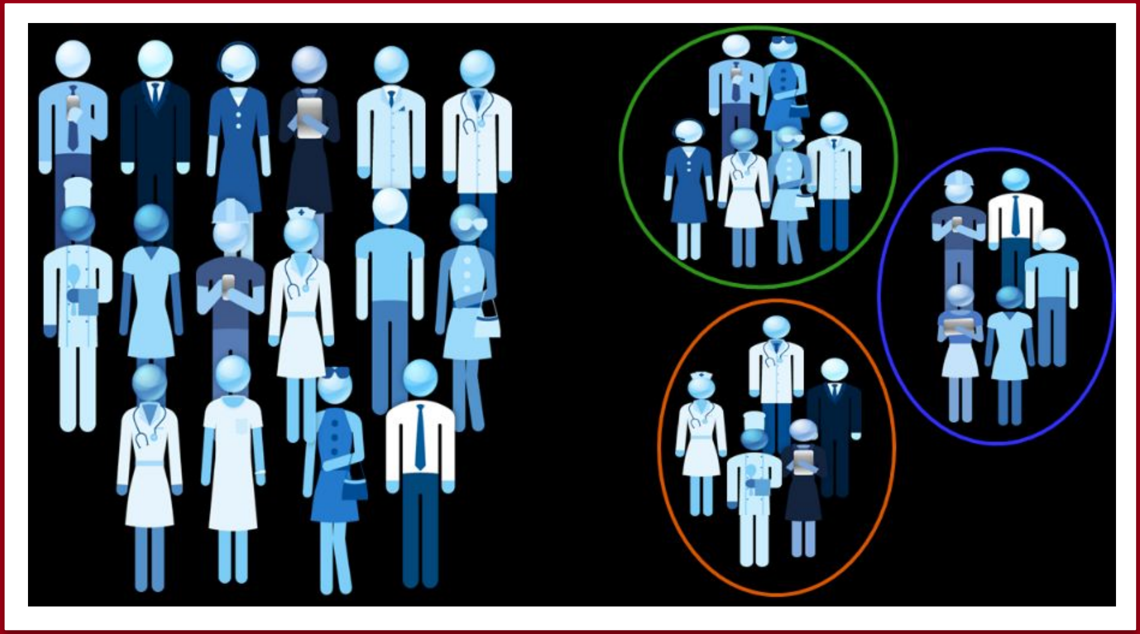
DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA



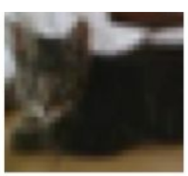
Clustering

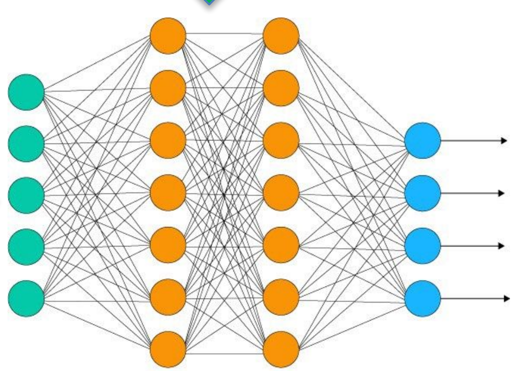
Machine Learning 2022-23
UML book chapter 22

Supervised Learning

Training data
with labels

Training procedure



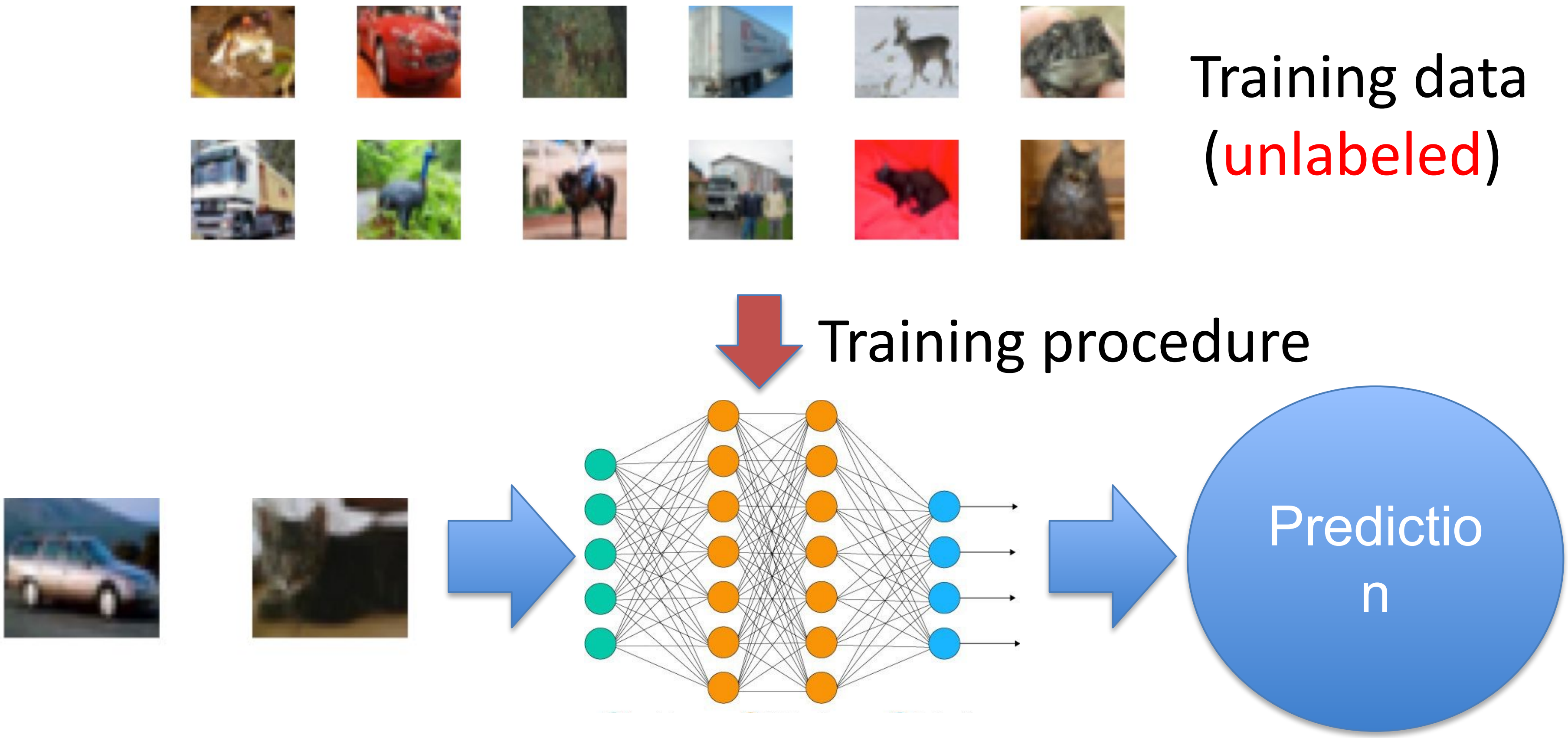


Predicted Label

ML model
(training: estimate parameters)

