Representative-based clustering

Davide Mottin

Data Mining

Learning goals for today

- What is Clustering?
- What characterizes representation-based approaches?
- Main algorithms: k-means, k-medoid, EM

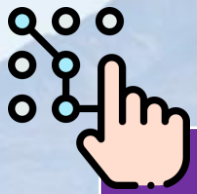
Roadmap

Clustering

- Representative-based
- Density-based
- Hierarchical and Subspace
- Outlier detection

Graph Mining

- Spectral Theory and clustering
- Community Detection
- Link Analysis
- Similarities and Graph Embeddings
- Graph Convolutional Networks



Pattern mining

- Frequent subgraph mining
- Frequent Items and Association Rules
- Sequence Mining
- Similarities and Stream Mining

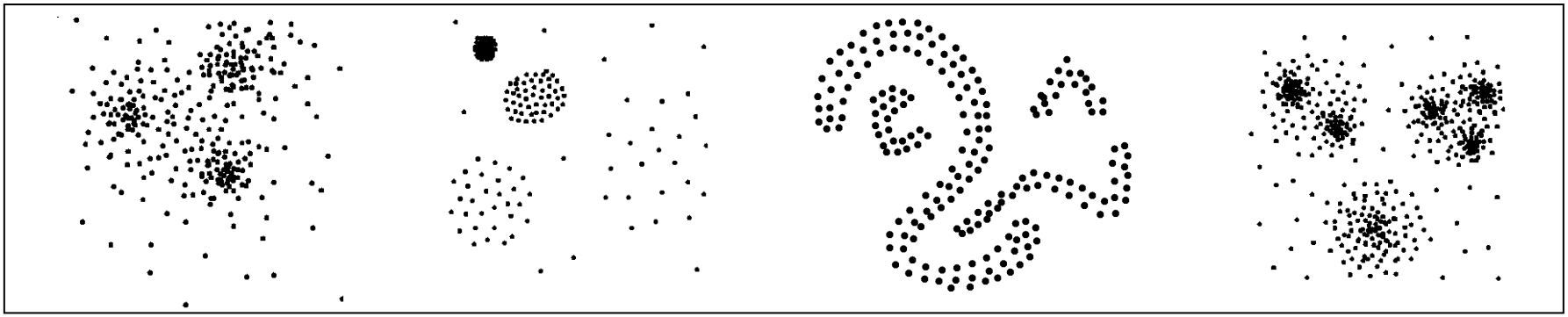
Icons made by Flat Icons, Freepik from www.flaticon.com

Review from last week

- Introduction to Data Mining
  - Unsupervised vs. supervised
- The Knowledge Discovery in Databases process
  - Data Mining as central step from task-relevant clean data to patterns (to be shown to user)
- Overview over main data mining tasks
- Data types
- Measures and notions for data analysis, exploratory analysis

What is Clustering?

- Grouping a set of data objects into clusters
  - Cluster: a collection of data objects
    - Similar to one another within the same cluster
    - Dissimilar to the objects in other clusters
- **Clustering** = unsupervised classification (no predefined classes)
- **Typical usage**
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms



