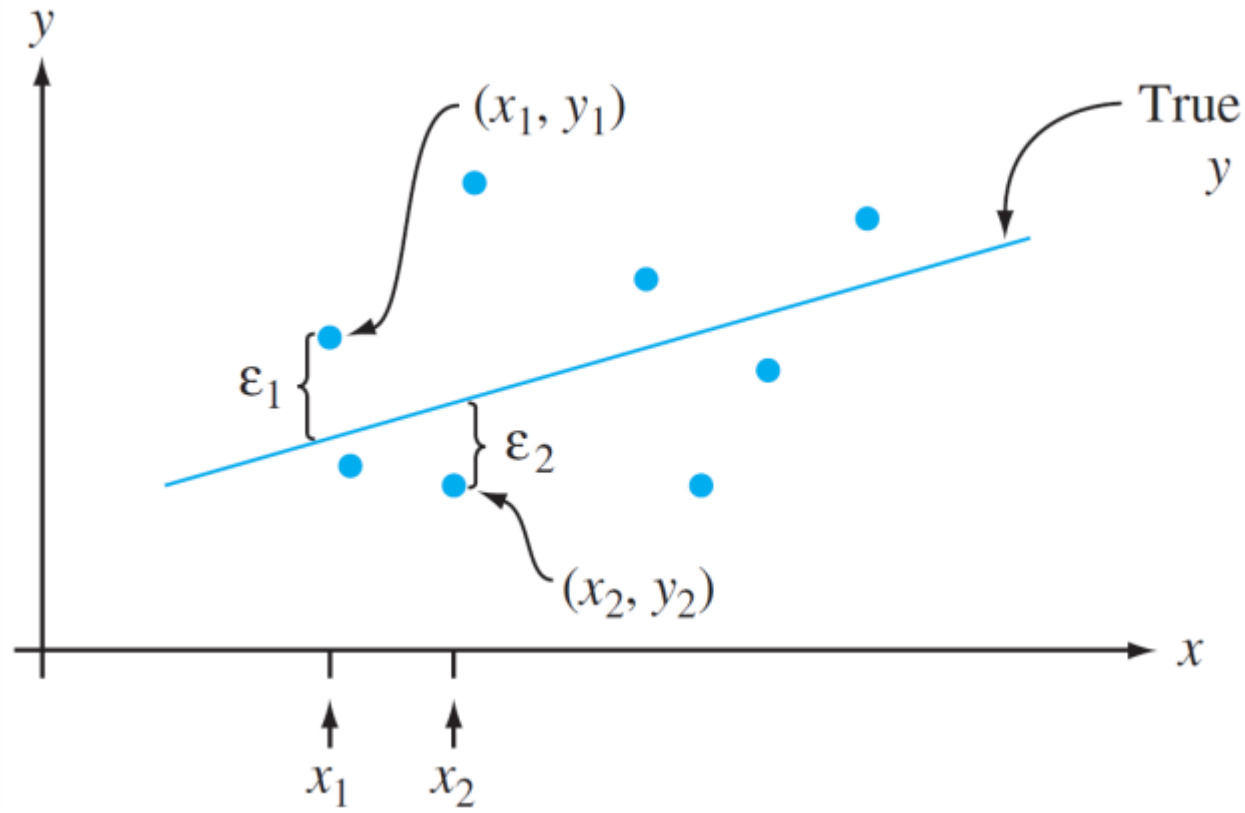
Simple linear regression

- The simplest deterministic mathematical relationship between two variables x and y is a linear relationship: $y = \beta_0 + \beta_1 x$. (True regression line)
- The objective is to develop an equivalent linear probabilistic model.
- If the two (random) variables are probabilistically related, then for a fixed value of x, there is uncertainty in the value of the second variable.
- So, we assume $y = \beta_0 + \beta_1 x + \varepsilon$, where ε is a random variable.

$$b_1 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \qquad b_0 = \bar{y} - b_1\bar{x}$$

Simple linear regression

- The points (x1, y1), …, (xn, yn) resulting from n independent observations will then be scattered about the true regression line:



$$y = \beta_0 + \beta_1 x + \varepsilon$$

Simple linear regression

Estimating Model parameters:

- The values of $\beta_{0,}$ β_1 and ε will almost never be known to an investigator.
- Instead, sample data consists of **n** observed pairs $(x_1, y_1), \ldots, (x_n, y_n)$, from which the model parameters and the true regression line itself can be estimated.
- Where $Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$ for i = 1, 2, … , n and the ***n*** *deviations* $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ are independent r.v.' s.
- Aim is to find the **Best Fit Line:** the sum of the squared vertical distances (deviations) from the observed points to that line is as small as it can be.

Simple linear regression

The sum of squared vertical deviations from the points $(x_1, y_1), \ldots, (x_n, y_n)$, to the line is then

$$f(b_0, b_1) = \sum_{i=1}^{n} [y_i - (b_0 + b_1 x_i)]^2$$

The point estimates of β_0 and β_1, denoted by b_1 and b_0, are called the least squares estimates – they are those values that minimize using partial derivatives.

$$b_1 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \qquad b_0 = \bar{y} - b_1 \bar{x}$$

The predicted values are obtained using:

$$\hat{y} = b_0 + b_1 x$$

$$b_1 = \frac{SS_{xy}}{SS_{xx}}$$

$$b_0 = \bar{y} - b_1 \bar{x}$$

$$SS_{xy} = \sum x_i y_i - \frac{\left(\sum x_i\right)\left(\sum y_i\right)}{n}$$

$$SS_{xx} = \sum x_i^2 - \frac{\left(\sum x_i\right)^2}{n} = (n-1)s_x^2$$

Simple linear regression

We interpret the fitted value as the value of *y* that we would predict or expect when using the estimated regression line with $x = x_i$; thus $\hat{y}_i$ is the ***estimated true mean*** for that population when $x = x_i$ (based on the data).

The residual $y_i - \hat{y}_i$ is a positive number if the point lies above the line and a negative number if it lies below the line. $(x_i, \hat{y}_i)$

The residual can be thought of as a measure of deviation and we can summarize the notation in the following way:

$$Y_i - \hat{Y}_i = \hat{\epsilon}_i$$

Simple linear regression

Suppose we have the following data on filtration rate (x) versus moisture content (y):

x	125.3	98.2	201.4	147.3	145.9	124.7	112.2	120.2	161.2	178.9
y	77.9	76.8	81.5	79.8	78.2	78.3	77.5	77.0	80.1	80.2
x	159.5	145.8	75.1	151.4	144.2	125.0	198.8	132.5	159.6	110.7
y	79.9	79.0	76.7	78.2	79.5	78.1	81.5	77.0	79.0	78.6

Relevant summary quantities (*summary statistics*) are

$\Sigma x_i = 2817.9$, $\Sigma y_i = 1574.8$, $\Sigma x^2_i = 415{,}949.85$,

$\Sigma x_i\, y_i = 222{,}657.88$, and $\Sigma y^2_i = 124{,}039.58$,

From $S_{xx} = 18{,}921.8295$, $S_{xy} = 776.434$.

Calculation of residuals?

Simple linear regression

x	y	x^2	xy
3	8	9	24
9	6	81	54
5	4	25	20
3	2	9	6
∑x = 20	**∑y = 20**	**$\sum x^2 = 124$**	**∑xy = 104**

$$b_1 = \frac{SS_{xy}}{SS_{xx}}$$

$$b_0 = \bar{y} - b_1\bar{x}$$

$$SS_{xy} = \sum x_i y_i - \frac{\left(\sum x_i\right)\left(\sum y_i\right)}{n}$$

$$SS_{xx} = \sum x_i^2 - \frac{\left(\sum x_i\right)^2}{n} = (n-1)s_x^2$$

Using formula,

b1 = {4(104) – 20*20} / {4*(124) – 20^2} = 16/96 = 0.166*

b0 = 20/4 - 0.166(20/4) = 4.17*

So, linear regression equation is, y= b0 + b1x => y = 4.17 + 0.166x

Simple linear regression

Linear regression, while a powerful tool, has certain limitations that should be considered:

• **Linearity:** Assumes a linear relationship between the dependent and independent variables. If the relationship is non-linear, the model may not accurately capture the underlying pattern.

• **Independence:** Assumes that the errors are independent of each other. If there is autocorrelation in the errors, the model's estimates may be biased and inefficient.

• **Homoscedasticity:** Assumes that the variance of the errors is constant across all levels of the independent variable. If the variance is not constant (heteroscedasticity), the model's estimates may be inefficient.

• **Normality:** Assumes that the errors are normally distributed. If the errors are not normally distributed, the model's inferences may be invalid.

• **Sensitivity to Outliers:** Linear regression can be sensitive to outliers, which can have a significant impact on the model's estimates. Outliers can distort the relationship between the variables and lead to biased results.

• **Limited Flexibility:** Linear regression can only model linear relationships. If the relationship between the variables is complex or non-linear, linear regression may not be able to adequately capture the pattern.

Regression Metrics

Some common regression metrics are

- **Mean Absolute Error (MAE):** $MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i|$
- **Mean Squared Error (MSE):** $MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2$
- **Root Mean Squared Error (RMSE):** $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}$
- **R-squared (R^2) Score:** $R^2 = 1 - (SSR / SST)$

$$\text{r2\_score} = 1 - \frac{\text{total\_error\_model}}{\text{total\_error\_baseline}} = 1 - \frac{\sum_{i=1}^{N}(\text{predicted}_i - \text{actual}_i)^2}{\sum_{i=1}^{N}(\text{average\_value} - \text{actual}_i)^2}$$

where,

- x_i represents the actual or observed value for the i-th data point.
- y_i represents the predicted value for the i-th data point.
- SSR (Sum of Squared Residuals) and SST (Total Sum of Squares).

Regression Metrics

Q. A real estate company is trying to predict the selling price of houses based on their size (in square feet). They trained a regression model and obtained the following predicted prices and actual selling prices for a sample of five houses:

Calculate the MAE, MSE, RMSE, R2 Score.

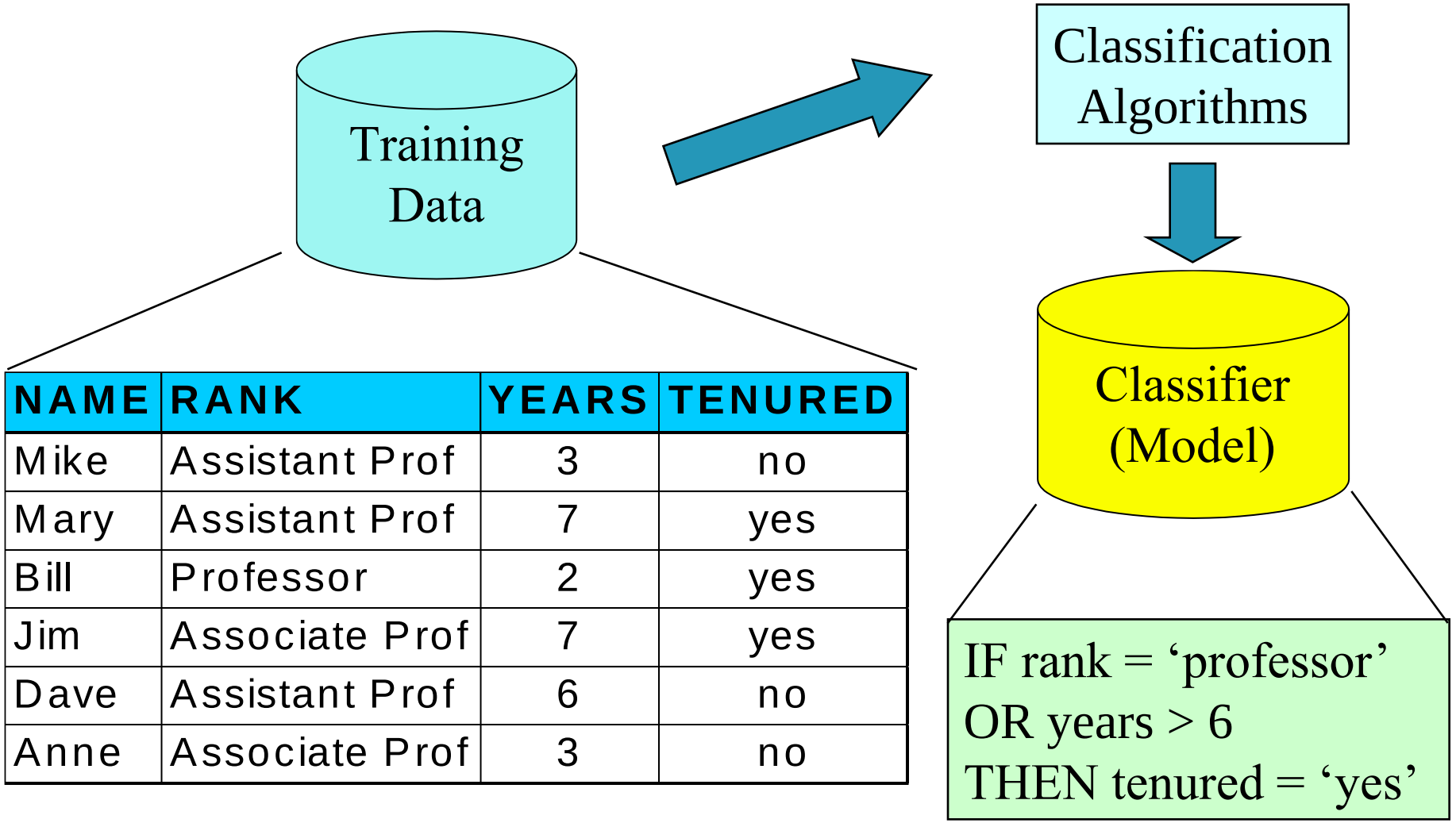
House	Actual Price (in $1000)	Predicted Price (in $1000)
1	300	280
2	350	360
3	420	410
4	280	310
5	500	480

