# Clustering

COMP4211

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

# Supervised Learning vs Unsupervised Learning

## Supervised learning

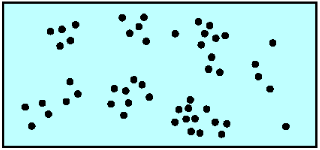
- The learner is provided with a set of inputs together with the corresponding desired outputs
- Given training set: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N; y_N)$
- Find a general function $y = h(\mathbf{x})$
- An approximation to a target (true) function $y = f(\mathbf{x})$
  - $h$: hypothesis

## Unsupervised learning

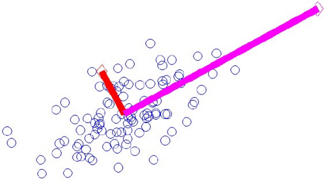
- training examples as input patterns, with no associated output patterns
- Given training set $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$
- unlabeled training examples
- no teacher

# Uses of Unsupervised Learning

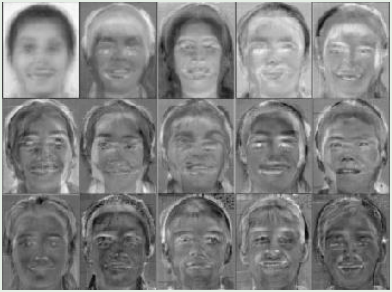
- find clusters



  - in the early stages of an investigation, it may be helpful to perform exploratory data analysis to gain some insight into the nature or structure of the data
- find features or preprocess existing features for the subsequent pattern classification problem (supervised learning)

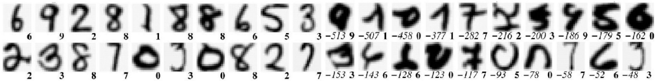


# Example (eigenface)



# Uses of Unsupervised Learning...

- find the least likely observations from a dataset (outlier detection)




**Example (network intrusion detection)**

Detect whether someone is trying to hack the network or doing anything else unusual on the network

**Example (database cleaning)**

want to find out whether someone stored bogus information in a database (typos, etc.), mislabelled digits, ugly digits, bad photographs in an electronic album

**Example (fraud detection)**

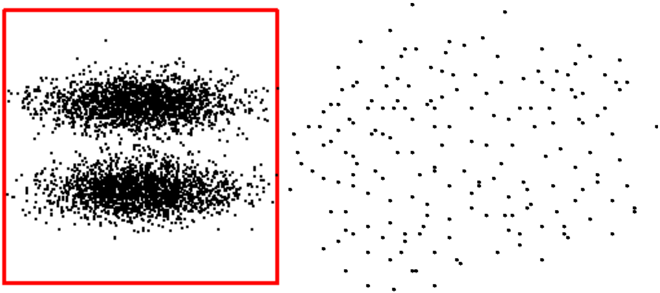
credit cards, telephone bills, medical records

# Problem

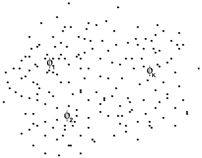
Given:

- $x_1, x_2, \ldots, x_n$
- they fall into $k$ clusters

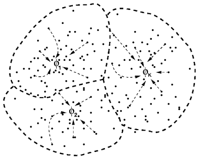
Determine: the cluster centers (centroids) $m_1, m_2, \ldots, m_k$



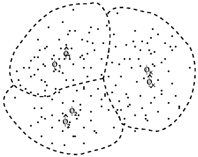
1. Make initial guesses for $m_1, m_2, \ldots, m_k$
   - usually, just randomly choose $k$ of the examples



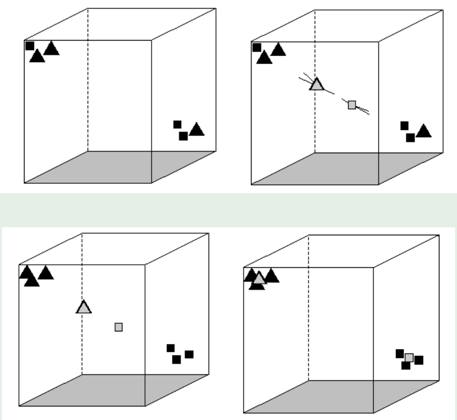
2. Use the estimated cluster centers to put the patterns into clusters
   - put $x_j$ into cluster $i$ if $\|x_j - m_i\|$ is the minimum of all the $k$ distances
   - the feature space is partitioned into $k$ clusters