Labelled datasets

Join at menti.com | use code 1941 7577

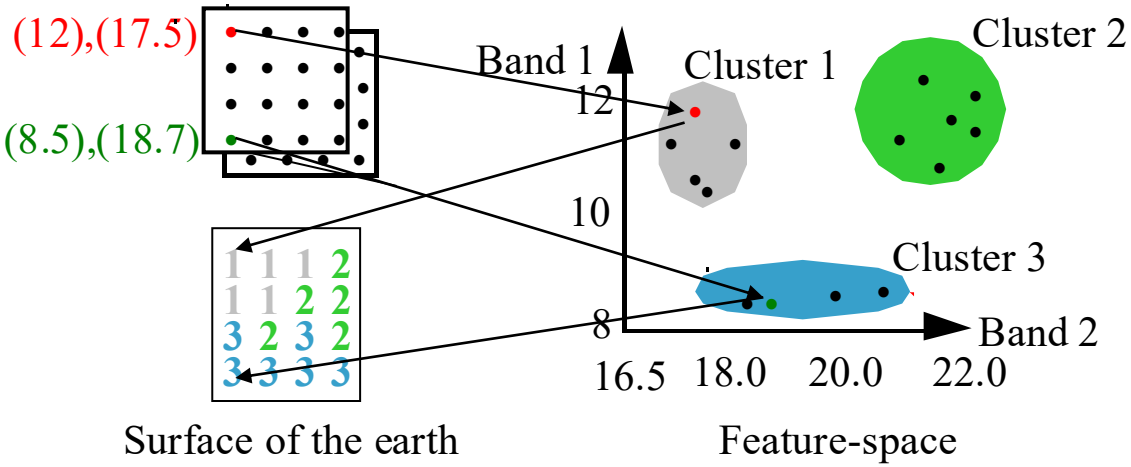
General Applications of Clustering

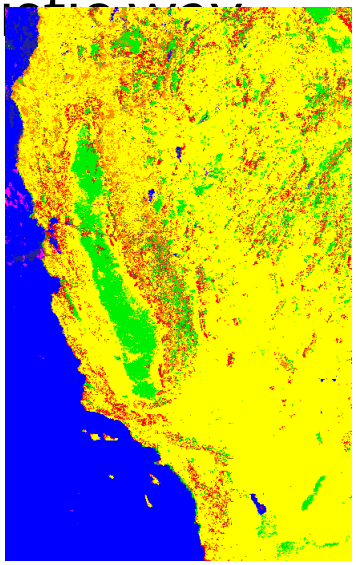
- **Pattern Recognition and Image Processing**
- **Spatial Data Analysis**
  - create thematic maps in GIS (Geographic Information Systems) by clustering feature spaces
  - detect spatial clusters and explain them in spatial data mining
- **WWW**
  - Documents (Web Content Mining)
  - Web-logs (Web Usage Mining)
- **Biology**
  - Clustering of gene expression data
  - taxonomy of living things: kingdom, phylum, class, order, family, genus and species
- **Information retrieval:** document clustering
- **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
- **Social networks**: finding communities of people with the same interests

Spoiler: Community detection in this course

Application #1: Thematic Maps

- Satellite images of a region in different wavelengths
  - Each point on the surface maps to a high-dimensional feature vector $p = (x_1, \ldots, x_d)$ where x_i is the recorded intensity at the surface point in band i.
  - **Assumption**: each different land-use reflects and emits light of different wavelengths in a characteristic way





Application #2: Web Usage Mining

Determine Web User Groups

Sample content from a web-log file

```
romblon.informatik.uni-muenchen.de lopa - [04/Mar/1997:01:44:50 +0100] "GET /~lopa/ HTTP/1.0" 200 1364
romblon.informatik.uni-muenchen.de lopa - [04/Mar/1997:01:45:11 +0100] "GET /~lopa/x/ HTTP/1.0" 200 712
fixer.sega.co.jp unknown - [04/Mar/1997:01:58:49 +0100] "GET /dbs/porada.html HTTP/1.0" 200 1229
scooter.pa-x.dec.com unknown - [04/Mar/1997:02:08:23 +0100] "GET /dbs/kriegel_e.html HTTP/1.0" 200 1241
```

Generation of sessions

⇨ Session::= <IP_address, user_id, [URL_1, . . ., URL_k]>

What's the overall picture in the web log?
Cluster the sessions to find similar web users.

Jaccard: **Proportion of different elements in the two sets**

Distance function for sessions:

$$d(x,y) = \frac{|x \cup y| - |x \cap y|}{|x \cup y|}$$

Major Clustering Approaches

- **Representative-based/Partitioning algorithms** [Today!]
  - Find *k* partitions, minimizing some objective function
  - Expectation Maximization
- **Density-based [week 7]**
  - Find clusters based on connectivity and density functions
- **Hierarchy algorithms [week 8]**
  - Create a hierarchical decomposition of the set of objects
- **Subspace Clustering [week 8]**
- Other methods
  - Grid-based
  - Neural networks (SOM's)
  - Graph-theoretical methods
  - . . .