Supervised Learning

Classification—A Two-Step Process

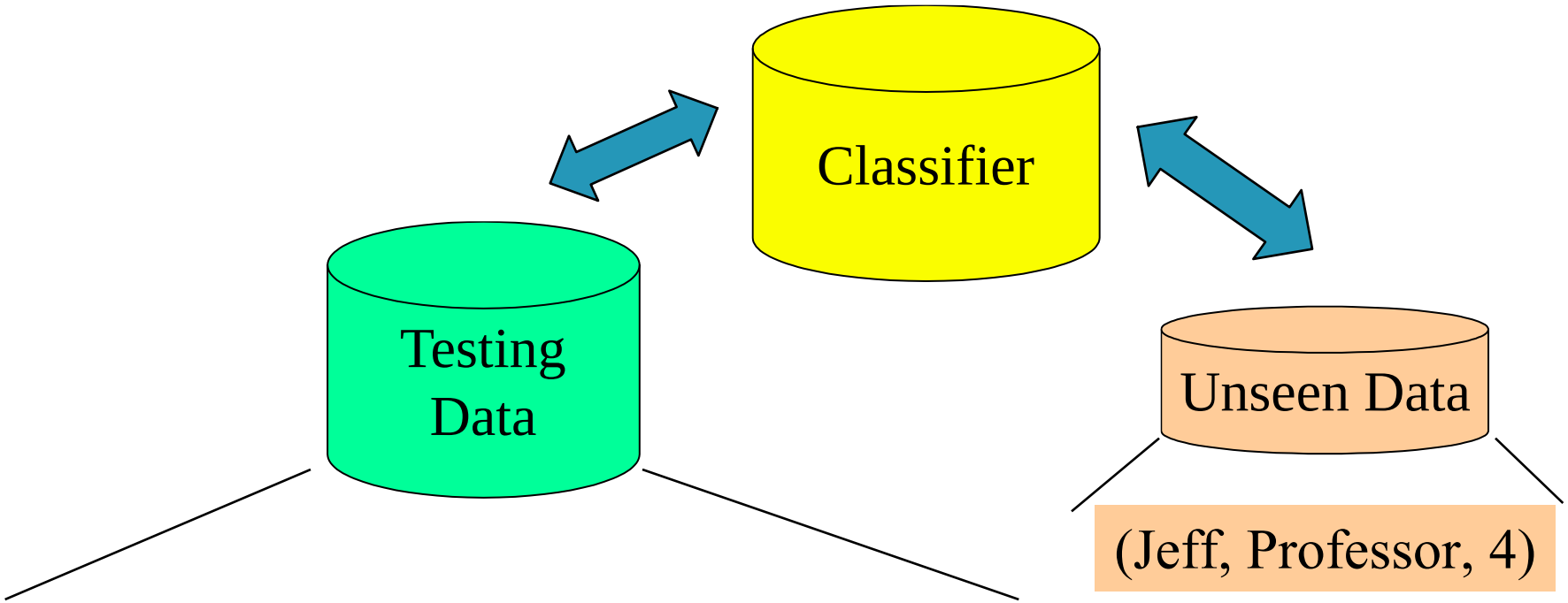
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- 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 overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If *the test set* is used to select models, it is called validation (test) set

Process (1): Model Construction



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

Process (2): Using the Model in Prediction



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

Decision Tree

- A decision tree is a tree-like structure where each internal node tests on attribute, each branch corresponds to attribute value and each leaf node represents the final decision or prediction.
- The decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.

How is Decision Tree formed?

- The process of forming a decision tree involves recursively partitioning the data based on the values of different attributes.
- The algorithm selects the best attribute to split the data at each internal node, based on certain criteria such as information gain or Gini impurity.
- This splitting process continues until a stopping criterion is met, such as reaching a maximum depth or having a minimum number of instances in a leaf node.

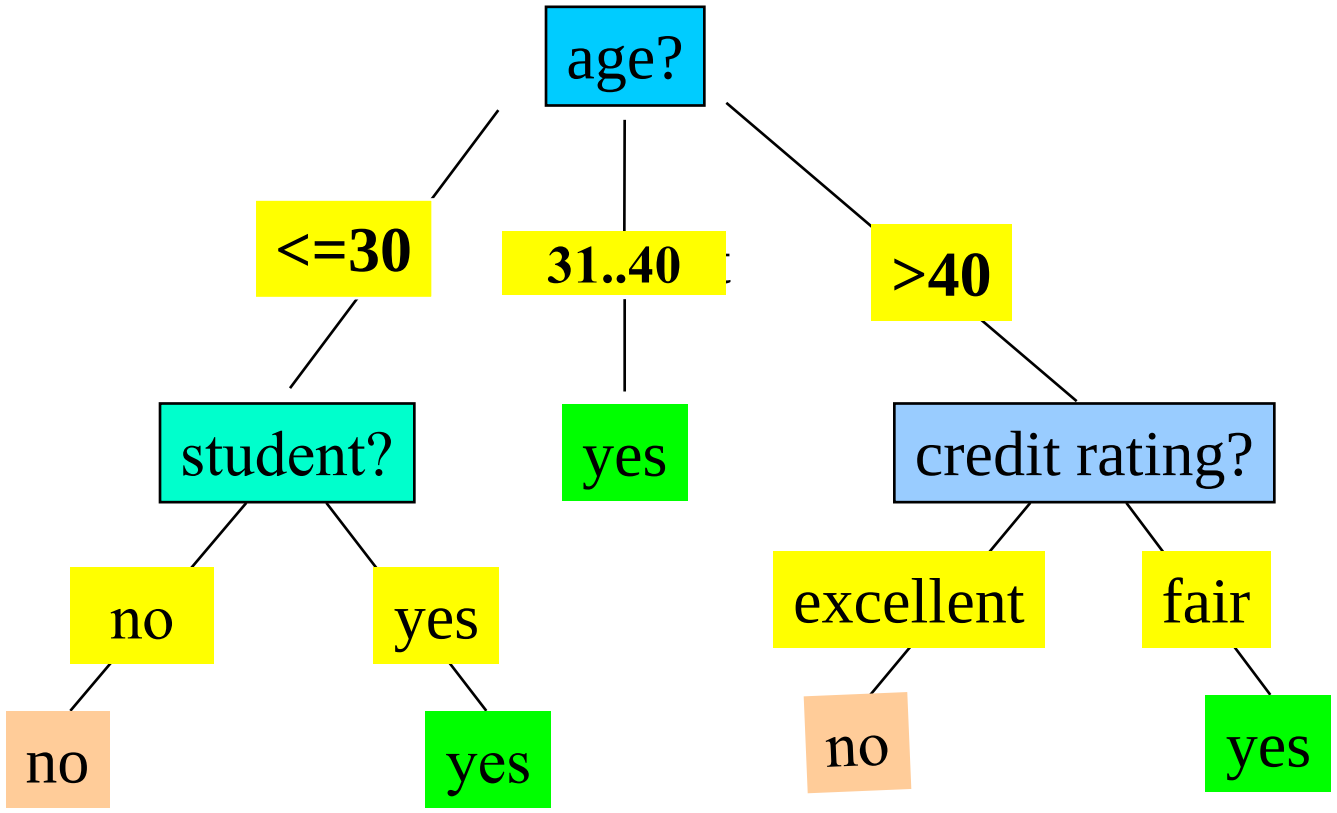
Decision Tree

There are specialized terms associated with decision trees that denote various components and facets of the tree structure and decision-making procedure. :

- **Root Node:** A decision tree's root node, which represents the original choice or feature from which the tree branches, is the highest node.
- **Internal Nodes (Decision Nodes):** Nodes in the tree whose choices are determined by the values of particular attributes. There are branches on these nodes that go to other nodes.
- **Leaf Nodes (Terminal Nodes):** The branches' termini, when choices or forecasts are decided upon. There are no more branches on leaf nodes.
- **Branches (Edges):** Links between nodes that show how decisions are made in response to particular circumstances.
- **Splitting:** The process of dividing a node into two or more sub-nodes based on a decision criterion. It involves selecting a feature and a threshold to create subsets of data.
- **Decision Criterion:** The rule or condition used to determine how the data should be split at a decision node. It involves comparing feature values against a threshold.
- **Pruning:** The process of removing branches or nodes from a decision tree to improve its generalization and prevent overfitting.

Decision Tree Induction: An Example

- Training data set: Buys_computer
- The data set follows an example of Quinlan's ID3 (Playing Tennis)
- Resulting tree:



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

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 based on 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

Attribute Selection Measures

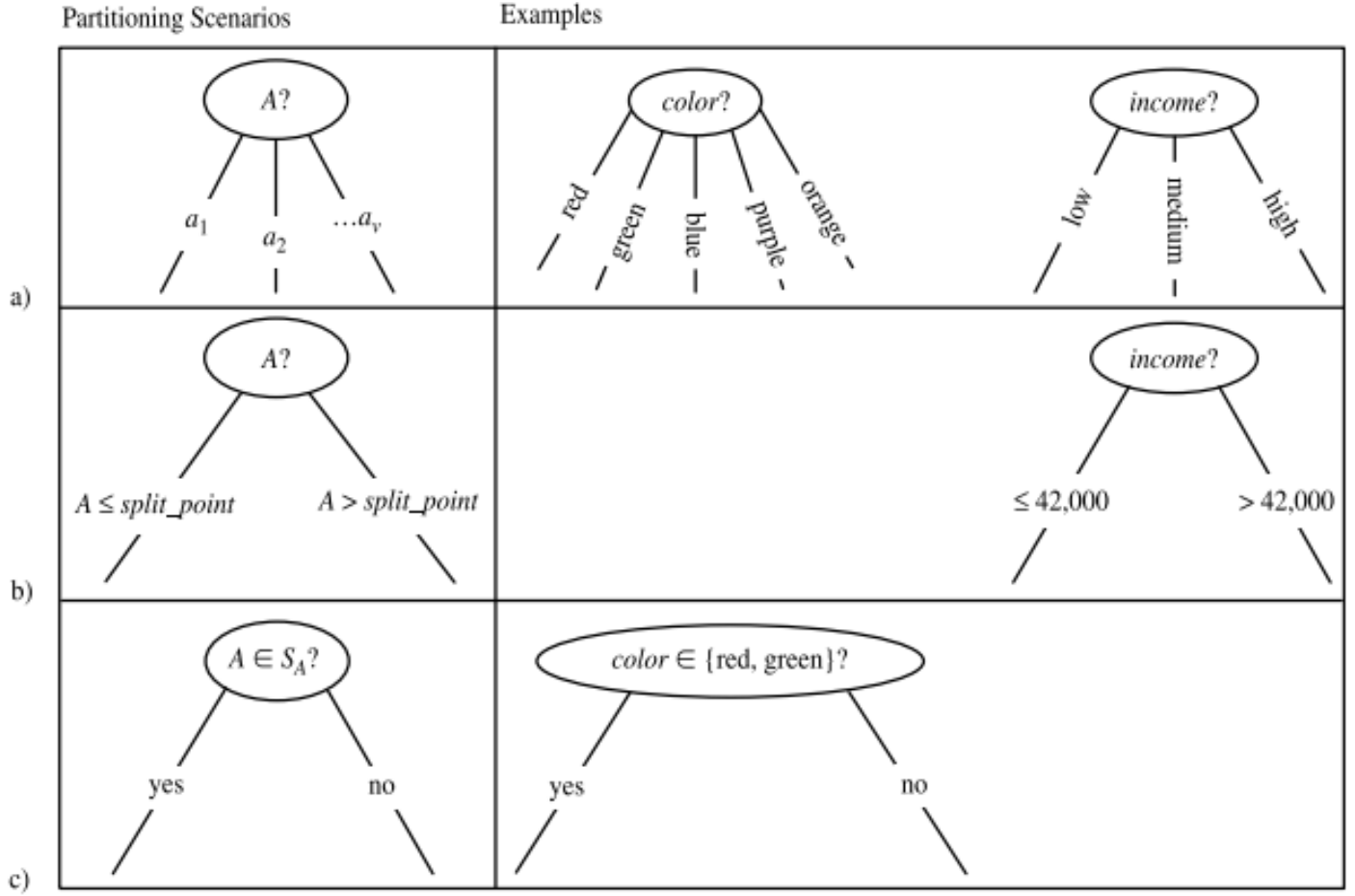
An attribute selection measure is a heuristic for selecting the splitting criterion that "best" separates a given data partition, D, of class-labeled training tuples into individual classes.