Data to be analyzed

Unsupervised Learning

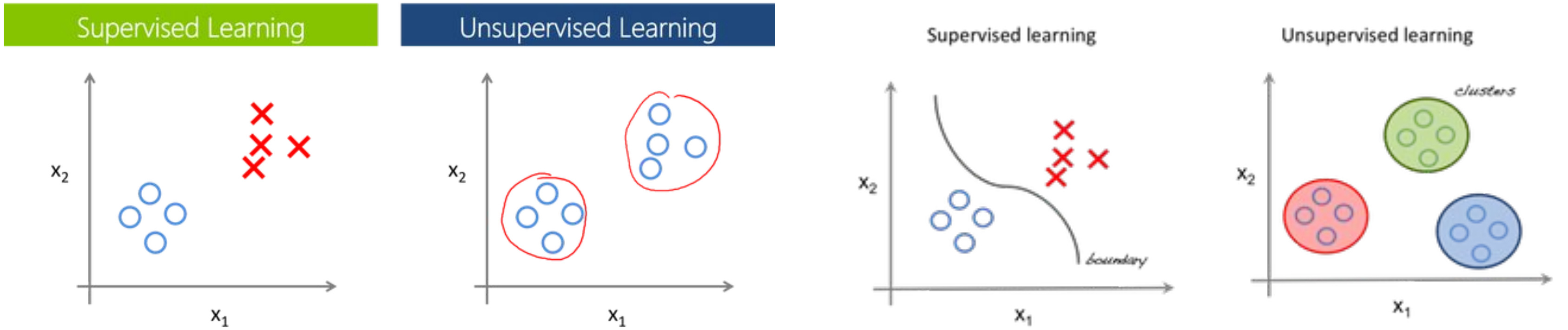


Data to be analyzed

ML model
(training: estimate parameters)

Unsupervised Learning

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE



- *Supervised learning*: There is a labeled training/validation set that can be used to tune the algorithm parameters
- ***Unsupervised Learning***: Training data is not labeled

Unsupervised Learning:

- We are interested in finding some interesting structure in the data, or, equivalently, to organize it in some meaningful way
- *Target:* find a function that describes the structure of "*unlabeled*" data (i.e., data that has not been classified or categorized)
- Several approaches are based on the idea that the data is the realization of an *hidden probability density function*, i.e., unsupervised learning is linked to the density estimation of an hidden PDF producing the data

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Unsupervised Learning Techniques

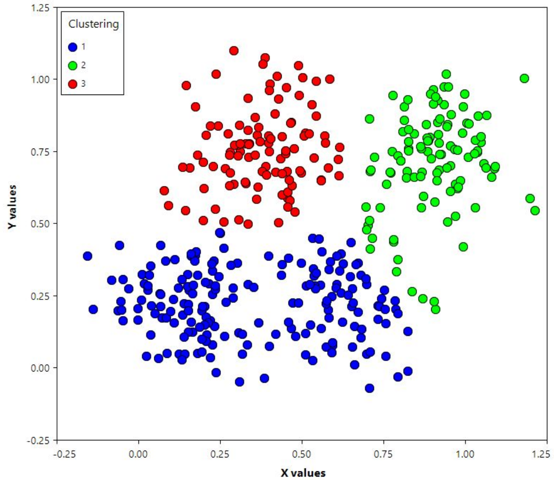
We are going to see only a couple of unsupervised learning techniques and a few very simple and commonly used methods

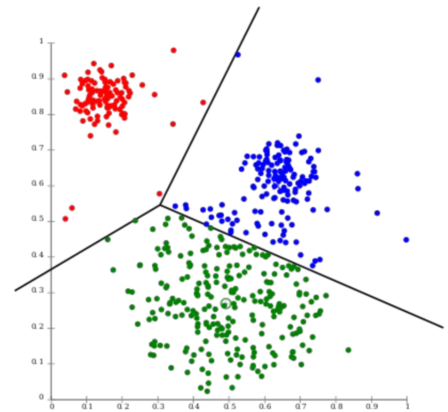
1. *Clustering*
   - *K-means*
   - *Linkage-based clustering*
2. Dimensionality reduction
   - Principal Component Analysis (PCA)

There are many other techniques (*not part of this course*)

- Mean shift clustering, spectral clustering….
- Compressive sensing

Clustering





Clustering

Idea: Divide a set of objects represented by *N*-dimensional vectors into groups (*clusters*) of similar objects

- Key target: *identifying meaningful groups among data points*
- The definition is not rigorous and may be ambiguous, different definitions have been proposed leading to different algorithms