Representative-based approaches: basic concept

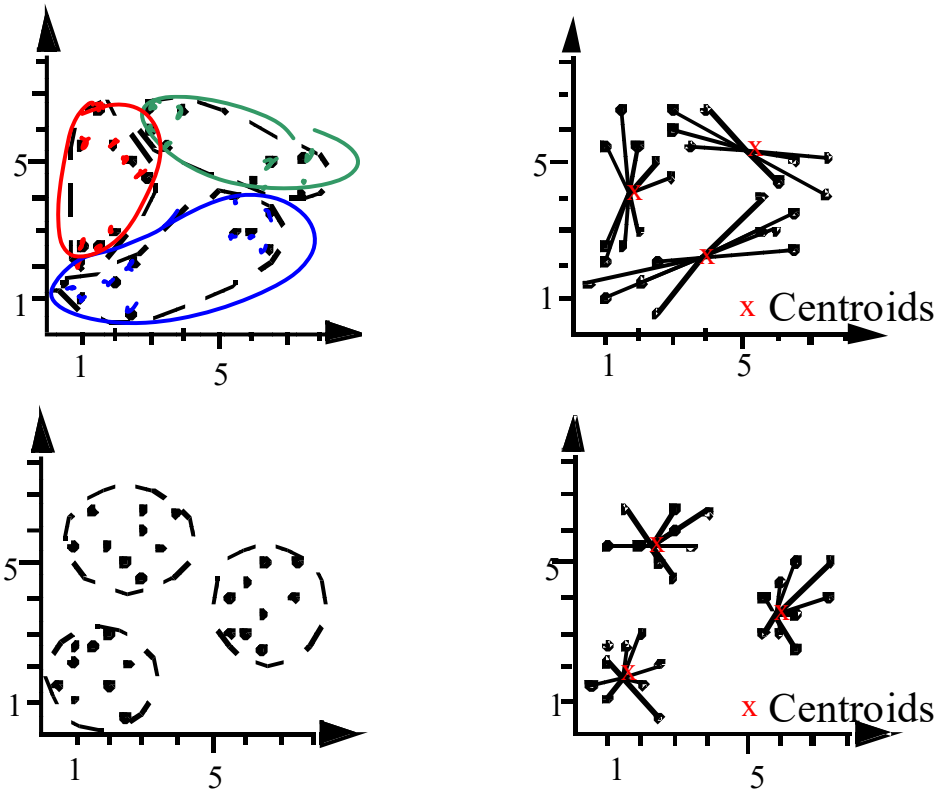
- ***Goal***: Construct a partition of a database *D* of *n* objects into a set of *k* clusters minimizing an objective function
  - Exhaustively enumerating all possible partitions into *k* sets in order to find the global minimum is too expensive.
  - Example objective: minimize the max distance (Manhattan, Euclidian, Geodesic, …) among the points in the cluster.
- **Heuristic methods:**
  - Choose *k* representatives for clusters, e.g., randomly
  - Improve these initial representatives iteratively:
    - Assign each object to the cluster it "fits best" in the current clustering
    - Compute new cluster representatives based on these assignments
    - Repeat until the change in the objective function from one iteration to the next drops below a threshold
- Types of cluster representatives
  - k-means: Each cluster is represented by the center of the cluster
  - k-medoid: Each cluster is represented by one of its objects

K-means clustering

Average is the best

K-Means Clustering: Basic Idea

- **Objective**: For a given *k*, form *k* groups so that the sum of the **squared** distances between the mean of the groups (a.k.a. centroids) and their elements is minimal.
  - Poor Clustering
  - Optimal Clustering



K-Means Clustering: Basic Notions

- **Objects** $\mathbf{x} = (x_1, \ldots, x_d)$ are points in a d-dimensional vector space (the mean of a set of points must be defined)
- ***Centroid*** μ_C: Mean of all points in a cluster C,

$$\mu_C = \frac{1}{|C|} \sum_{\mathbf{x}_j \in C} \mathbf{x}_j$$

- Measure for the **compactness** („Total Distance“) of a **cluster** C_i:

$$TD(C_i) = \sqrt{\sum_{\mathbf{x}_j \in C_i} dist(\mathbf{x}_j, \mu_{C_i})^2}$$

- Measure for the **compactness** of a **clustering**

$$TD = \sqrt{\sum_{i=1}^{k} TD^2(C_i)}$$

The ideal clustering **minimizes** this objective function

K-Means Clustering: Lloyd's Algorithm

- Given k, **Lloyd's algorithm** performs 2 steps:

  Init: Partition the objects into k nonempty subsets

  1. **Centroid Update**: Compute the centroids of the clusters of the current partition. The centroid is the center (mean point) of the cluster.
  2. **Cluster Assignment**: Assign each object to the cluster with the nearest representative.

  - Repeat Step 1 **until representatives change** (substantially).

Pseudocode k-means

K-MEANS $(\mathbf{D}, k, \epsilon)$:

1 $t = 0$

2 Randomly initialize k centroids: $\boldsymbol{\mu}_1^t, \boldsymbol{\mu}_2^t, \ldots, \boldsymbol{\mu}_k^t \in \mathbb{R}^d$

3 **repeat**

4 $\quad t \leftarrow t + 1$

5 $\quad C_j \leftarrow \emptyset$ for all $j = 1, \cdots, k$

$\quad$ `// Cluster Assignment Step`

6 $\quad$ **foreach** $\mathbf{x}_j \in \mathbf{D}$ **do**

7 $\quad\quad j^* \leftarrow \arg\min_i \left\{ \left\| \mathbf{x}_j - \boldsymbol{\mu}_i^t \right\|^2 \right\}$ `// Assign` $\mathbf{x}_j$ `to closest centroid`