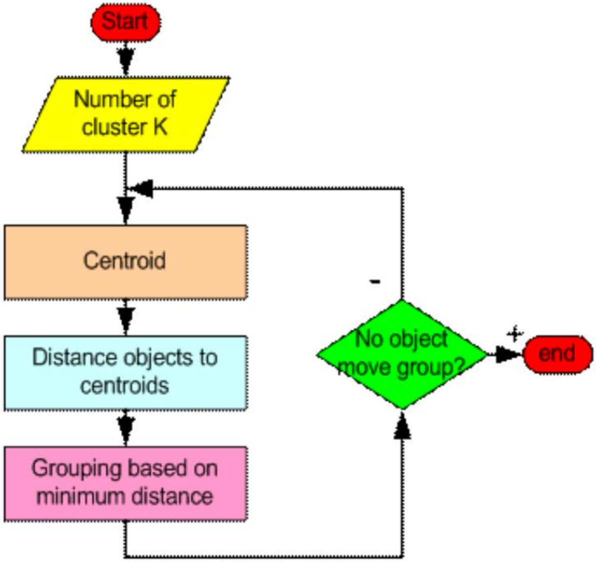
3. for $i = 1$ to $k$, replace $m_i$ with the mean of all examples for cluster $i$
4. Go back to step 2 until there are no changes in the $m_i$'s



## Example

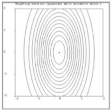


Demo

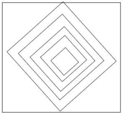

Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

# Distance Measures

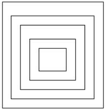
- Euclidean distance: $d(\mathbf{x}, \mathbf{z}) = \sqrt{\sum_{i=1}^{n} (x_i - z_i)^2}$
- scaled Euclidean distance: $d(\mathbf{x}, \mathbf{z}) = \sqrt{\sum_{i=1}^{n} w_i (x_i - z_i)^2}$



- $L_1$ distance: $d(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} |x_i - z_i|$



- $L_\infty$ distance: $d(\mathbf{x}, \mathbf{z}) = \max(|x_i - z_i|)$



# Similarity Measures

similarity functions

- gives a large value when two feature vectors are similar

## Example

Normalized inner product

$$s(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1'\mathbf{x}_2}{\|\mathbf{x}_1\| \cdot \|\mathbf{x}_2\|}$$

- cosine of angle between vectors
- for binary-valued $(0/1)$ features, the normalized inner product gives a relative count of features shared by the two vectors
- a simple variation is the fraction of features shared:

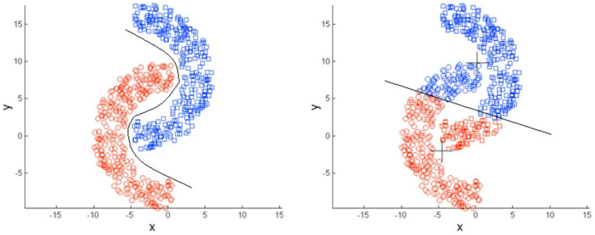
$$s(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1'\mathbf{x}_2}{d}$$

Different initialization means that you may get different clusters each time

- multiple runs
- pick the solution with minimum sum of squared error $\sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mathbf{m}_i\|^2$

Implicit assumptions about the shapes of clusters

- can get wrong results when clusters have other shapes

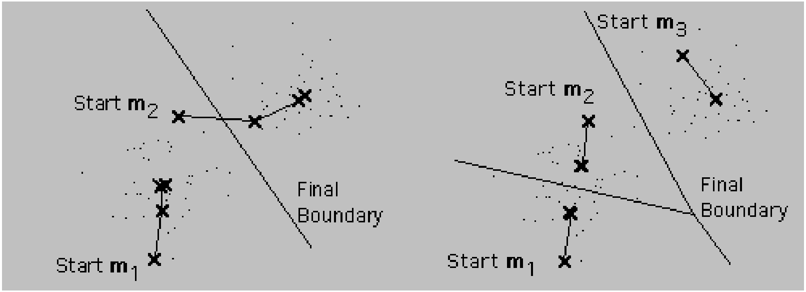


# Issues...

Data points are assigned to only one cluster (hard assignment)

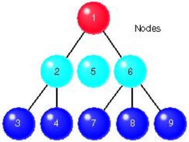
You have to pick the number of clusters

- in general, clustering result depends on $k$



# Hierarchical Clustering

## Hierarchy




- clusters $\rightarrow$ subclusters $\rightarrow$ subsubclusters $\rightarrow \cdots$

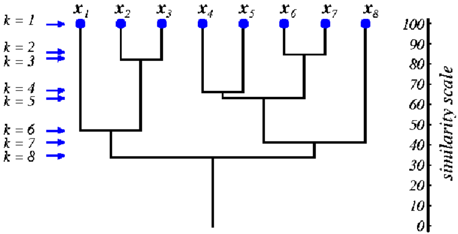
Why do we need hierarchies?