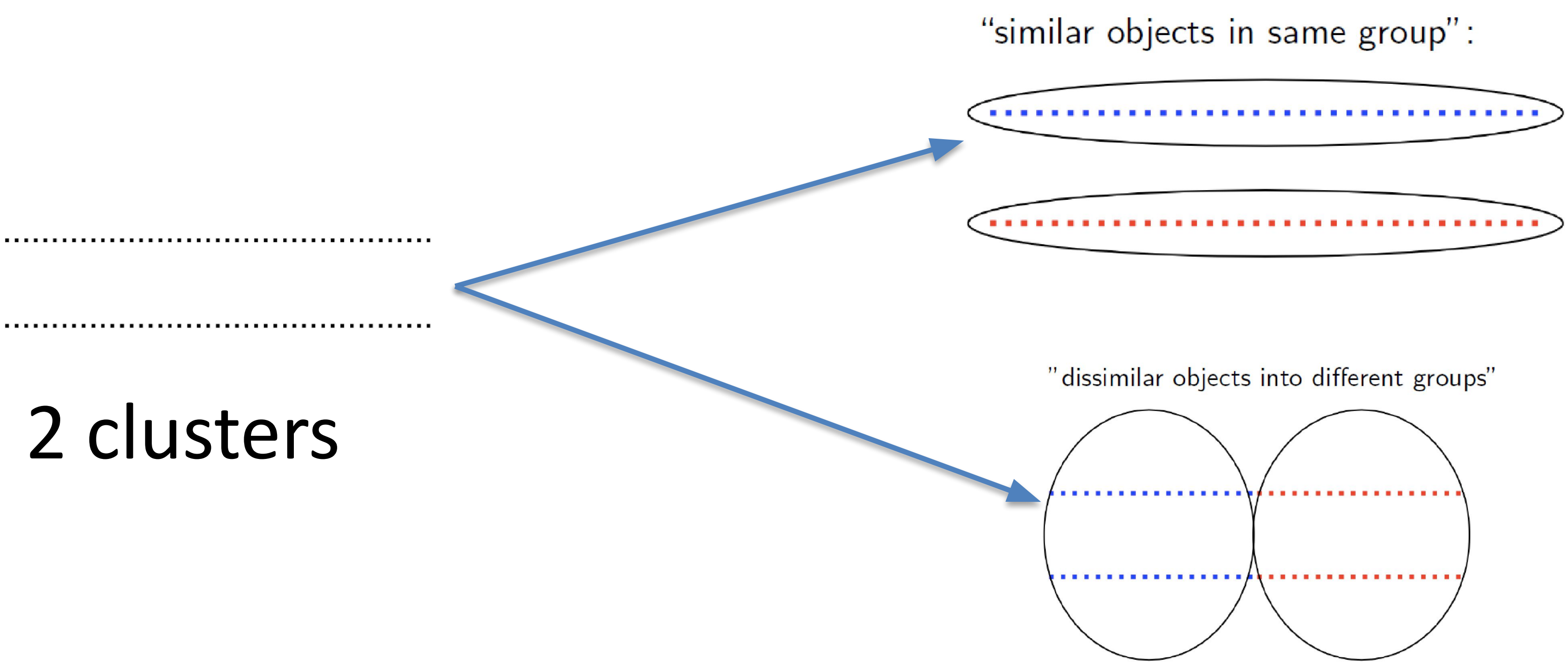
Formal Definition:

Clustering is the task of grouping a set of objects such that *similar objects end up in the same group* and *dissimilar objects are separated into different groups*

Challenges (1)

Similarity is not transitive:

"*similar objects in same group*" and "*dissimilar objects into different groups*" may contradict each other...

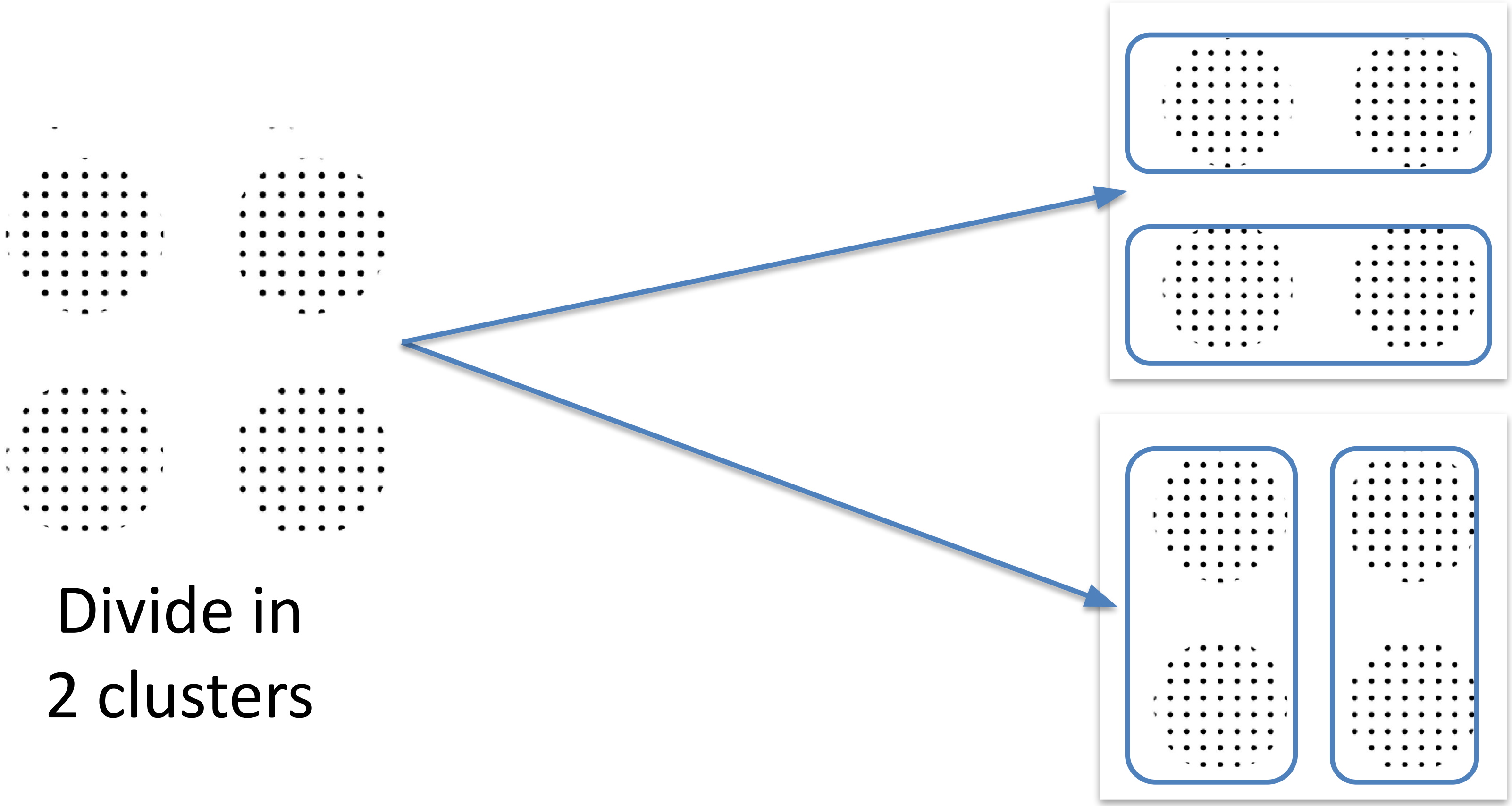


DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Challenges (2)

There is no ground truth:

How To Evaluate Performances ?