Three popular attribute selection measures are :

- information gain,
- gain ratio, and
- gini index.

Let,

- D be the data partition, be a training set of class-labeled tuples.
- Suppose the class label attribute has m distinct values defining m distinct classes, C_i (for i = 1,..., m).
- Let $C_{i,D}$ be the set of tuples of class C_i in D.
- Let |D| and | $C_{i,D}$ | denote the number of tuples in D and $C_{i,D}$, respectively.



Three possibilities for partitioning tuples based on the splitting criterion, shown with examples. Let A be the splitting attribute. (a) If A is discrete-valued, then one branch is grown for each known value of A. (b) If A is continuous-valued, then two branches are grown, corresponding to $A \leq split\_point$ and $A > split\_point$. (c) If A is discrete-valued and a binary tree must be produced, then the test is of the form $A \in S_A$, where S_A is the splitting subset for A.

Attribute Selection Measures

1. Information Gain:
ID3 uses information gain as its attribute selection measure. When we use a node in a decision tree to partition the training instances into smaller subsets the information/entropy changes.
Information gain is a measure of this change in entropy.

- Expected information/entropy needed to classify a tuple in D is given by

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

Where, where p_i is the probability that an arbitrary tuple in D belongs to class C_i and is estimated by $|C_i,D|/|D|$.

- Now, suppose the tuples in D are partitioned on some attribute A having *v* distinct values, $\{a_1, a_2,..., a_v\}$.

D is thus split into *v* partitions $\{D_1, D_2,..., D_v\}$, where D_j contains those tuples in D that have outcome a_j of A.

- We then calculate How much more information would we still need (after the partitioning) in order to arrive at an exact classification? This amount is measured by

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

- Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on A). That is,
- The attribute A with the highest information gain, (Gain(A)), is chosen as the splitting attribute at node N.

$$Gain(A) = Info(D) - Info_A(D).$$

Attribute Selection Measures

Class-labeled training tuples from the *AllElectronics* customer database.

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

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits}.$$

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

$$\begin{aligned} Info_{age}(D) &= \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) \\ &+ \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right) \\ &+ \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) \\ &= 0.694 \text{ bits}. \end{aligned}$$

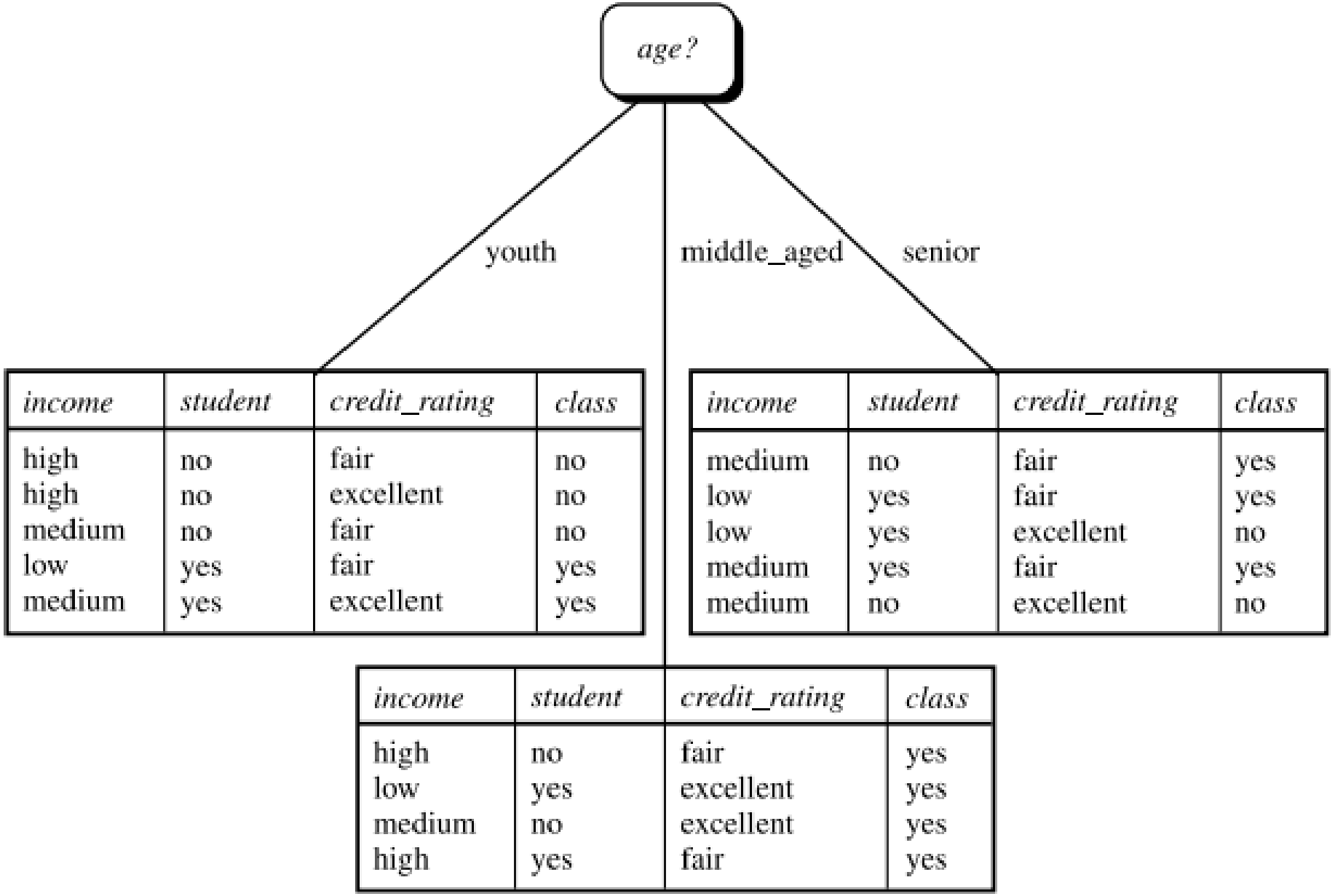
$$Gain(A) = Info(D) - Info_A(D).$$

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits}.$$

Gain(income) = 0.029 bits, Gain(student) = 0.151 bits,
Gain(credit rating) = 0.048 bits.

Since, **age** has the highest information gain among the attributes, it is selected as the splitting attribute.

Attribute Selection Measures



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

Attribute Selection Measures

2. Gain Ratio: The information gain measure is biased toward tests with many outcomes. That is, it prefers to select attributes having a large number of values.
C4.5, a successor of ID3, uses an extension to information gain known as gain ratio, which attempts to overcome this bias. It applies a kind of normalization to information gain using a "split information" value defined analogously with Info(D) as

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

The gain ratio is defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}.$$

Attribute Selection Measures

3. Gini Index

- Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.
- It means an attribute with a lower Gini index should be preferred.
- The Gini index is used in CART algorithm.
- The Gini value or D is calculated by:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

Where, where p_i is the probability that an arbitrary tuple in D belongs to class C_i and is estimated by $|C_i,D|/|D|$.

- The Gini index considers a binary split for each attribute. To determine the best binary split on A, we examine all of the possible subsets that can be formed using known values of A.
- If A has v possible values, then there are 2^v possible subsets.
    - For example, if income has three possible values, namely {low, medium, high}, then the possible subsets are {low, medium, high}, {low, medium}, {low, high}, {medium, high}, {low}, {medium}, {high}, and {}.
    - We exclude the power set, {low, medium, high}, and the empty set from consideration since, conceptually, they do not represent a split.
    - Therefore, there are $2^v - 2$ possible ways to form two partitions of the data, D, based on a binary split on A.

Attribute Selection Measures

3. Gini Index

The Gini value or D is calculated by:

Where, where p_i is the probability that an arbitrary tuple in D belongs to class C_i and is estimated by $|C_i,D|/|D|$.

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

- The Gini index considers a binary split for each attribute. To determine the best binary split on A, we examine all of the possible subsets that can be formed using known values of A.
- When considering a binary split, we compute a weighted sum of the impurity of each resulting partition. For example, if a binary split on A partitions D into D1 and D2, the gini index of D given that partitioning is

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2).$$

The reduction in impurity that would be incurred by a binary split on a discrete- or continuous-valued attribute A is $\Delta Gini(A) = Gini(D) - Gini_A(D)$.

The attribute that maximizes the reduction in impurity (or, equivalently, has the minimum Gini index) is selected as the splitting attribute.

Attribute Selection Measures

Example of Gini index

Let D be the training data that has 9 tuples belonging to the class buys_computer = yes and the remaining 5 tuples belong to the class buys_computer = no.

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

Let's start with the attribute income, and its subset {low, medium}. The Gini index value computed based on this partitioning is

$$\begin{aligned} & Gini_{income \in \{low,medium\}}(D) \\ &= \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2) \\ &= \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right) \\ &= 0.450 \\ &= Gini_{income \in \{high\}}(D). \end{aligned}$$

Similarly, the Gini index values for splits on the remaining subsets are: 0.315 (for the subsets {low, high} and {medium}) and 0.300 (for the subsets {medium, high} and {low}). Therefore, the best binary split for attribute income is on {medium, high} (or {low}) because it minimizes the gini index.

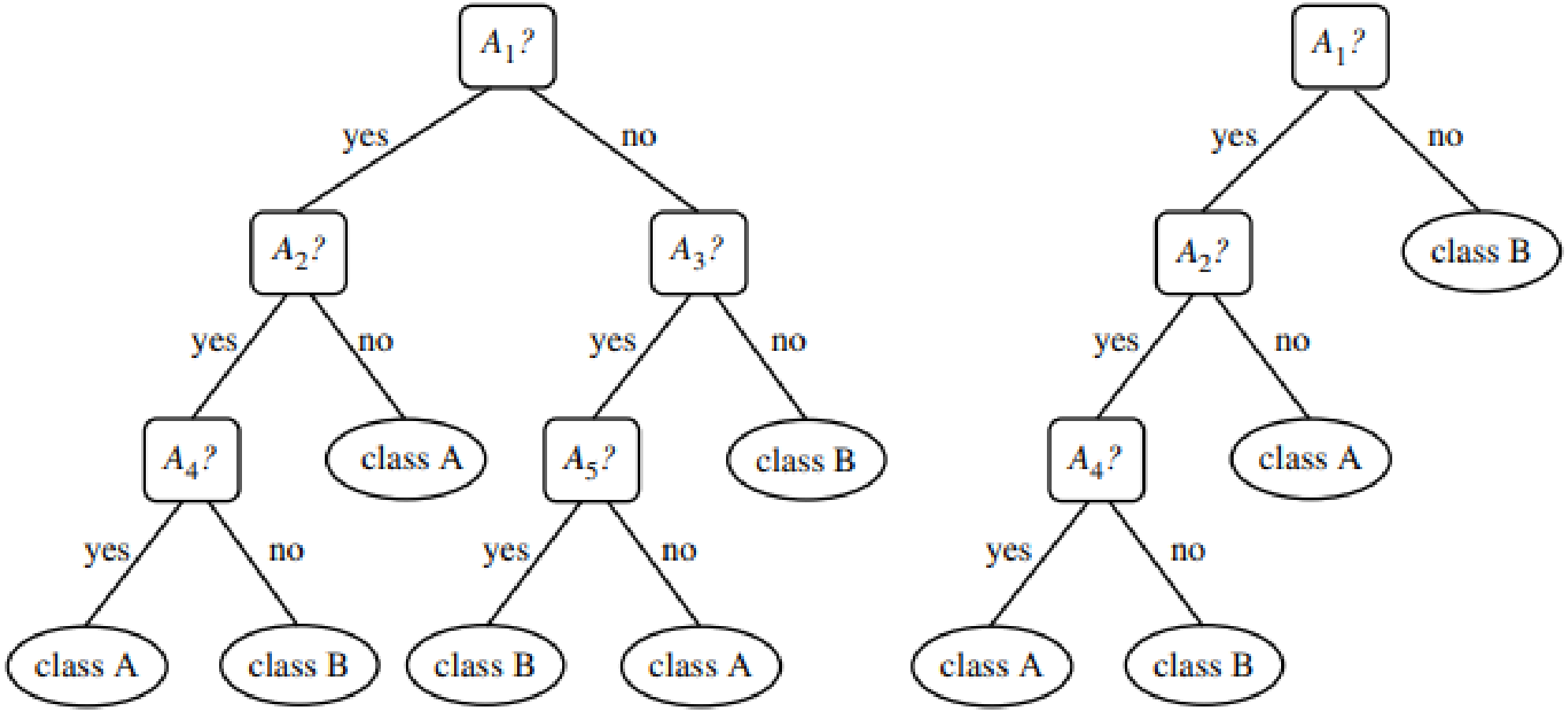
Evaluating the attribute, we obtain {youth, senior} (or {middle aged}) as the best split for age with a Gini index of 0.375; the attributes {student} and {credit rating} are both binary, with Gini index values of 0.367 and 0.429, respectively.