8 $\quad\quad C_{j^*} \leftarrow C_{j^*} \cup \{\mathbf{x}_j\}$

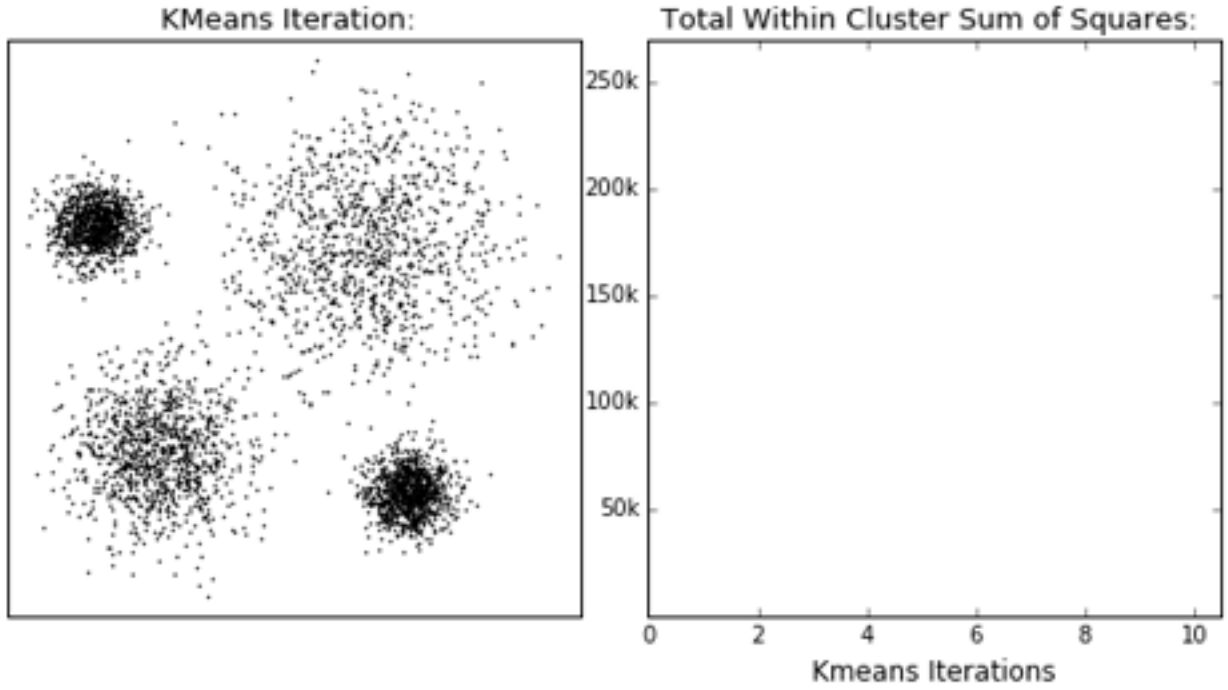
$\quad$ `// Centroid Update Step`

9 $\quad$ **foreach** $i = 1$ *to* k **do**

10 $\quad\quad \boldsymbol{\mu}_i^t \leftarrow \frac{1}{|C_i|} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$

11 **until** $\sum_{i=1}^{k} \left\| \boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1} \right\|^2 \leq \epsilon$

K-means example



Normalization

Join at menti.com | use code 1941 7577

Our example dataset

Iris setosa

Iris versicolor

Iris virginica

Four attributes: Sepal length, Sepal width, Petal length, Petal width

Selecting two attributes for visualization using PCA (two principal components).