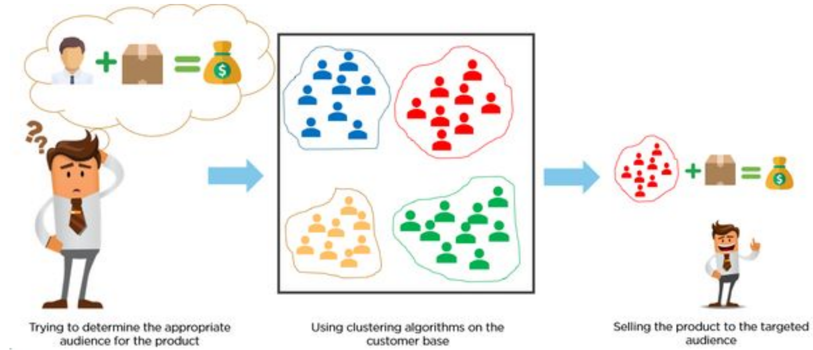
Divide in
2 clusters

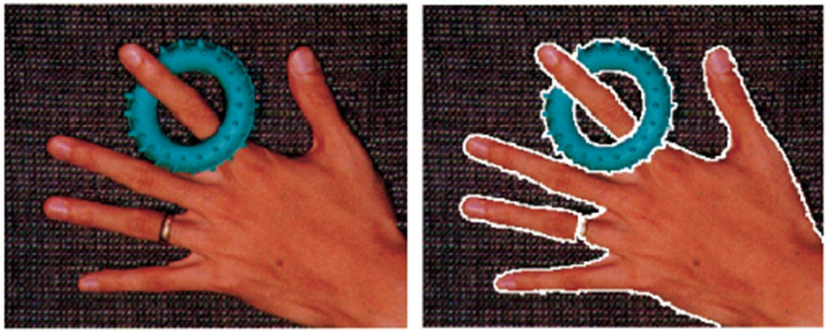
DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

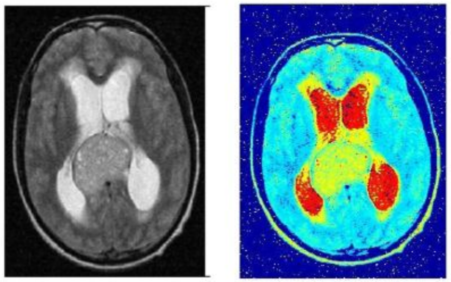
Clustering: Applications

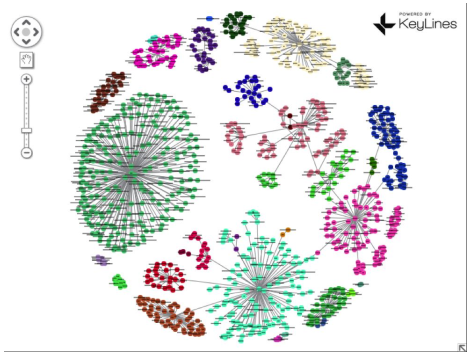
Many Applications:

- Business and marketing
  - Market research
  - Grouping of shopping items
- World Wide Web
  - Search engines
  - Social network analysis
- Image segmentation
- Medicine, Medical imaging
- Biology and bioinformatics
- Recommender systems
- Anomaly detection
- Natural language processing





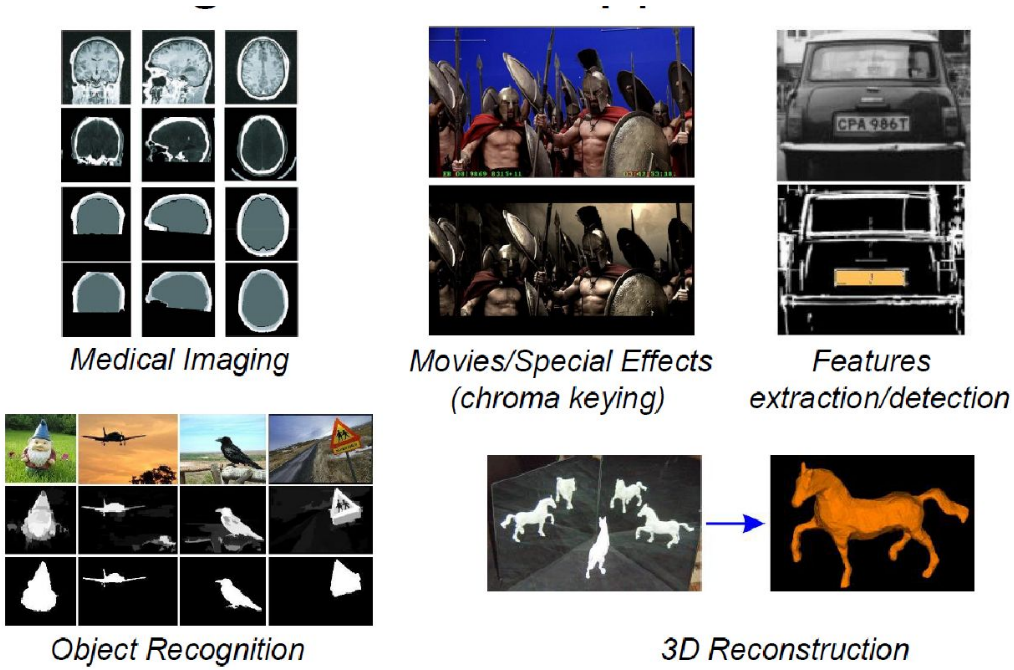


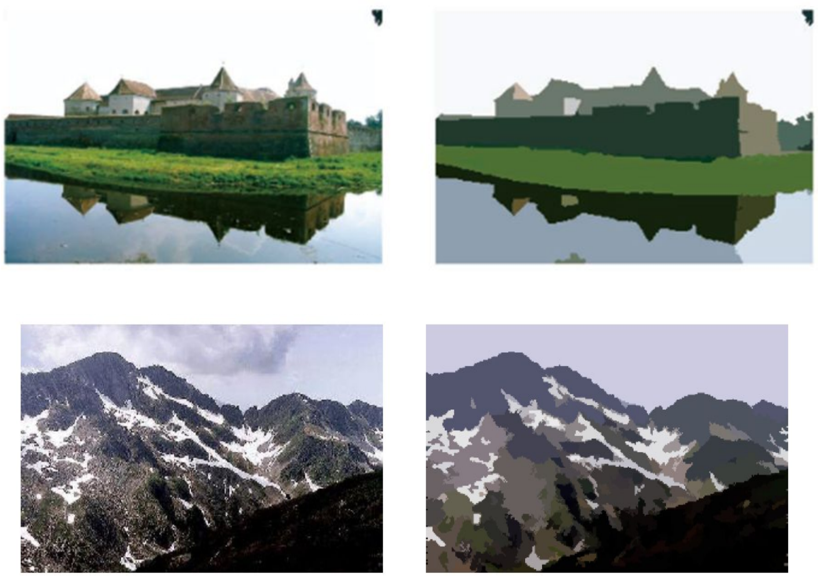


Example (1): Customer Segmentation

- Data: features (e.g. product bought, demographic info, etc.) for a large number of customers
- Goal: **customers segmentation** = identify subgroups of homogeneous customers
- useful for: advertizing, product development, ...