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”

Overfitting and Tree Pruning

- Example of pruning



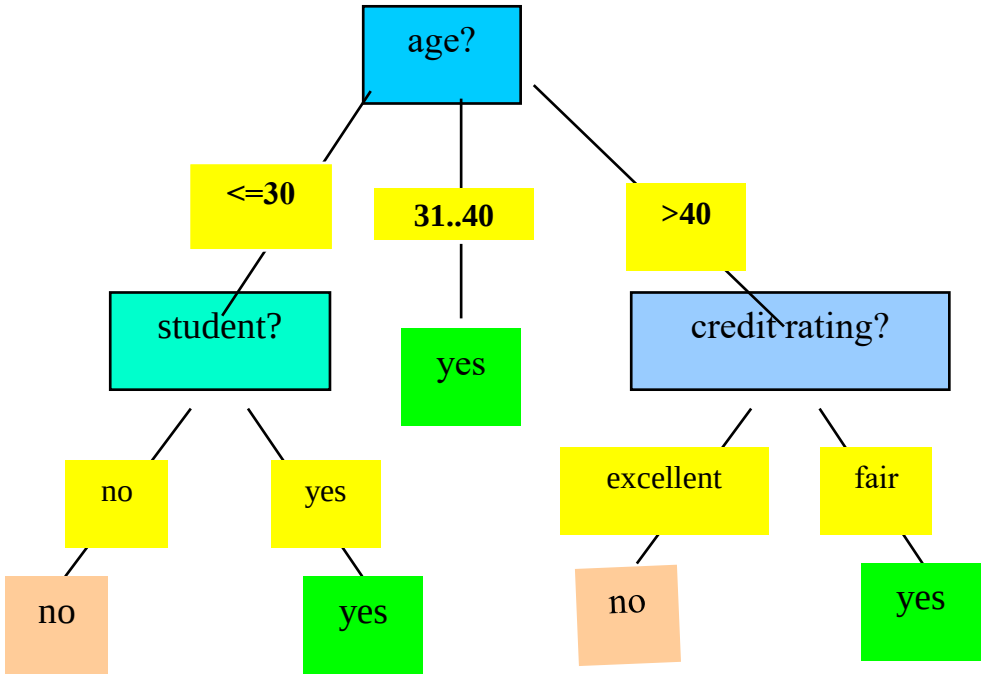
Practice question

decision-trees-for-classification

Day	Outlook	Temperature	Humidity	Wind	Play Golf
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

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



- Example: Rule extraction from our *buys_computer* decision-tree

IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*
IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*
IF *age* = mid-age THEN *buys_computer* = *yes*
IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *no*
IF *age* = old AND *credit_rating* = *fair* THEN *buys_computer* = *yes*

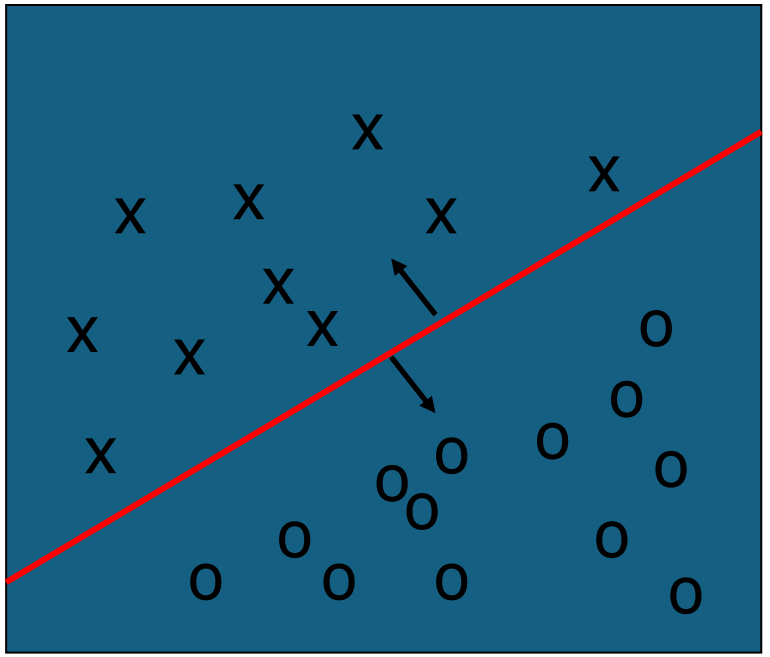
Supervised Learning

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
- 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 overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If *the test set* is used to select models, it is called validation (test) set

Classification: A Mathematical Mapping

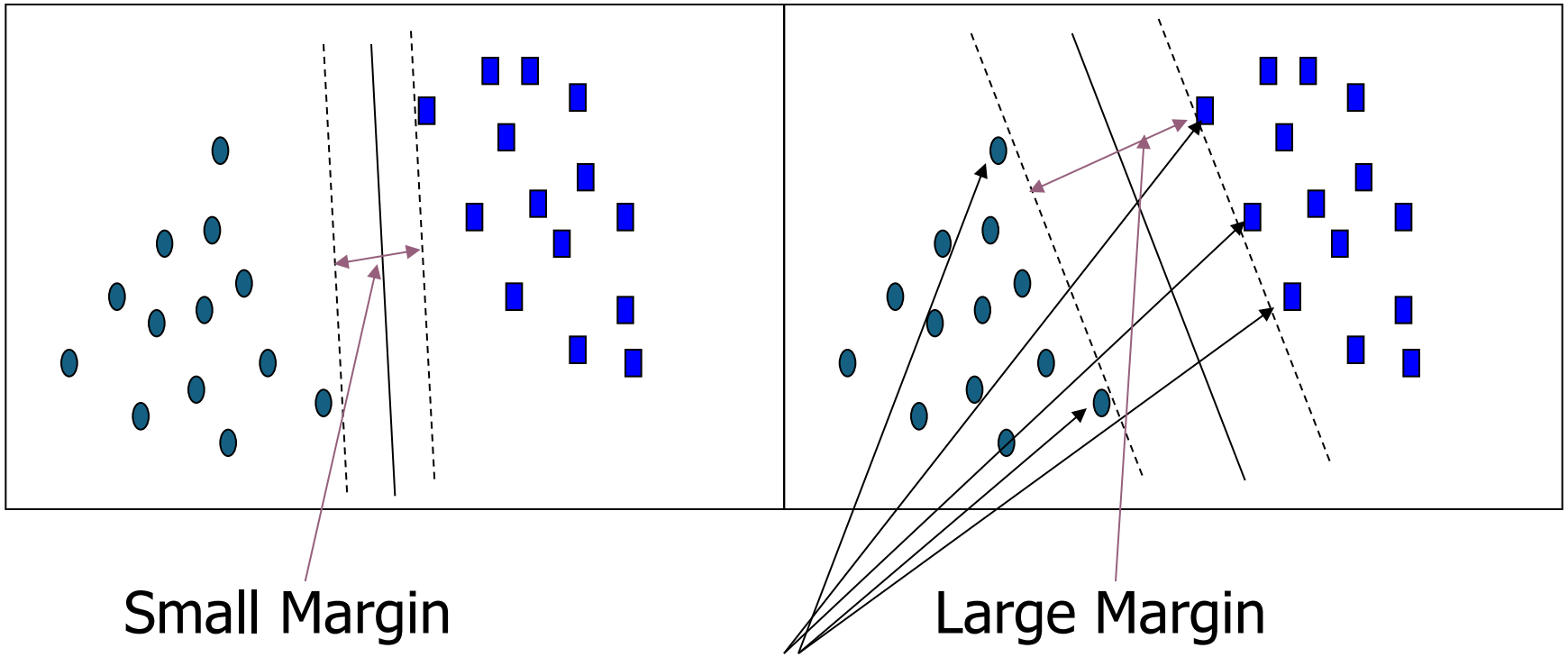
- Classification: predicts categorical class labels
  - E.g., Personal homepage classification
    - $x_i = (x_1, x_2, x_3, \ldots)$, $y_i = +1$ or -1
    - x_1 : # of word "homepage"
    - x_2 : # of word "welcome"
- Mathematically, $x \in X = \Re^n$, $y \in Y = \{+1, -1\}$,
  - We want to derive a function f: X → Y
- Linear Classification
  - Binary Classification problem
  - Data above the red line belongs to class 'x'
  - Data below red line belongs to class 'o'
  - Examples: SVM, Perceptron, Probabilistic Classifiers



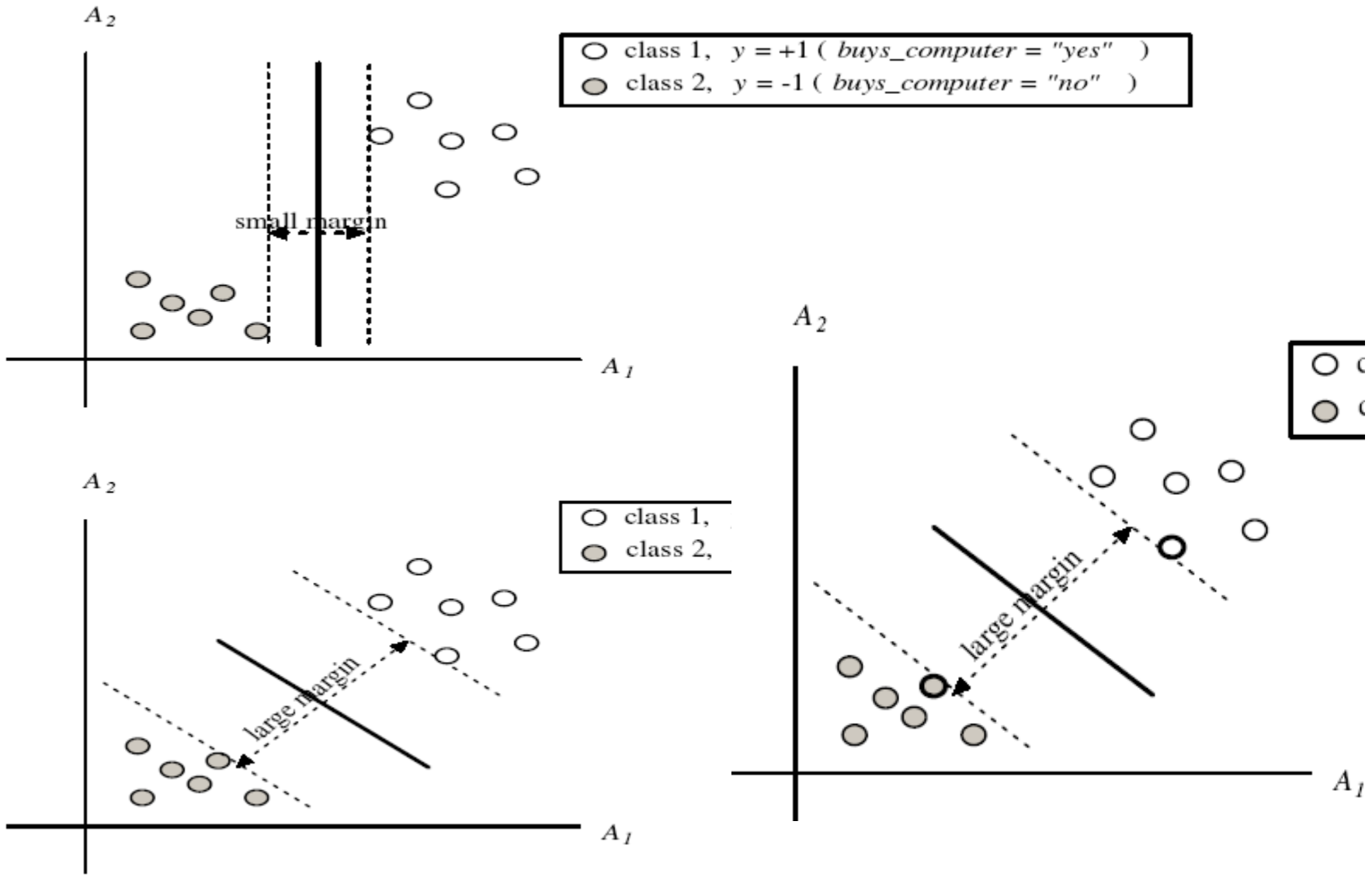
SVM—Support Vector Machines

- A relatively new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension.
- With the new dimension, it searches for the linear optimal separating **hyperplane** (i.e., "decision boundary").
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane.
- SVM finds this hyperplane using **support vectors** ("essential" training tuples) and **margins** (defined by the support vectors).
- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- Used for: classification and numeric prediction
- Applications: handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