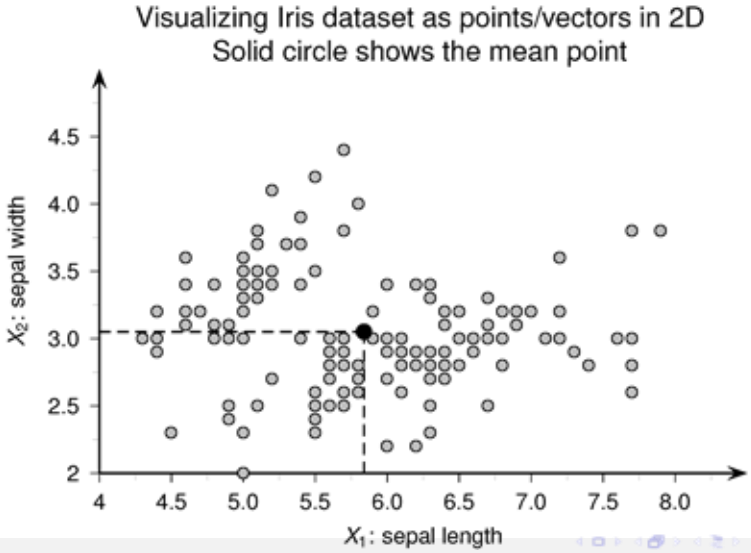
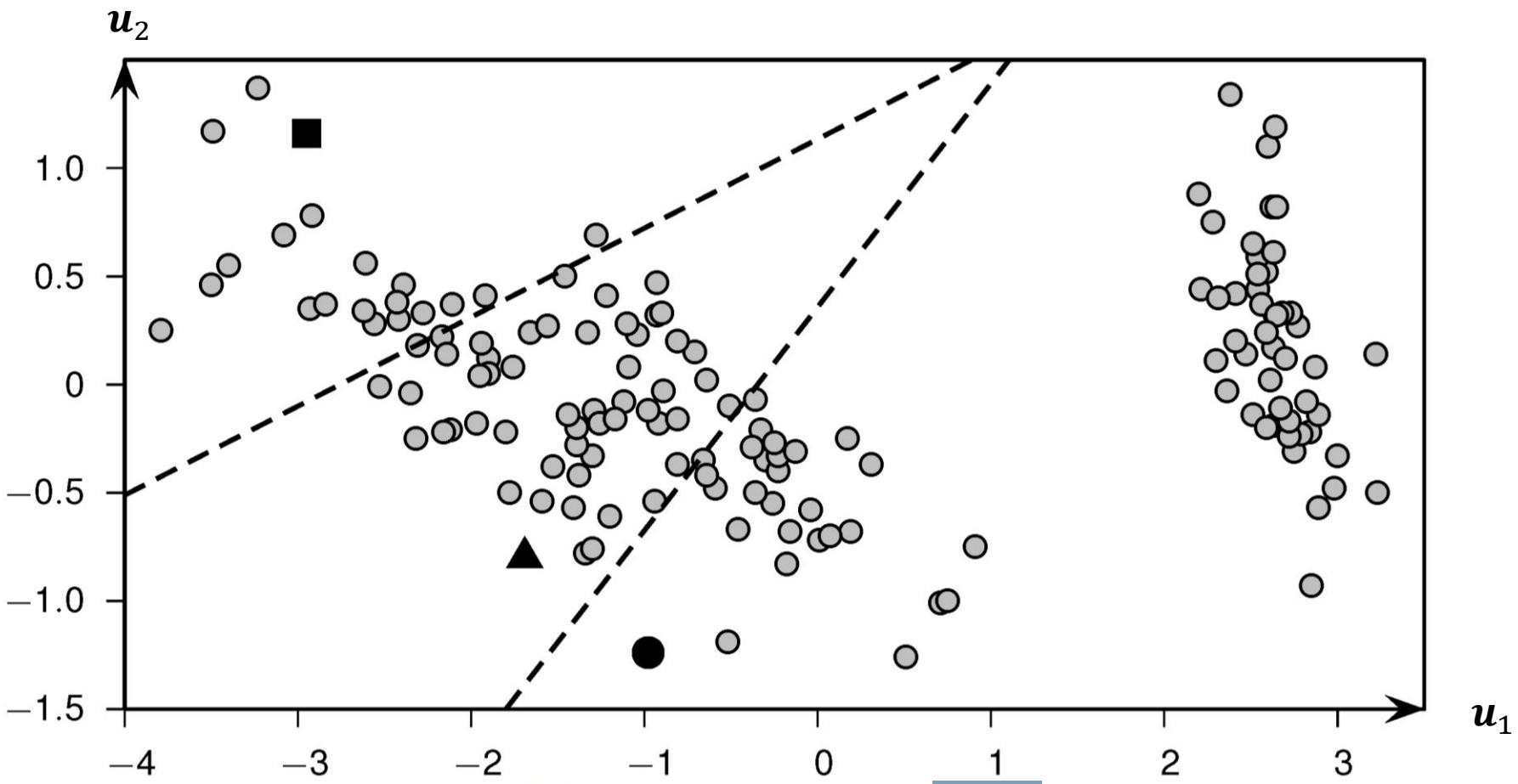