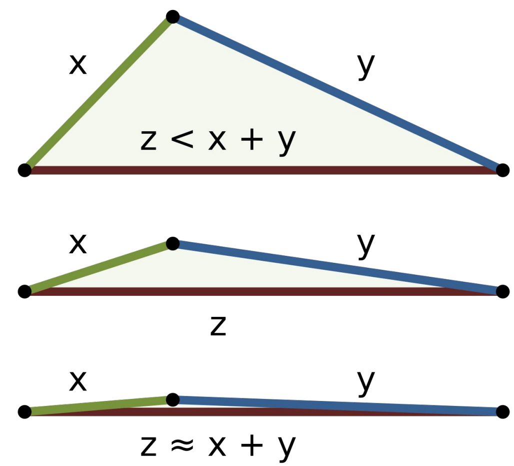
Example (2): Image Segmentation





- *Data*: one sample for each pixel
- *Samples*: vectors containing features (e.g., color or spatial position of pixels)
- *Goal*: divide the image into regions (*clusters*) with uniform properties
- Useful for medical imaging, image analysis, background segmentation in movies, object recognition, ….

Clustering Model



Input:

- Set of elements $\boldsymbol{x} \in \mathcal{X}$
- Distance function $d\colon\ \mathcal{X} x \mathcal{X} \rightarrow \mathbb{R}_+$
  - symmetric, i.e.,: $d(\boldsymbol{x},\boldsymbol{y}) = d(\boldsymbol{y},\boldsymbol{x})\ \forall\ \boldsymbol{x},\boldsymbol{y}$
  - $d(\boldsymbol{x},\boldsymbol{y}) \geq 0\ \forall\ \boldsymbol{x},\boldsymbol{y}$ and $d(\boldsymbol{x},\boldsymbol{x}) = 0$
  - Triangle inequality $d(\boldsymbol{x},\boldsymbol{z}) \leq d(\boldsymbol{x},\boldsymbol{y}) + d(\boldsymbol{y},\boldsymbol{z})$

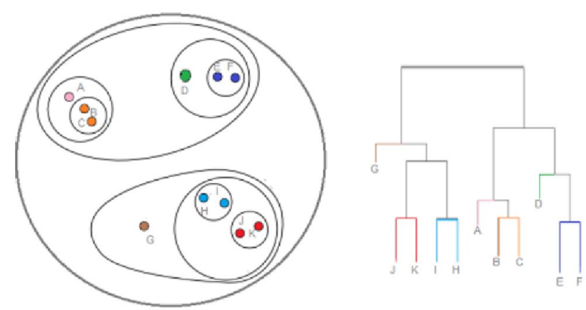
Output:

A partition $C = (C_1, C_2, .., C_k)$ of set $\mathcal{X}$ into k clusters

- $\cup_{i=1}^{k} C_i = \mathcal{X}$
- $\forall i \neq j\colon C_i \cap C_j = \emptyset$

- k (# of clusters): sometimes given in input, sometimes computed by the algorithm

Dendrogram: tree, with input points $\mathbf{x} \in \mathcal{X}$ as leaves, that shows the arrangement/relation between clusters.



Sometimes, the output is a dendrogram (from Greek Dendron = tree, gramma = drawing), a tree diagram showing the arrangement of the clusters

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Distance(cost)-Based Clustering

Very Common approach in clustering:

- Define a cost function over possible partitions of the objects
- Find the partition (→clustering) of minimal cost

Assumptions:

- Data points come from a larger space $\mathcal{X}'$ (typically $\mathbb{R}^n$)
- Distance function $d(\boldsymbol{x}, \boldsymbol{x}')$ for $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$
- For simplicity: assume $\mathcal{X}' = \mathbb{R}^n$ and $d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2$

K-Means

- The simplest clustering algorithm (proposed in 1957, also known as Lloyd algorithm)
- Choose a fixed number of clusters
- Find cluster centers and point-cluster allocations in order to minimize the error made by approximating the points with the cluster centers
- Can't do this by exhaustive search, because there are too many possible allocations
- Iterative algorithm
  - fix cluster centers; assign each point to the closest cluster
  - fix allocation; compute best cluster centers
- Vectors ***x*** can be any set of features for which we can compute a distance (*careful about scaling for non-homogenous data*)

Some material on K-Means D.A. Forsyth

K-Means: Notation and Target

	Set of vectors to be clustered
	Vector to be clustered
k	Number of clusters (*parameter of the algorithm*)
	Clusters (each vector ***x*** is associated to a cluster)
	Centroids of the clusters

- Find cluster centers and allocations in order to minimize the error made by approximating the points with the cluster centers:

if using euclidean distance

$$\boldsymbol{\mu}_i(C_i) = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \sum_{x\in C_i} d(\boldsymbol{x},\boldsymbol{\mu})^2 = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \sum_{x\in C_i} ||\boldsymbol{x}-\boldsymbol{\mu}||^2$$

$$\mathrm{G_{km}}\big((\mathcal{X},d),(C_1,\ldots,C_k)\big) = \sum_{i=1}^{k} \sum_{x\in \mathrm{C}_i} d(\boldsymbol{x},\boldsymbol{\mu}_i)^2$$

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Centroid Computation

Theorem:
Given a cluster C_i, the center $\boldsymbol{\mu_i}$ that minimizes $\sum_{x \in C_i} d(\boldsymbol{x}, \boldsymbol{\mu_i})^2$ is

$$\boldsymbol{\mu_i} = \frac{1}{|C_i|} \sum_{x \in C_i} \boldsymbol{x}$$

- Demonstration: compute gradient and set to 0

$$\frac{\partial}{\partial \boldsymbol{\mu_i}} \left(\sum_{x \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2 \right) = \sum_{x \in C_i} 2(\boldsymbol{x} - \boldsymbol{\mu}_i) = \boldsymbol{0} \rightarrow \sum_{x \in C_i} \boldsymbol{x} = |C_i| \boldsymbol{\mu}_i \rightarrow \boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{x \in C_i} \boldsymbol{x}$$