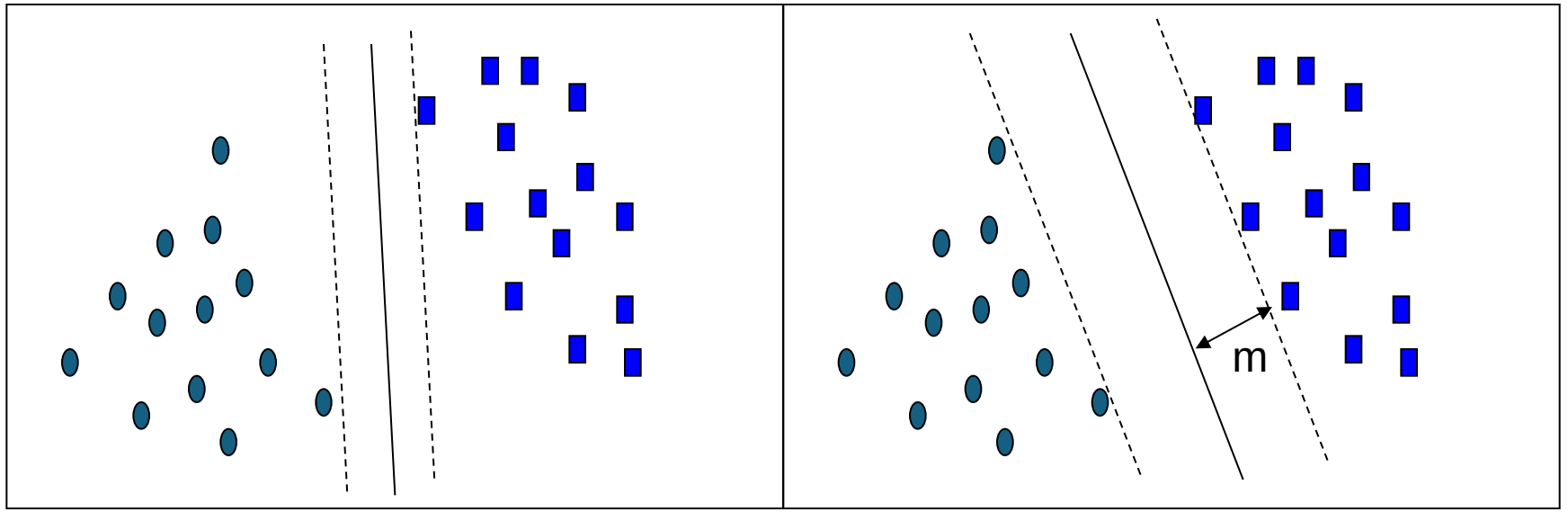
SVM—General Philosophy



SVM—Margins and Support Vectors



SVM—When Data Is Linearly Separable



Let data D be ($\mathbf{X}_1$, y_1), …, ($\mathbf{X}_{|D|}$, $y_{|D|}$), where $\mathbf{X}_i$ is the set of training tuples associated with the class labels y_i

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)

SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)

SVM—Linearly Separable

- A separating hyperplane can be written as

  $\mathbf{W} \bullet \mathbf{X} + b = 0$

  where $\mathbf{W}=\{w_1, w_2, \ldots, w_n\}$ is a weight vector and b a scalar (bias)
- For 2-D it can be written as

  $w_0 + w_1 x_1 + w_2 x_2 = 0$
- The hyperplane defining the sides of the margin:

  $H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

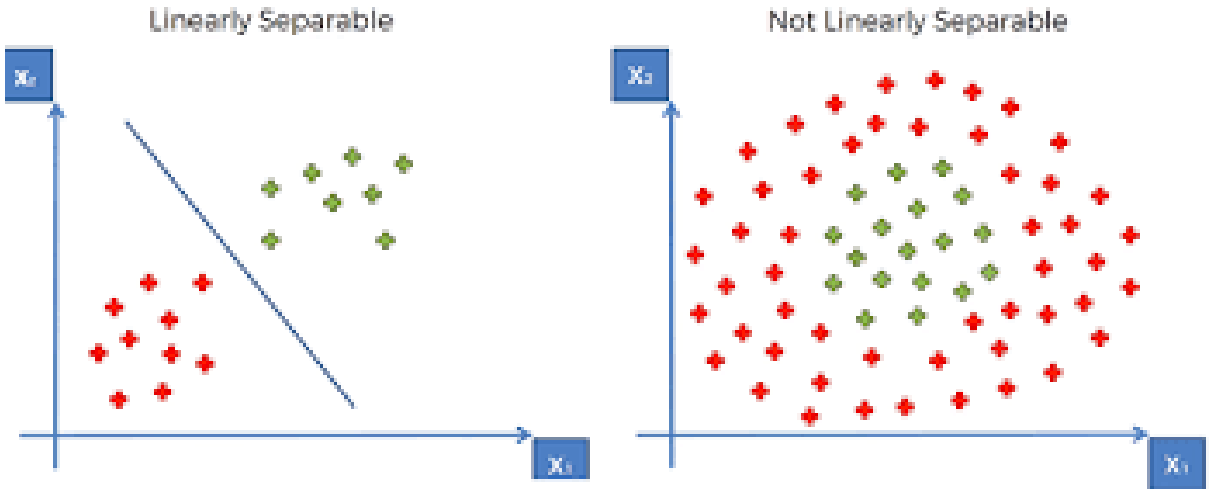
  $H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**
- This becomes a **constrained (convex) quadratic optimization** problem: Quadratic objective function and linear constraints → *Quadratic Programming (QP)* → Lagrangian multipliers

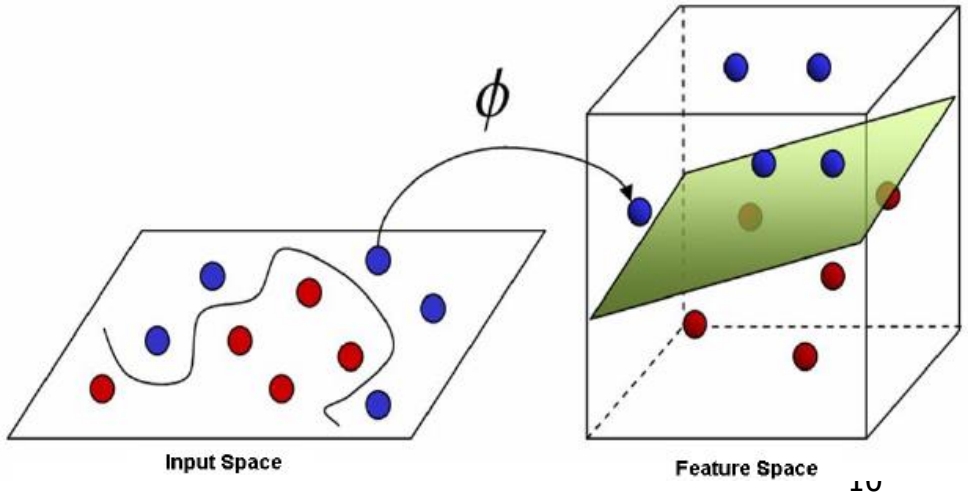
Why Is SVM Effective on High Dimensional Data?

- The **complexity** of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The **support vectors** are the essential or critical training examples —they lie closest to the decision boundary (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

SVM—Linearly Inseparable

- Transform the original input data into a higher dimensional space.
- Non-linear SVMs use kernel functions to transform data into higher-dimensional spaces, allowing for the creation of complex, non-linear decision boundaries that are not possible with linear SVMs, enabling effective classification of non-linearly separable data.
- Instead of computing the dot product on the transformed data, it is math. equivalent to applying a kernel function $K(\mathbf{X}_i, \mathbf{X}_j)$ to the original data, i.e., $K(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i)\ \Phi(\mathbf{X}_j)$
- Search for a linear separating hyperplane in the new space.





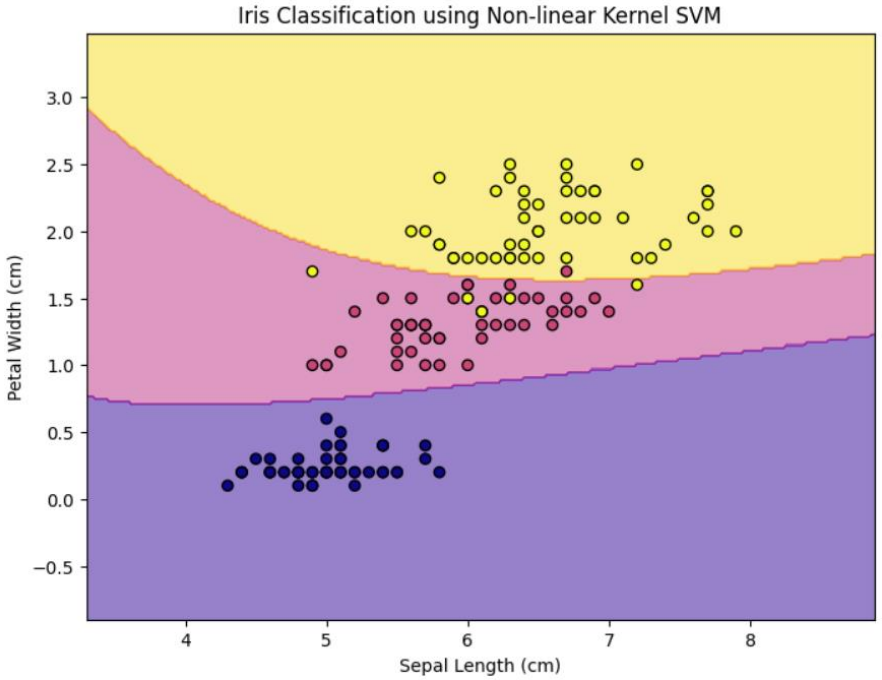
SVM: Different Kernel functions

Typical Kernel Functions

$$\textbf{Polynomial kernel of degree } h: \quad K(\boldsymbol{X_i}, \boldsymbol{X_j}) = (\boldsymbol{X_i} \cdot \boldsymbol{X_j} + 1)^h$$