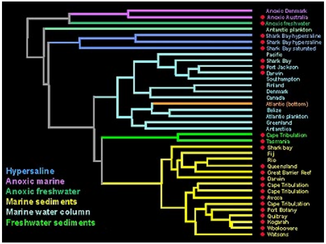
**Example**

Biology, library book categorization, web directories

Construct a sequence of partitions of the $n$ samples into $c$ clusters

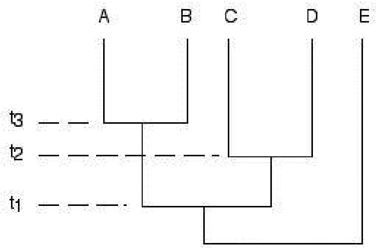
- a partition into $n$ clusters
  - each cluster contains exactly one sample (level 1)
- next a partition into $n-1$ clusters (level 2)
- next a partition into $n-2$ clusters, ... (level 3)
- all the samples form one cluster (level $n$)



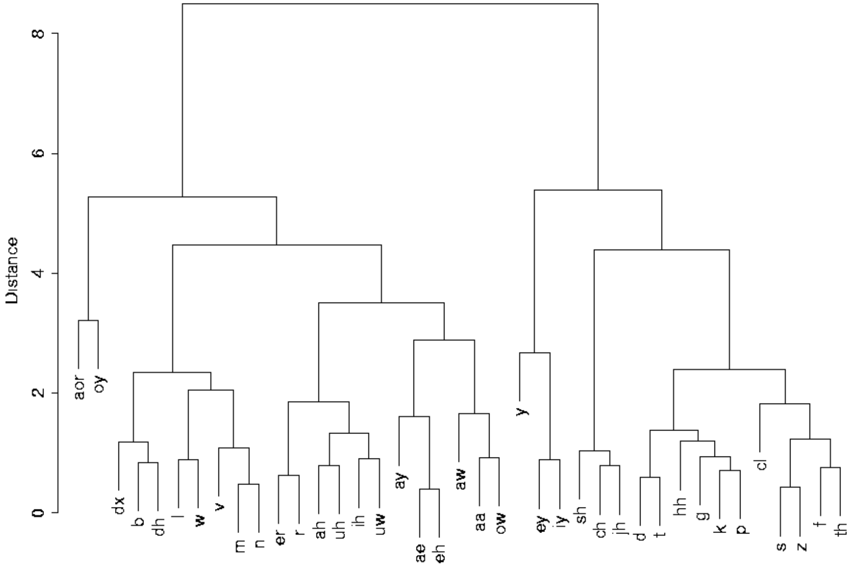

A natural representation: tree (dendrogram)

# Dendrogram




- Given any two samples $x$ and $x'$, at some level they will be grouped together in the same cluster
- If two samples are in the same cluster at some level $c \rightarrow$ at all higher levels ($> c$), they remain together
  - e.g. at level 2, $\mathbf{x}_6$ and $\mathbf{x}_7$ group together and they stay together at all subsequent levels

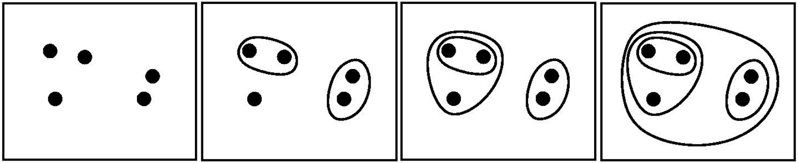
# Example: Dendrogram of 39 English Sounds



# Hierarchical Clustering

How to perform hierarchical clustering?