- Naive (brute-force) algorithm to solve K-Means Clustering?
- Try all possible partitions of the *m* points into *k* clusters, evaluate each partition, and find the best one
- Is it efficient?
    - Number of possible partitions is exponential in *m*
    - NP-Hard problem

K-Means: Algorithm

Procedure:

1. Select *k* random centroids (*or use some more advanced initialization strategy*)
2. Each point is associated to the closest centroid (according to the distance measure)

$$\forall i\colon C_i = \{\boldsymbol{x} \in \mathcal{X} : i = \underset{j}{\operatorname{argmin}} \|\boldsymbol{x} - \boldsymbol{\mu}_j\|\}$$

3. Compute the new centroids (each centroid is the barycentre of the associated points)

$$\forall i : \boldsymbol{\mu}_i = \frac{\sum_{x \in C_i} \boldsymbol{x}}{|C_i|}$$

4. Repeat step 2 and 3 until the algorithm converges

Theorem: at each iteration the value of the objective function G_{km} does not increase

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

Demonstration

Theorem: at each iteration the value of the objective function G_{km} does not increase

1. Consider K-means objective func. (simplified notation) $G(C_1, \dots, C_k) = \min_{\boldsymbol{\mu}_1,\dots,\boldsymbol{\mu}_k \in \mathbb{R}^n} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2$
2. Centroids minimize distance w.r.t points in the associated cluster

$$\boldsymbol{\mu}(C_i) \stackrel{\text{def}}{=} \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} \boldsymbol{x} = \underset{\boldsymbol{\mu} \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2$$

3. We can rewrite the objective function as $G(C_1, \dots, C_k) = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}(C_i)\|^2$
4. Define with $C_i^{(t)}$ the *i-th* cluster at time *t*
5. *Centroid computation:* The new centroids minimize the distance w.r.t the points in the cluster

$$G\left(C_1^{(t)}, \dots, C_k^{(t)}\right) = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i^{(t)}} \left\|\boldsymbol{x} - \boldsymbol{\mu}_i^{(t)}\right\|^2 \le \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i^{(t)}} \left\|\boldsymbol{x} - \boldsymbol{\mu}_i^{(t-1)}\right\|^2$$

6. *Points allocation:* Each point is assigned to the closest centroid

$$\sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i^{(t)}} \left\|\boldsymbol{x} - \boldsymbol{\mu}_i^{(t-1)}\right\|^2 \le \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i^{(t-1)}} \left\|\boldsymbol{x} - \boldsymbol{\mu}_i^{(t-1)}\right\|^2$$

7. By placing all together:

From 5

$$G\left(C_1^{(t)}, \dots, C_k^{(t)}\right) = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i^{(t)}} \left\|\boldsymbol{x} - \boldsymbol{\mu}_i^{(t)}\right\|^2 \le \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i^{(t)}} \left\|\boldsymbol{x} - \boldsymbol{\mu}_i^{(t-1)}\right\|^2 \le \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i^{(t-1)}} \left\|\boldsymbol{x} - \boldsymbol{\mu}_i^{(t-1)}\right\|^2 = G\left(C_1^{(t-1)}, \dots, C_k^{(t-1)}\right)$$

From 6

Note: monotonic not decreasing, but no guarantees on # iterations to converge and could fall in local min.

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

K-Means: Stopping Criteria

1. The centroids positions and allocations do not change any more
2. Error improvement below threshold in 2 consecutive iterations ($\Delta G < T_1$)
3. Maximum number of iterations
4. Reached a target value for *G* ($G < T_2$)

Complexity:

- Assignment of *m* points in $\mathbb{R}^n$ to *k* clusters : time *O(kmn)*
- Computation of centers: time *O(mn)*
- If convergence after *t* iterations: *O(tkmn)*
- In practice convergence after a few iterations but can be very long on some critical cases

Notation: m: #samples, k: # clusters, n: dimensionality of data

K-Means: *Pros* and *Cons*

Pros

- Fast and simple
  - True in practice, in theory it is a NP-hard problem
- Always converges and typically also very fast

Cons

- It does not guarantee an optimal solution
- The solution depends on the initial centroids
- K must be known a priori
- Forces spherical symmetry of clusters (in the *n*-dimensional space)