Image sources: Wikipedia

K-Means example

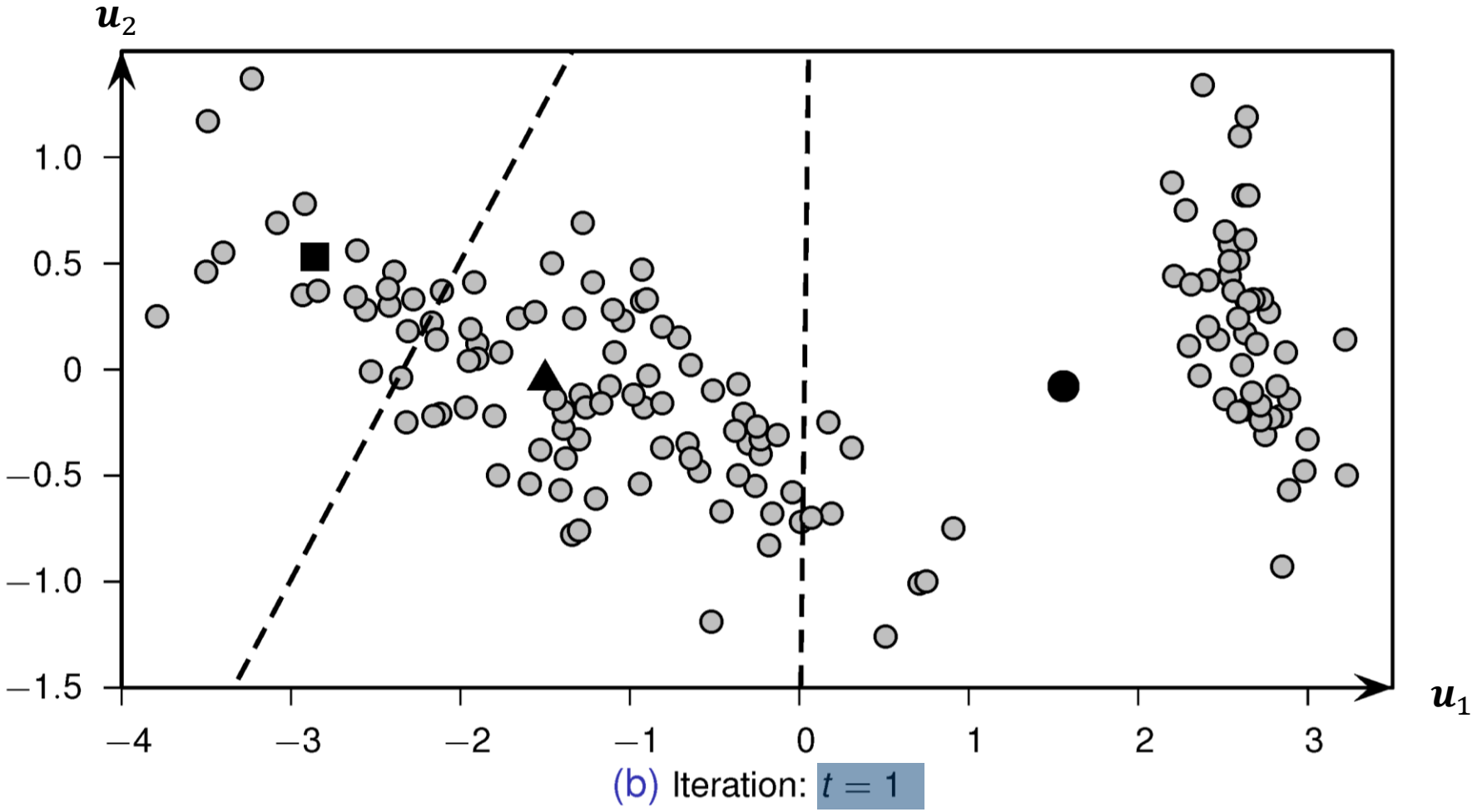
- $k = 3 \Rightarrow$ Find 3 clusters



(a) Random initialization: $t = 0$

K-Means example

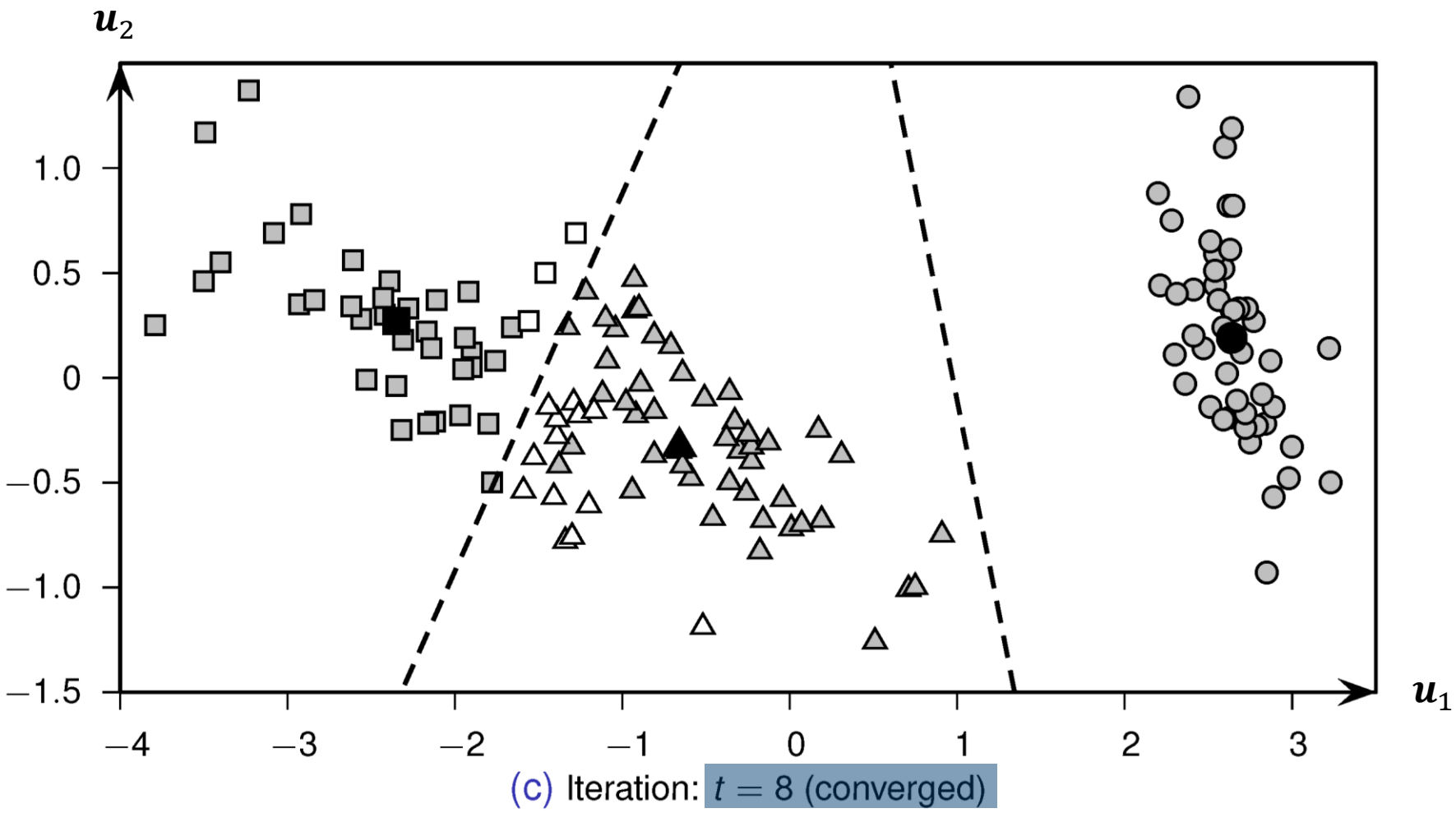
- $k = 3 \Rightarrow$ Find 3 clusters



(b) Iteration: $t = 1$

K-Means example

- $k = 3 \Rightarrow$ Find 3 clusters



K-Means - Discussion

Advantages

- Relatively efficient: $\mathcal{O}(tkn)$, where n is #objects, k is #clusters, and t is #iterations
- Normally: $k, t \ll n$
- Easy implementation

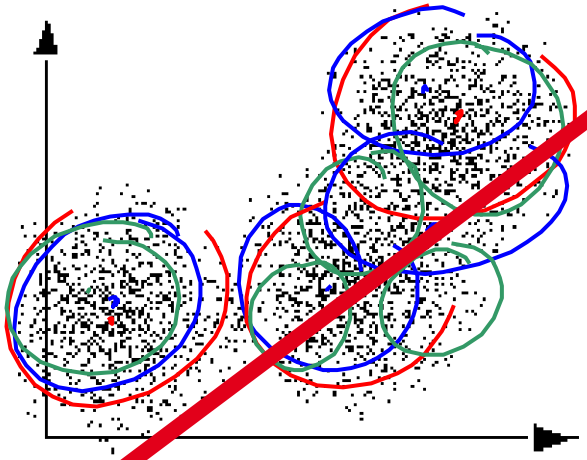
Disadvantages

- Applicable only when mean is defined
- Need to specify k, the number of clusters, in advance
- Sensitive to noisy data and outliers
- Clusters are forced to have convex shapes
- Result and runtime are very **dependent on the initial partition**; often terminates at a *local optimum*
  - **however: methods for good initialization exist**