Agglomerative (bottom-up)



- start with $n$ singleton clusters
- find 2 “nearest” clusters, merge
- re-iterate the process

Divisive (top-down)

- start with all the samples in one cluster
- form the sequence by successively splitting clusters

# Agglomerative Hierarchical Clustering

Suppose $c$ is the desired number of final clusters

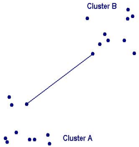
1. initialize: $\hat{c} \leftarrow n, D_i \leftarrow \{\mathbf{x}_i\}, i = 1, \ldots, n$
2. do $\hat{c} \leftarrow \hat{c} - 1$
    1. find "nearest" clusters $D_i$ and $D_j$
    2. merge $D_i$ and $D_j$
3. until $c = \hat{c}$
4. return $c$ clusters

How to measure the distance between two clusters?

# Example Distance Measures between Clusters $D_i, D_j$

The distance between clusters can be defined based on the point-to-point or mean-to-mean distances

- minimum point-to-point distance




$$d_{min}(D_i, D_j) = \min_{\mathbf{x}_1 \in D_i, \mathbf{x}_2 \in D_j} \|\mathbf{x}_1 - \mathbf{x}_2\|$$

- maximum point-to-point distance

$$d_{max}(D_i, D_j) = \max_{\mathbf{x}_1 \in D_i, \mathbf{x}_2 \in D_j} \|\mathbf{x}_1 - \mathbf{x}_2\|$$

- average point-to-point distance

$$d_{avg}(D_i, D_j) = \frac{1}{n_i n_j} \sum_{\mathbf{x}_1 \in D_i, \mathbf{x}_2 \in D_j} \|\mathbf{x}_1 - \mathbf{x}_2\|$$

- mean-to-mean distance

$$d_{mean}(D_i, D_j) = \|\mathbf{m}_i - \mathbf{m}_j\|$$

# Nearest-Neighbor Algorithm

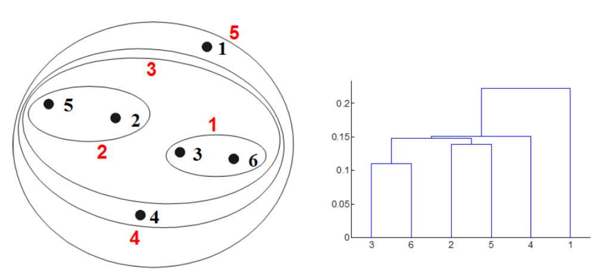
Uses the minimum point-to-point distance

- aka nearest-neighbor clustering algorithm, minimum algorithm

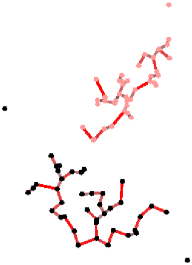
Single-linkage algorithm (demo)

- a variation
- the algorithm is terminated when the distance between nearest clusters exceeds an arbitrary threshold

## Example



# Nearest-Neighbor Algorithm...

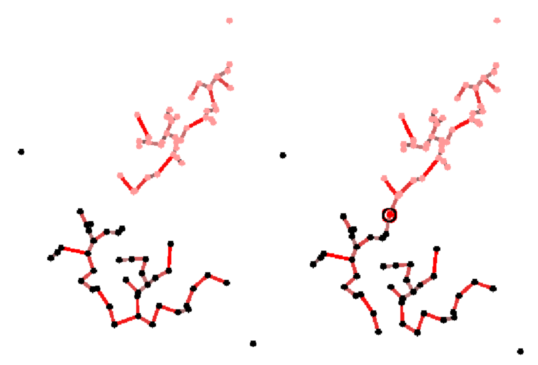


- merging of $D_i$ and $D_j$ $\rightarrow$ add an edge between the nearest pair of nodes in $D_i$ and $D_j$
- tends to produce long, "loose" clusters

# Limitation

Sensitive to noise or slight changes in positions of the data points

## Example



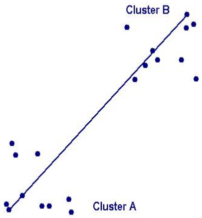
- different result due to addition of a new point

- single-linkage

# Farthest-Neighbor Clustering

Uses the maximum point-to-point distance

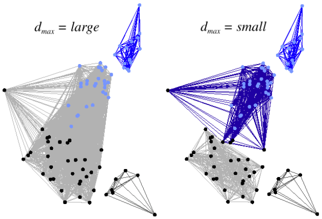
- aka farthest-neighbor clustering algorithm, maximum algorithm




complete-linkage algorithm (demo)

- a variation
- the algorithm is terminated when the distance between nearest clusters exceeds an arbitrary threshold

# Farthest-Neighbor Clustering...





- At each iteration, the size (largest diameter) of the partition is increased as little as possible
  - tends to produce very tight clusters
  - problematic if the true clusters are elongated