Examples

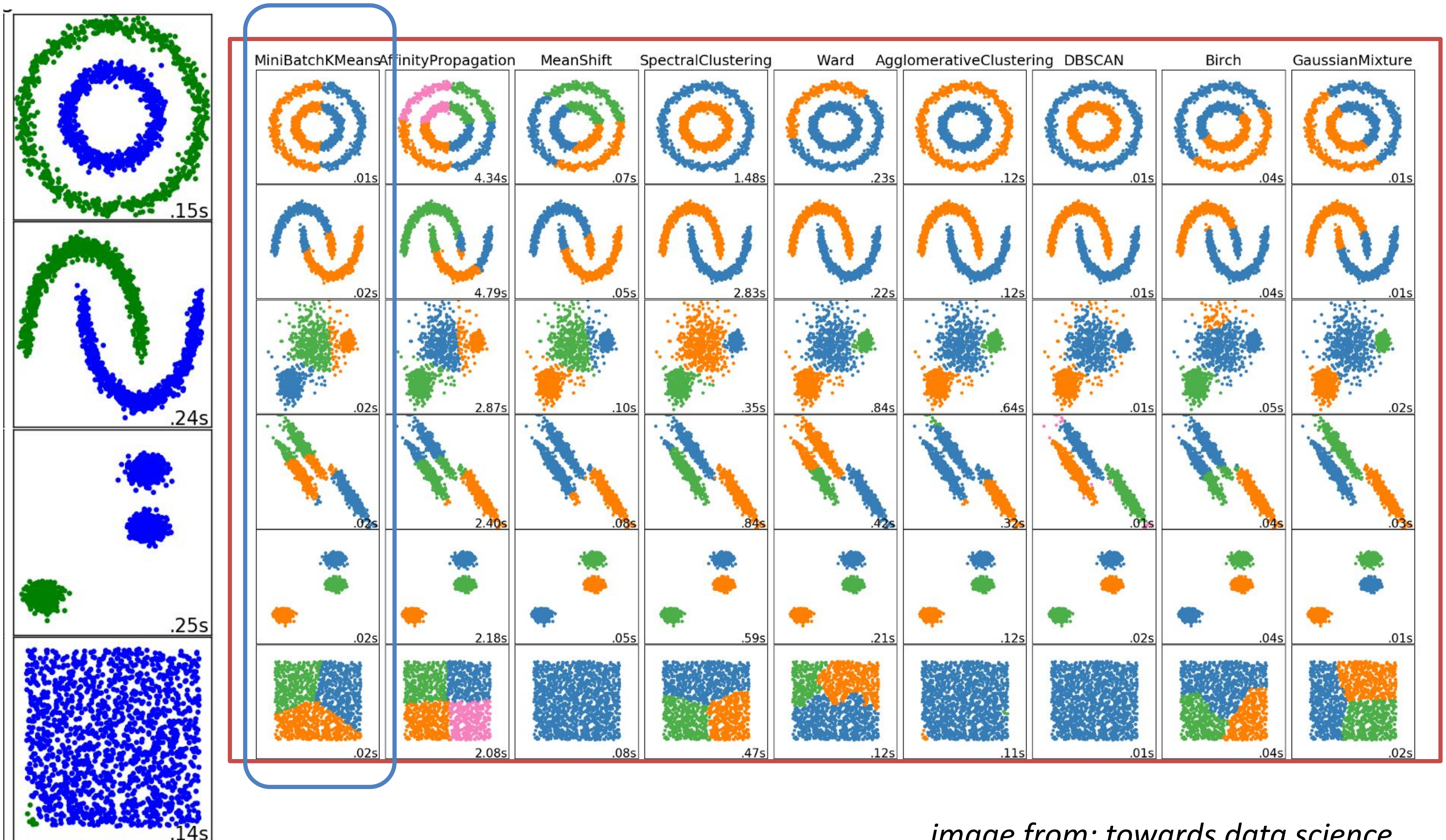


image from: towards data science

Example/Exercise

Draw (approximately) the solution (clusters and centers) found by Lloyd algorithm for the 2 clusters ($K = 2$) problem, when the data ($x_i \in \mathbb{R}$) are the crosses in the figure below and the algorithm is initialised with center values indicated with the circle ($\circ$, cluster 1) and triangle ($\triangle$, cluster 2) shown in the figure.

Linkage-based Clustering

General class of algorithms that follow the general scheme below

1. Start from the trivial clustering: each data sample/point is a (single-point) cluster
2. Until "*termination condition*": repeatedly merge the "*closest*" clusters of the previous clustering

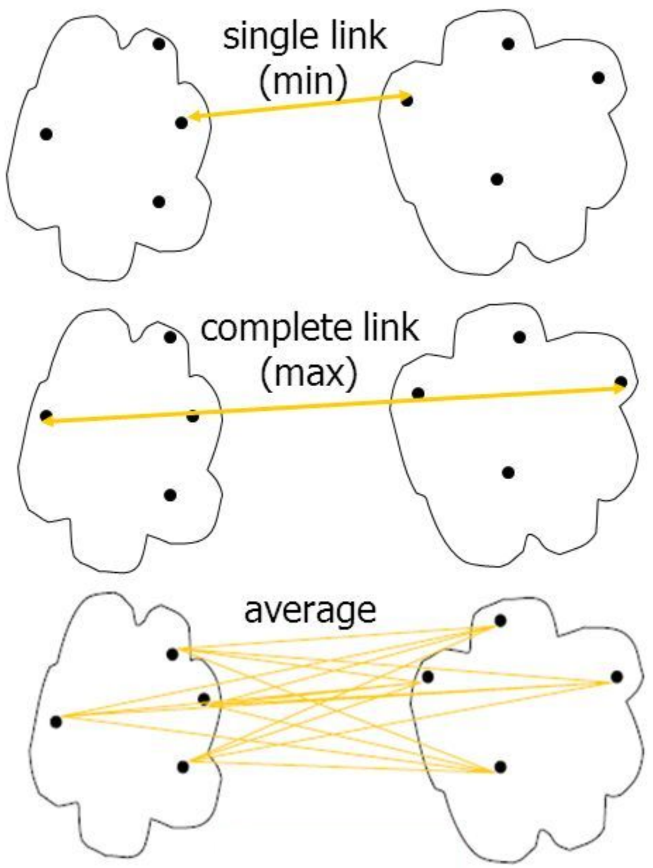
Two "*parameters*":

1. How to define distance *D(A,B)* between two clusters *A* and *B*
   - *Need cluster-to-cluster distance (not point-to-point)*
2. Termination condition

Linkage-based Clustering

Different distances $D(A, B)$ between two clusters A and B can be used, resulting into different linkage methods:

- **single linkage**: $D(A, B) = \min\{d(\mathbf{x}, \mathbf{x}') : \mathbf{x} \in A, \mathbf{x}' \in B\}$
- **average linkage**: $D(A, B) = \frac{1}{|A||B|} \sum_{\mathbf{x} \in A, \mathbf{x}' \in B} d(\mathbf{x}, \mathbf{x}')$
- **max linkage**: $D(A, B) = \max\{d(\mathbf{x}, \mathbf{x}') : \mathbf{x} \in A, \mathbf{x}' \in B\}$



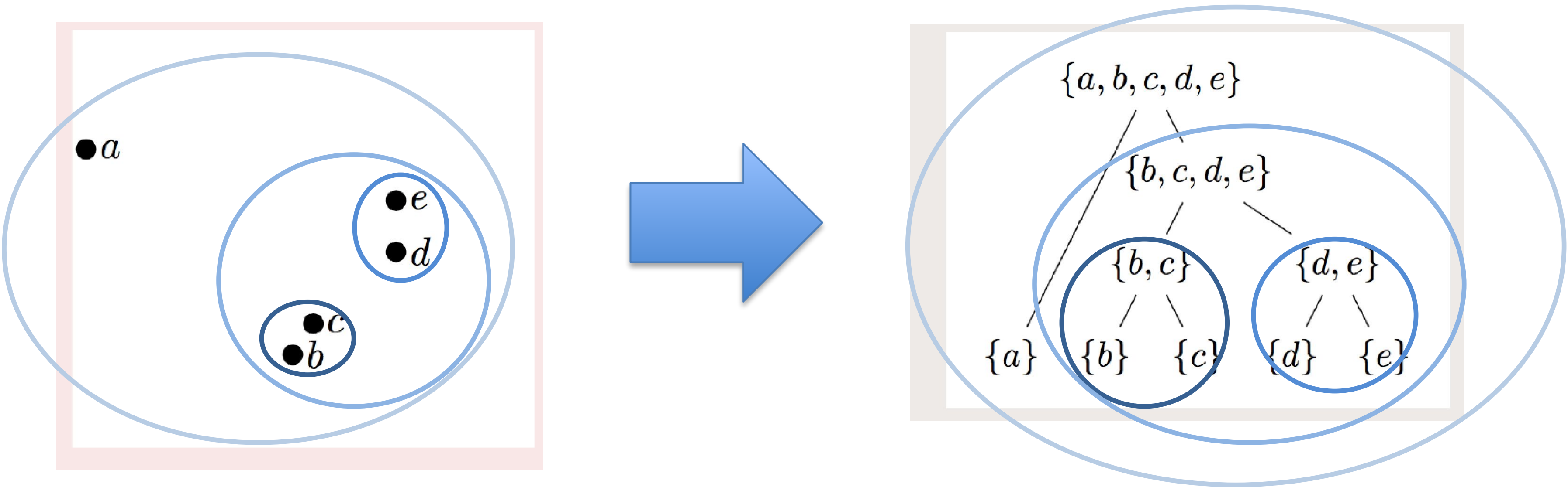
Common termination condition:

- data points are partitioned into k clusters
- minimum distance between pairs of clusters is $> r$, where r is a parameter provided in input
- all points are in a cluster $\Rightarrow$ output is a dendrogram

See next slide

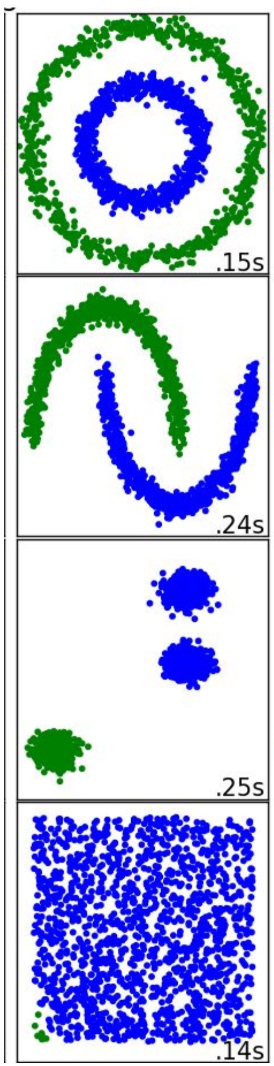
image from Univ. Manchester

Example: Single Linkage



- Single linkage (use minimum distance between points in the cluster)
- End when all points are in a single cluster $\longrightarrow$ output is a dendogram
  - from the dendogram various clusterings can be extracted

Examples



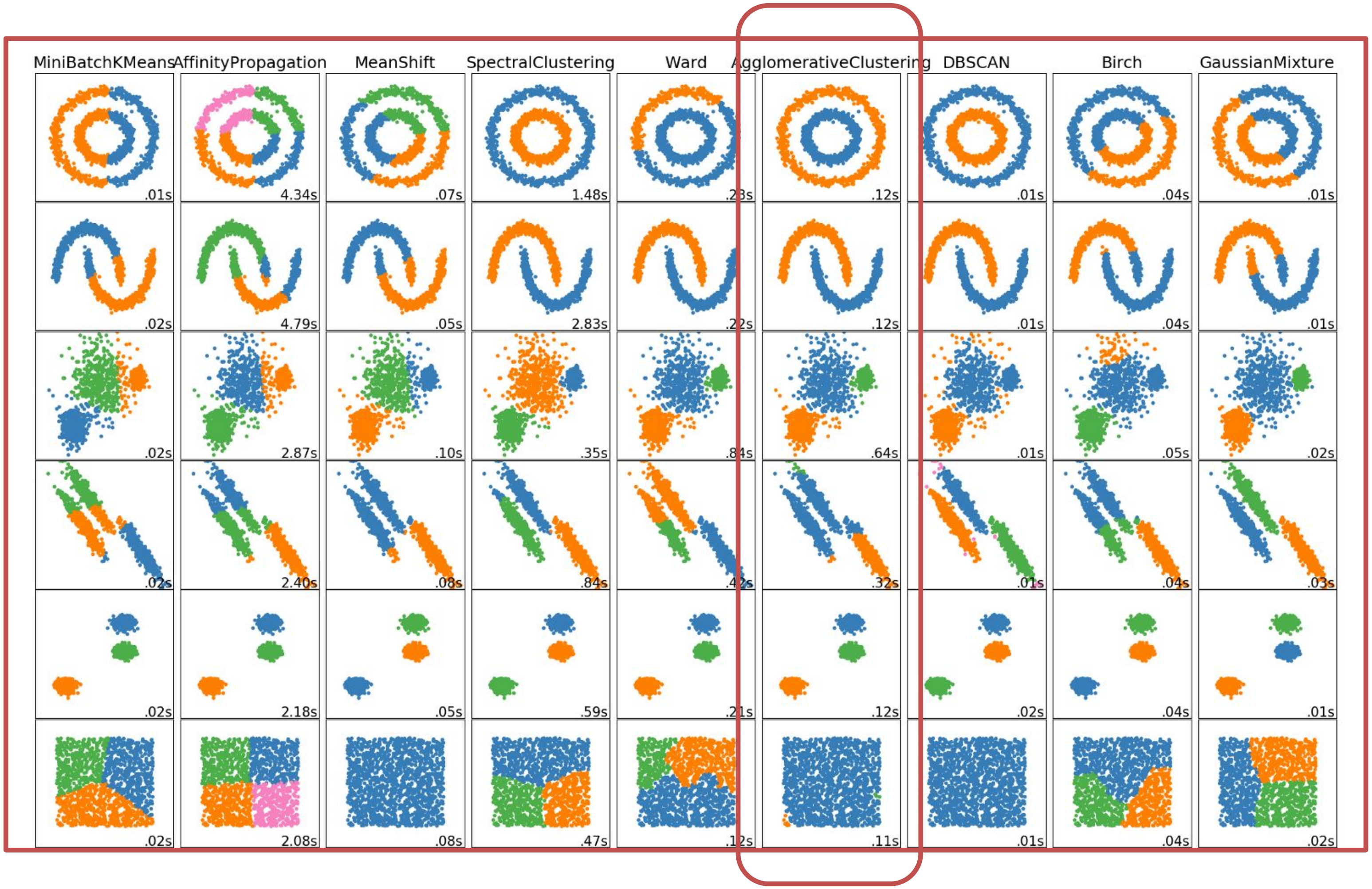


image from: towards data science