- **Several variants** of the *k*-means method exist, e.g. ISODATA
  - Extends *k*-means by methods to eliminate very small clusters, merging and split of clusters; user has to specify additional parameters

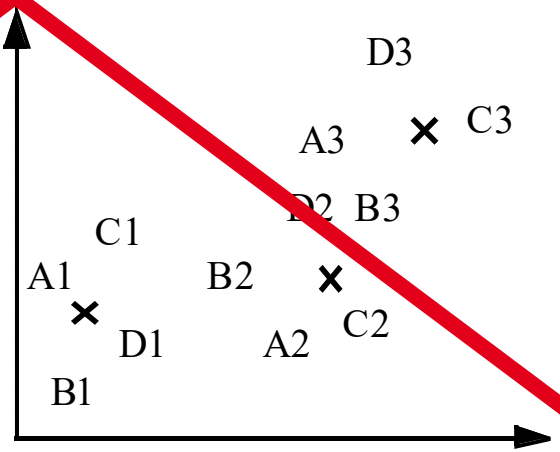
Initialization of Partitioning Clustering Methods

1. **Draw** m different (small) samples $A, B, \ldots, M$ of the dataset
2. **Cluster** each sample to get m estimates to compute k representatives (centroids)
   $A = (A_1, A_2, \ldots, A_k), B = (B_1, \ldots, B_k), \ldots, M = (M_1, \ldots, M_k)$
3. **Cluster** $A \cup B \cup \ldots \cup M$ m times, using $A, B, \ldots, M$ as respective initial partitioning
4. **Use the best** of these m clusterings as initialization for the partitioning clustering of the whole dataset



dataset

$k = 3$



DS

$m = 4$ samples A, B, C, D

✕ true cluster centers

[Bradley and Fayyad 1998]