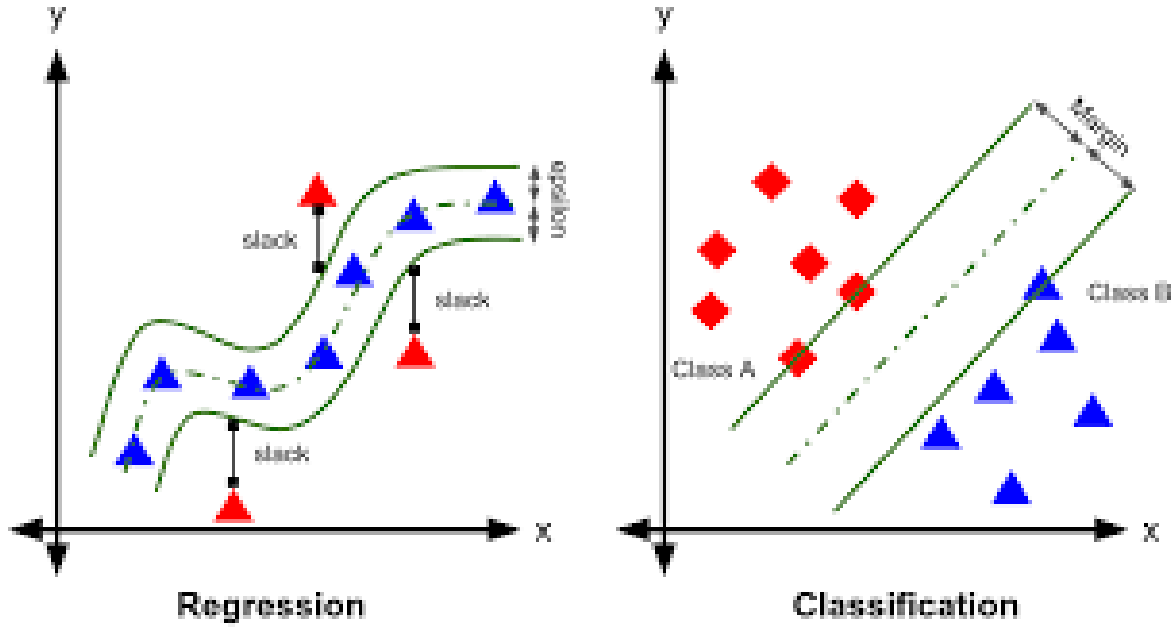
$$\textbf{Gaussian radial basis function kernel}: \quad K(\boldsymbol{X_i}, \boldsymbol{X_j}) = e^{-\|\boldsymbol{X_i} - \boldsymbol{X_j}\|^2 / 2\sigma^2}$$

$$\textbf{Sigmoid kernel}: \quad K(\boldsymbol{X_i}, \boldsymbol{X_j}) = \tanh(\kappa \boldsymbol{X_i} \cdot \boldsymbol{X_j} - \delta)$$



SVM can also be used for

- classifying multiple (> 2) classes and
- regression analysis



Model Evaluation and Selection

- 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 a classifier's accuracy:
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

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

Example of Confusion Matrix:

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

- Given *m* classes, an entry, $\boldsymbol{CM_{i,j}}$ 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

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified.

  Accuracy = (TP + TN)/All

- **Error rate = *1 – accuracy* = (FP + FN)/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

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive.
- **Recall:** completeness – what % of positive tuples did the classifier label as positive?
- Perfect score is 1.0

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN}$$

- Inverse relationship between precision & recall
- ***F* measure (*F_1* - score)**: harmonic mean of precision and recall.

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

Classifier Evaluation Metrics

Classification Metrics Examples

	Predicted		
Actual	Positive	Negative	Row Totals
Positive	60	10	70
Negative	5	25	30
Col Totals	65	35	100

$$\text{Recall} = \frac{60}{70} = 0.857$$

$$\text{Specificity} = \frac{25}{30} = 0.833$$

$$\text{Error} = \frac{15}{100} = 15\%$$

$$\text{Accuracy} = \frac{85}{100} = 85\%$$

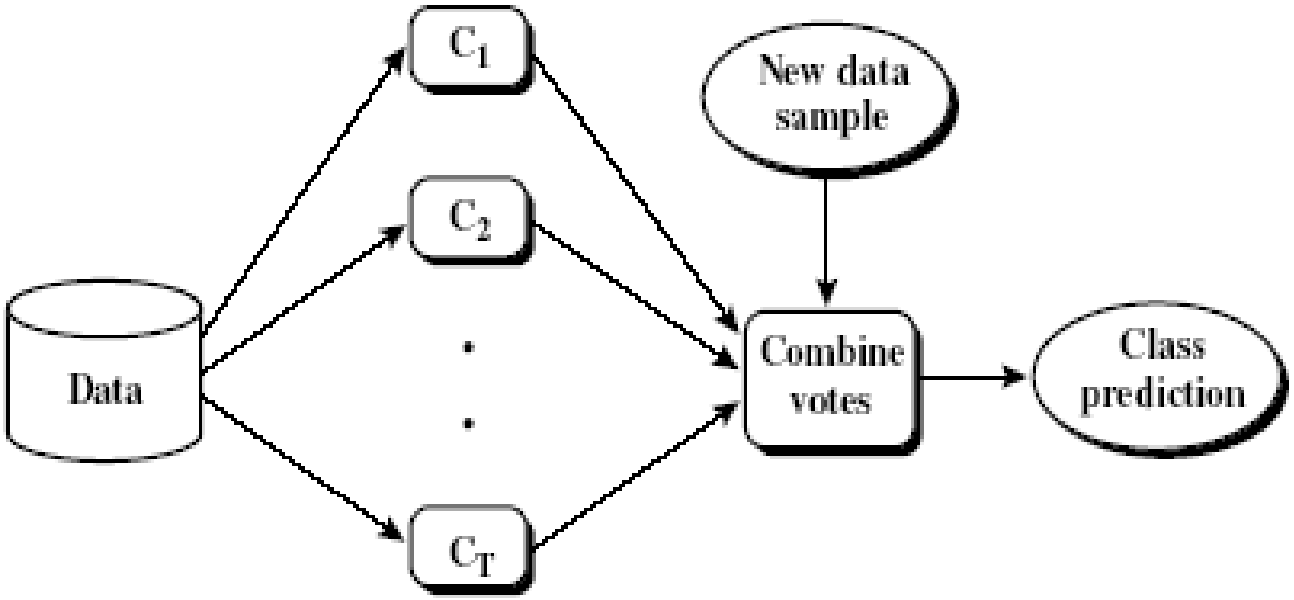
$$\text{Precision} = \frac{60}{65} = 0.923$$

$$F = 2 * \frac{0.857 * 0.923}{0.857 + 0.923} = 0.889$$

Holdout & Cross-Validation Methods

- **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 *i*-th 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

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: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

Random Forest (Breiman 2001)

- Random Forest:
  - Each classifier in the ensemble is a *decision tree* classifier and is generated using a random selection of attributes at each node to determine the split
  - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

Supervised Learning

k-Nearest Neighbor Classifier

- *k*-NN classification rule is to assign to a test sample the majority category label of its *k* nearest training samples.
- In practice, *k* is usually chosen to be odd, so as to avoid ties
- The *k* = 1 rule is generally called the nearest-neighbor classification rule.

k-Nearest Neighbor Classifier

- The model is **not trained** beforehand, it runs at a time of execution to find the query output. There's no point in training the model earlier as an input of the model is training data, **hyperparameter (k), and a query point (xq).**
- **Steps of K-NN classifier:**
  - **Distance Calculation:** KNN calculates the distance between a new data point and all other data points in the dataset.
  - **Finding Nearest Neighbors:** It then identifies the 'k' nearest neighbors to the new data point.
  - **Prediction:** For classification, the new data point is assigned to the class that is most common among its 'k' nearest neighbors. For regression, the predicted value is the average of the values of its 'k' nearest neighbors.

Nearest-Neighbor Classifiers: Issues

- The value of *k*, the number of nearest neighbors to retrieve
- Choice of Distance Metric to compute distance between records
- Computational complexity
  - Size of training set
  - Dimension of data

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

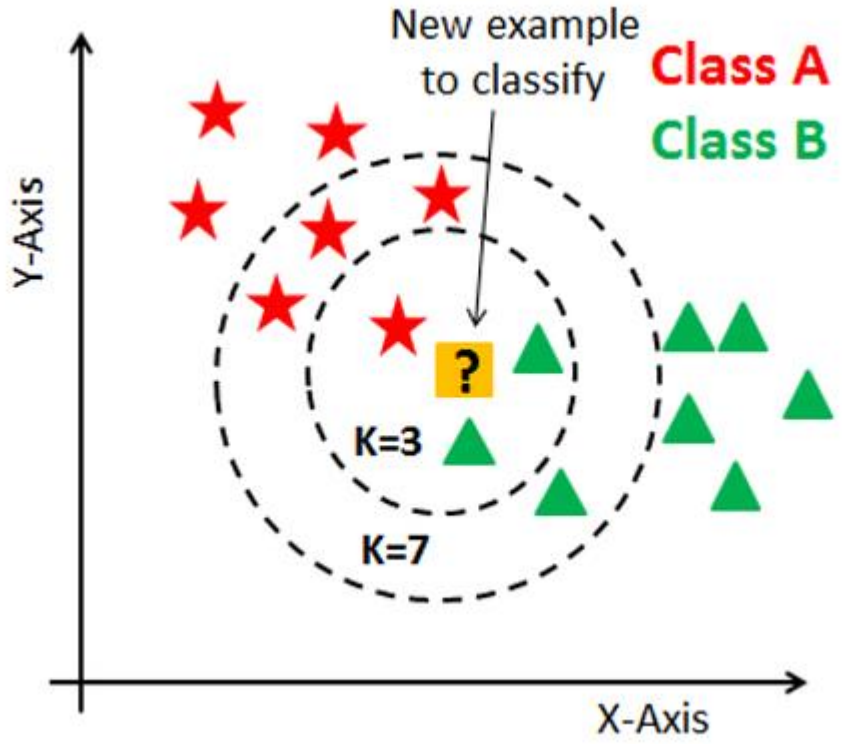
Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Value of K

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes

Rule of thumb:
K = sqrt(N)
N: number of training points



k-Nearest Neighbor Applications

KNN is used in various **applications**, including:

- Image Recognition: Identifying objects in images.
- Recommendation Systems: Suggesting products or content based on user preferences.
- Fraud Detection: Identifying fraudulent transactions.

Advantages:

- Simplicity: KNN is a straightforward algorithm to understand and implement.
- Versatility: It can be used for both classification and regression tasks.
- No Training Phase: It doesn't require a separate training phase, making it efficient for certain datasets.

Nearest Neighbour : Computational Complexity

- Expensive
  - To determine the nearest neighbour of a query point *q*, must compute the distance to all *N* training examples
    - \+ Pre-sort training examples into fast data structures (kd-trees)
    - \+ Compute only an approximate distance (LSH)
    - \+ Remove redundant data (condensing)
- Storage Requirements
  - Must store all training data **P**
    - \+ Remove redundant data (condensing)
    - \- Pre-sorting often increases the storage requirements
- High Dimensional Data
  - "Curse of Dimensionality"
    - Required amount of training data increases exponentially with dimension
    - Computational cost also increases dramatically
    - Partitioning techniques degrade to linear search in high dimension

Supervised Learning

Clustering

- Cluster Analysis: Basic Concepts 
- Partitioning Methods
- Hierarchical Methods
- Density-Based Methods
- Evaluation of Clustering
- Summary

What is Cluster Analysis?

- Cluster: A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- Cluster analysis (or *clustering*, *data segmentation,* …)
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- Unsupervised learning: no predefined classes (i.e., *learning by observations* vs. learning by examples: supervised)
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

Clustering for Data Understanding and Applications

- **Biology:** taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- **Information retrieval:** document clustering
- **Land use:** Identification of areas of similar land use in an earth observation database
- **Marketing:** Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- **City-planning:** Identifying groups of houses according to their house type, value, and geographical location
- **Earth-quake studies:** Observed earthquake epicenters should be clustered along continent faults
- **Climate:** understanding earth climate, find patterns of atmospheric and ocean
- **Economic Science:** market research

Clustering as a Preprocessing Tool (Utility)

- Summarization:
  - Preprocessing for regression, PCA, classification, and association analysis
- Compression:
  - Image processing: vector quantization
- Finding K-nearest Neighbors
  - Localizing search to one or a small number of clusters
- Outlier detection
  - Outliers are often viewed as those “far away” from any cluster

Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters
  - high intra-class similarity: cohesive within clusters
  - low inter-class similarity: distinctive between clusters
- The quality of a clustering method depends on
  - the similarity measure used by the method
  - its implementation, and
  - Its ability to discover some or all of the hidden patterns

Measure the Quality of Clustering

- Dissimilarity/Similarity metric
  - Similarity is expressed in terms of a distance function, typically metric: $d(i, j)$
  - The definitions of distance functions are usually rather different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables
  - Weights should be associated with different variables based on applications and data semantics
- Quality of clustering:
  - There is usually a separate "quality" function that measures the "goodness" of a cluster.
  - It is hard to define "similar enough" or "good enough"
    - The answer is typically highly subjective

Requirements and Challenges

- Scalability
  - Clustering all the data instead of only on samples
- Ability to deal with different types of attributes
  - Numerical, binary, categorical, ordinal, linked, and mixture of these
- Constraint-based clustering
  - User may give inputs on constraints
  - Use domain knowledge to determine input parameters
- Interpretability and usability
- Others
  - Discovery of clusters with arbitrary shape
  - Ability to deal with noisy data
  - Incremental clustering and insensitivity to input order
  - High dimensionality

Major Clustering Approaches (I)

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, CAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSACN, OPTICS, DenClue
- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE

Partitioning Algorithms: Basic Concept

- Partitioning method: Partitioning a database ***D*** of ***n*** objects into a set of ***k*** clusters, such that the sum of squared distances is minimized (where c_i is the centroid or medoid of cluster C_i)

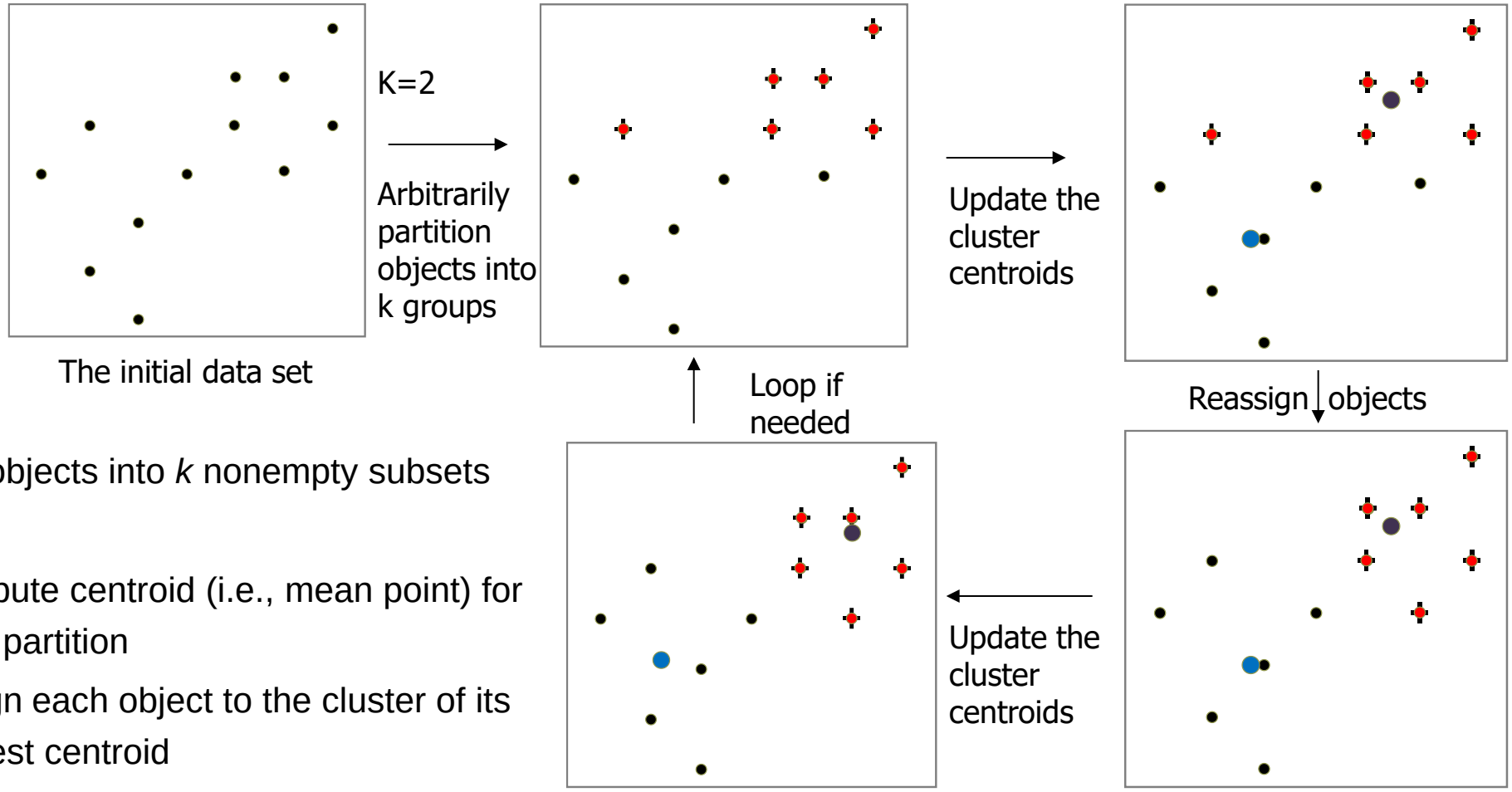
$$E = \Sigma_{i=1}^{k} \Sigma_{p \in C_i} (p - c_i)^2$$

- Given *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* (MacQueen'67, Lloyd'57/'82): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

The *K-Means* Clustering Method

- Given *k*, the *k-means* algorithm is implemented in four steps:
  - Partition objects into *k* nonempty subsets
  - Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., *mean point*, of the cluster)
  - Assign each object to the cluster with the nearest seed point
  - Go back to Step 2, stop when the assignment does not change

An Example of *K-Means* Clustering



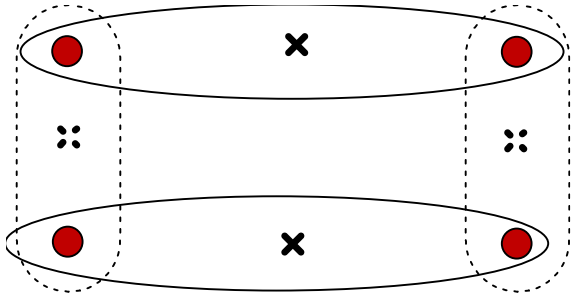
- Partition objects into *k* nonempty subsets
- Repeat
  - Compute centroid (i.e., mean point) for each partition
  - Assign each object to the cluster of its nearest centroid
- Until no change

Comments on the *K-Means* Method

- Strength: *Efficient*: *O*(*tkn*), where *n* is # objects, *k* is # clusters, and *t* is # iterations.
    - Normally, k, t << n.
    - Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$
- Comment: Often terminates at a *local optimal*.
- Weakness
    - Applicable only to objects in a continuous n-dimensional space
        - Using the k-modes method for categorical data
        - In comparison, k-medoids can be applied to a wide range of data
    - Need to specify *k,* the *number* of clusters, in advance (there are ways to automatically determine the best k (see Hastie et al., 2009)
    - Sensitive to noisy data and *outliers*
    - Not suitable to discover clusters with *non-convex shapes*

Variations of the *K-Means* Method

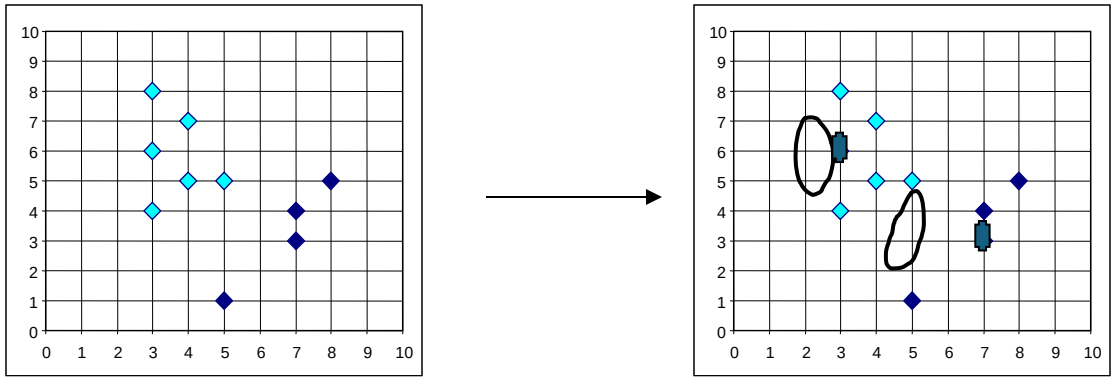
- Most of the variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means



- Handling categorical data: *k-modes*
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: *k-prototype* method

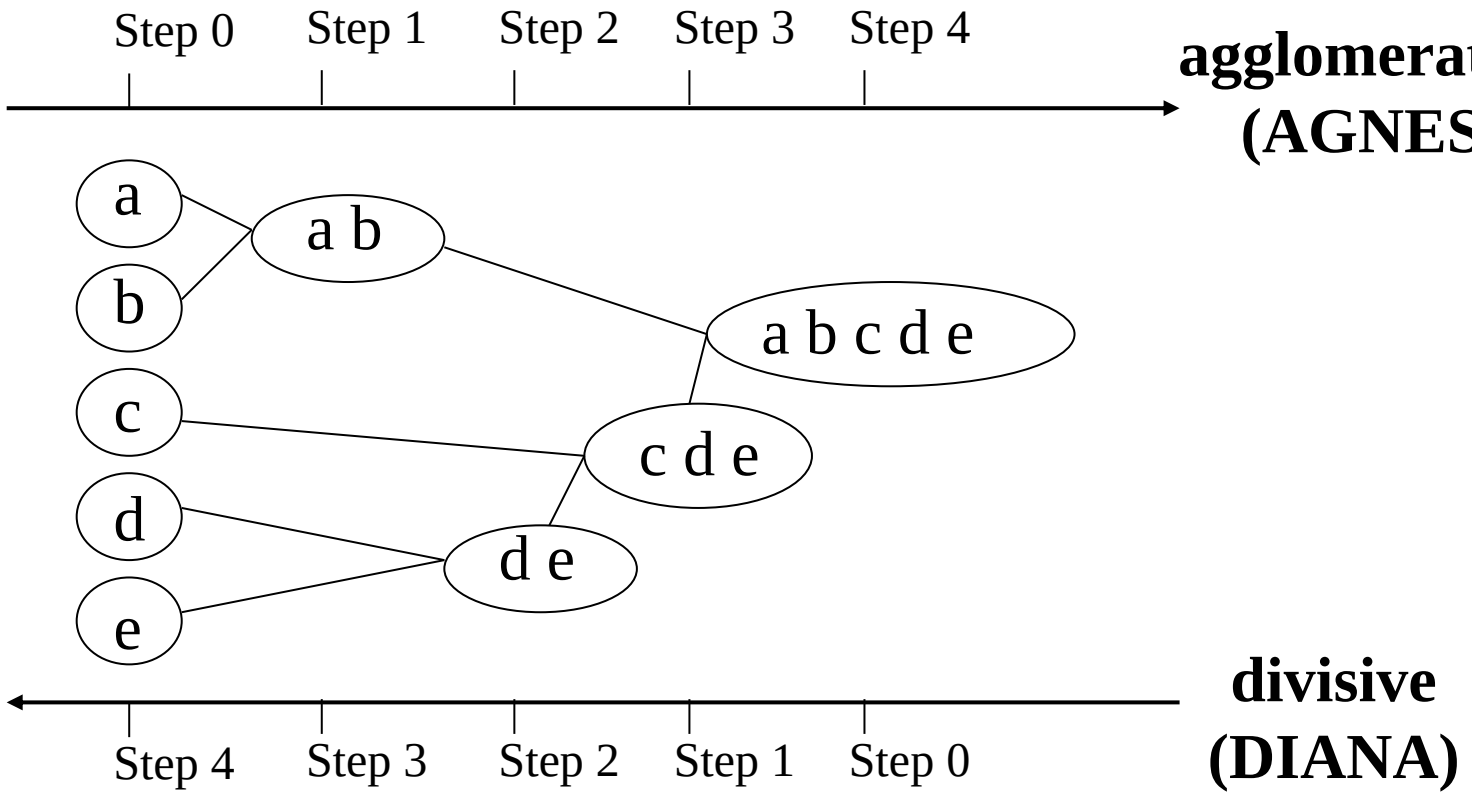
What Is the Problem of the K-Means Method?

- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data
- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster



Hierarchical Clustering

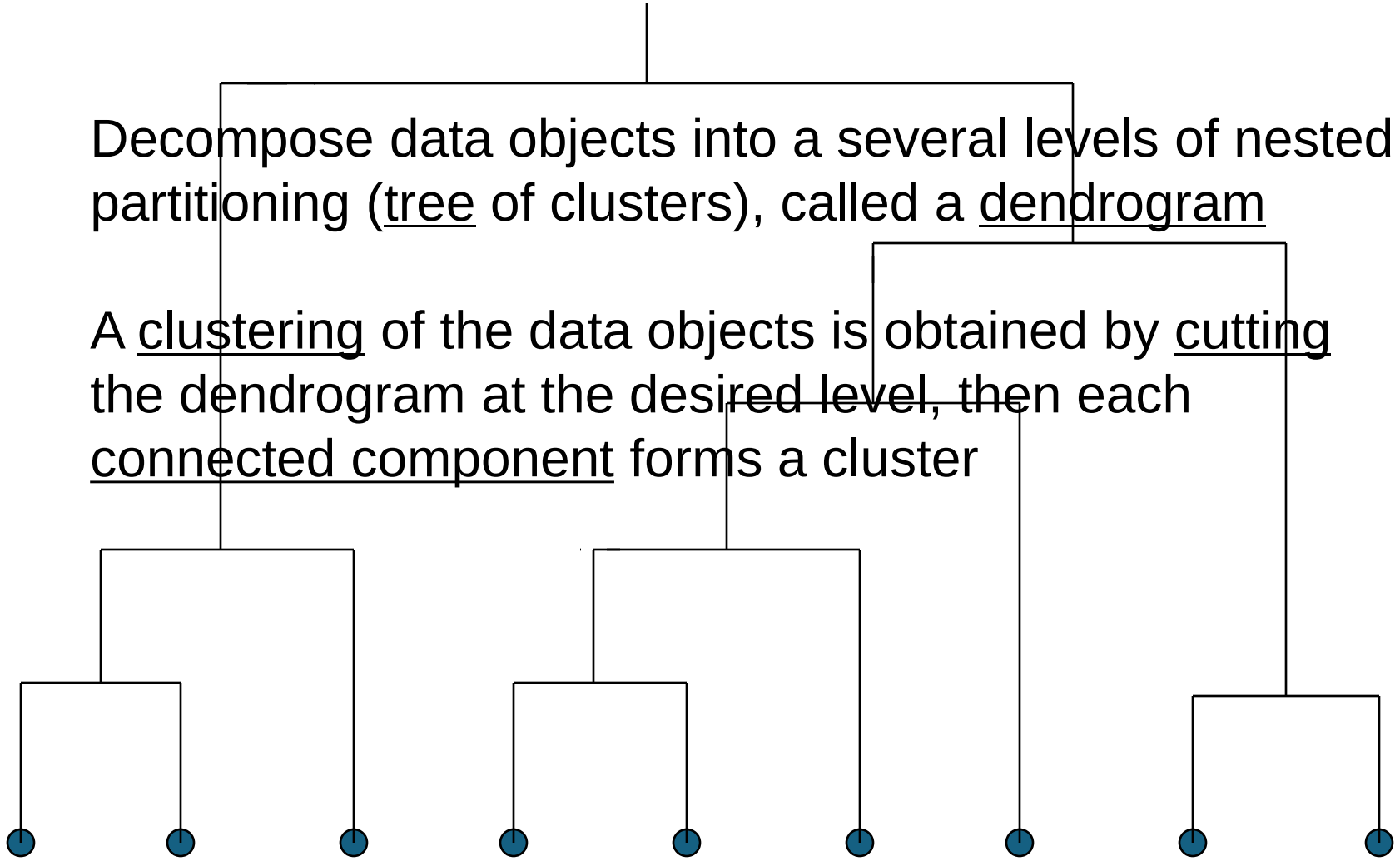
- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition



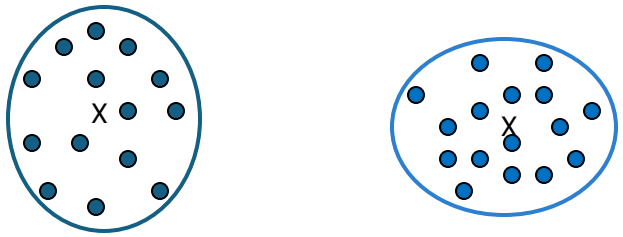
Dendrogram: Shows How Clusters are Merged

Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram

A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster



Distance between Clusters

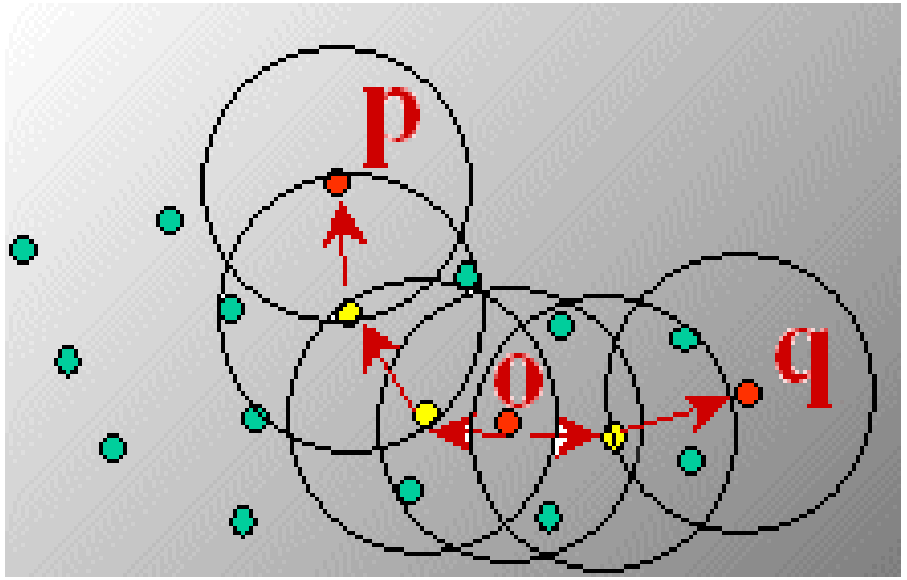


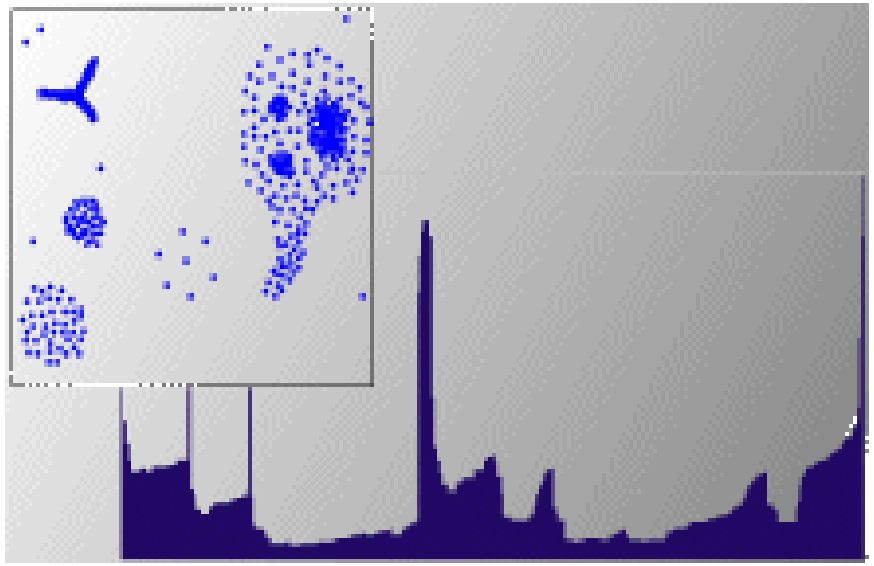
- Single link: smallest distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = min(t_{ip}, t_{jq})$
- Complete link: largest distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = max(t_{ip}, t_{jq})$
- Average: avg distance between an element in one cluster and an element in the other, i.e., $dist(K_i, K_j) = avg(t_{ip}, t_{jq})$
- Centroid: distance between the centroids of two clusters, i.e., $dist(K_i, K_j) = dist(C_i, C_j)$
- Medoid: distance between the medoids of two clusters, i.e., $dist(K_i, K_j) = dist(M_i, M_j)$
  - Medoid: a chosen, centrally located object in the cluster

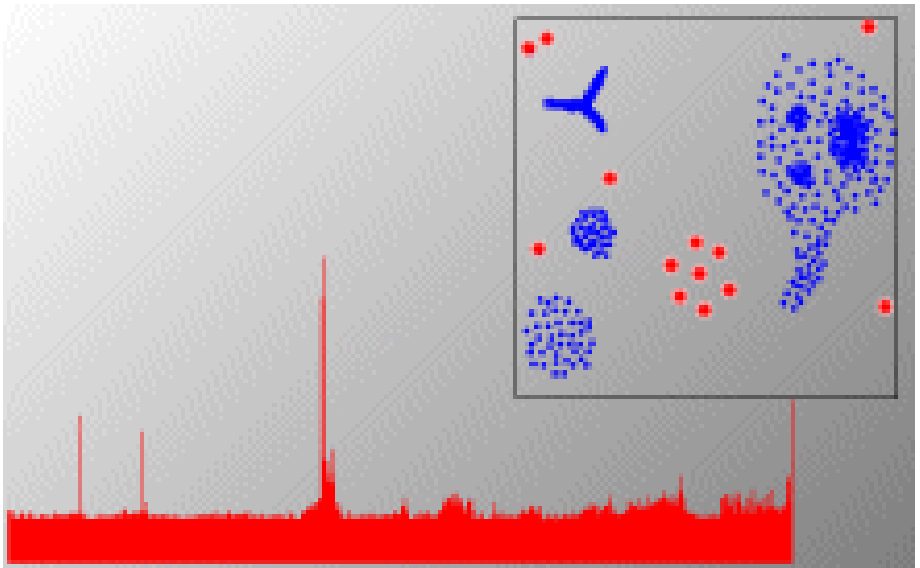
Density-Based Clustering Methods

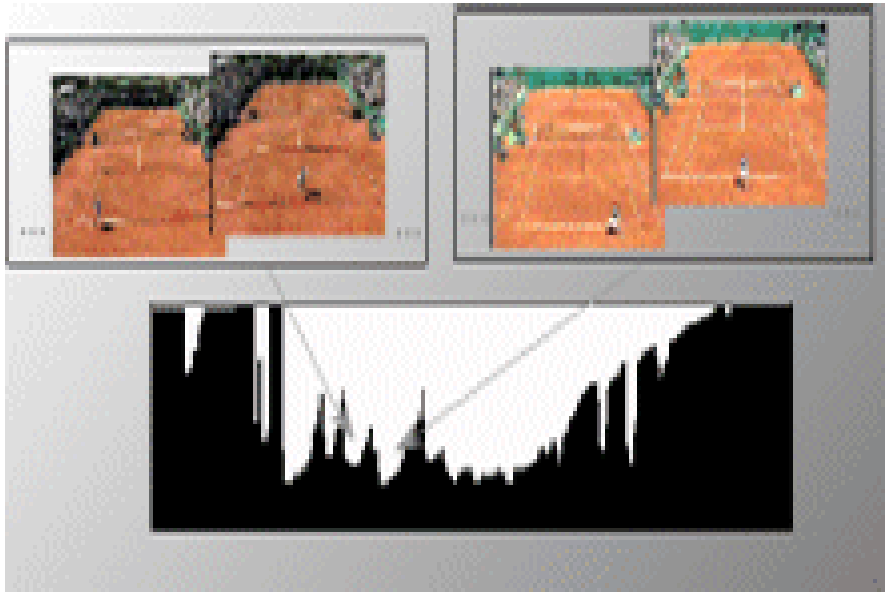
- Clustering based on density (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition
- Several interesting studies:
  - DBSCAN: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)
  - CLIQUE: Agrawal, et al. (SIGMOD'98) (more grid-based)

Density-Based Clustering: OPTICS & Its Applications









Measuring Clustering Quality

- Two methods: extrinsic vs. intrinsic
- Extrinsic: supervised, i.e., the ground truth is available
  - Compare a clustering against the ground truth using certain clustering quality measure
  - Ex. BCubed precision and recall metrics
- Intrinsic: unsupervised, i.e., the ground truth is unavailable
  - Evaluate the goodness of a clustering by considering how well the clusters are separated, and how compact the clusters are
  - Ex. Silhouette coefficient

Measuring Clustering Quality: Extrinsic Methods

- Clustering quality measure: $Q(C, C_g)$, for a clustering C given the ground truth C_g.
- Q is good if it satisfies the following **4** essential criteria
  - Cluster homogeneity: the purer, the better
  - Cluster completeness: should assign objects belong to the same category in the ground truth to the same cluster
  - Rag bag: putting a heterogeneous object into a pure cluster should be penalized more than putting it into a *rag bag* (i.e., "miscellaneous" or "other" category)
  - Small cluster preservation: splitting a small category into pieces is more harmful than splitting a large category into pieces