K-means++

This provides a good approximation (not in this course)

- Tries to prevent arbitrarily bad local minima?
- **Reweight** the points based on the distance to the current representative centers
- Let the shorted distance from a data point x to the closest centers already chosen

$$\mathrm{D}(\mathrm{x}) = \min_{c \in C} \|\mathrm{x} - c\|^2$$

Algorithm

1. Randomly choose first center c_1.
2. Pick new center $\mathrm{x} \in D$ with prob. proportional to

$$\frac{D(x)}{\sum_{x' \in X} D(x)}$$

   - Basically we want to find as good of an initialization as possible

1. Add x to C
2. Repeat until K centers.

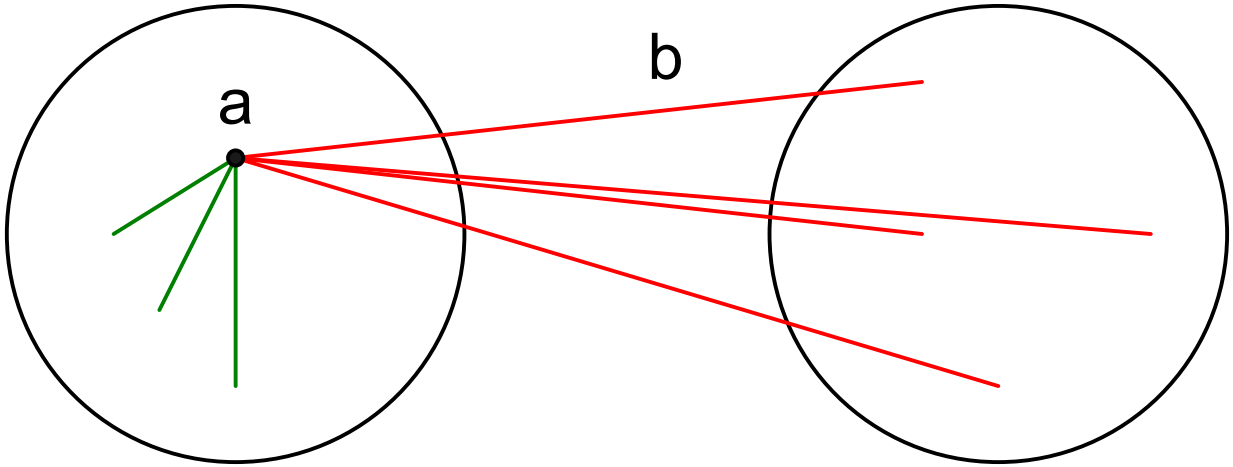
[Arthur and Vassilvitskii, 2006]

Choice of the Parameter k

- Idea for a method:
  - Determine a clustering for each $k = 2, \ldots, n-1$
  - Choose the *best* clustering
- **But how can we measure the quality of a clustering?**
  - A measure has to be independent of k.
  - The measures for the compactness of a clustering TD^2 and TD are monotonously decreasing with increasing value of k.
- Silhouette-Coefficient [Rousseeuw 1987]
  - Measure for the quality of a k-means or a k-medoid clustering that is independent of k.

The silhouette coefficient

- **How good is the clustering?**
  - how appropriate is the mapping of objects to clusters
- Elements in cluster should be „*similar*“ to their representative
  - int: measure the average distance of objects to their representative
- Elements in different clusters should be „*dissimilar*“
  - ext: measure the average distance of objects to alternative clusters



The silhouette coefficient, formally

- $int(o)$: average distance between object o and the objects **in its cluster** A

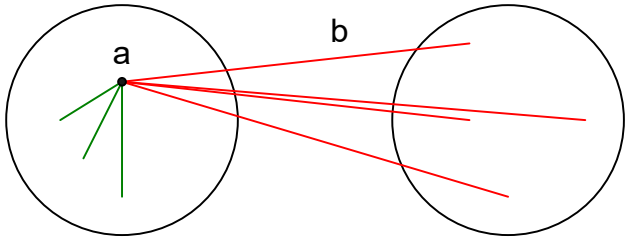
$$int(\mathbf{o}) = \frac{1}{|C_i|} \sum_{\substack{\mathbf{x} \in C(\mathbf{o}) \\ \mathbf{x} \neq \mathbf{o}}} dist(\mathbf{o}, \mathbf{x})$$

- $ext(o)$: average distance between object o and the objects **in its "second closest" cluster**

$$ext(\mathbf{o}) = \min_{C_i \neq C(\mathbf{o})} \left(\frac{1}{|C_i|} \sum_{p \in C_i} dist(\mathbf{o}, \mathbf{x}) \right)$$

- The silhouette of o is

$$s(o) = \frac{ext(o) - int(o)}{\max\{int(o), ext(o)\}}$$



- The values of the silhouette coefficient range from −1 to +1

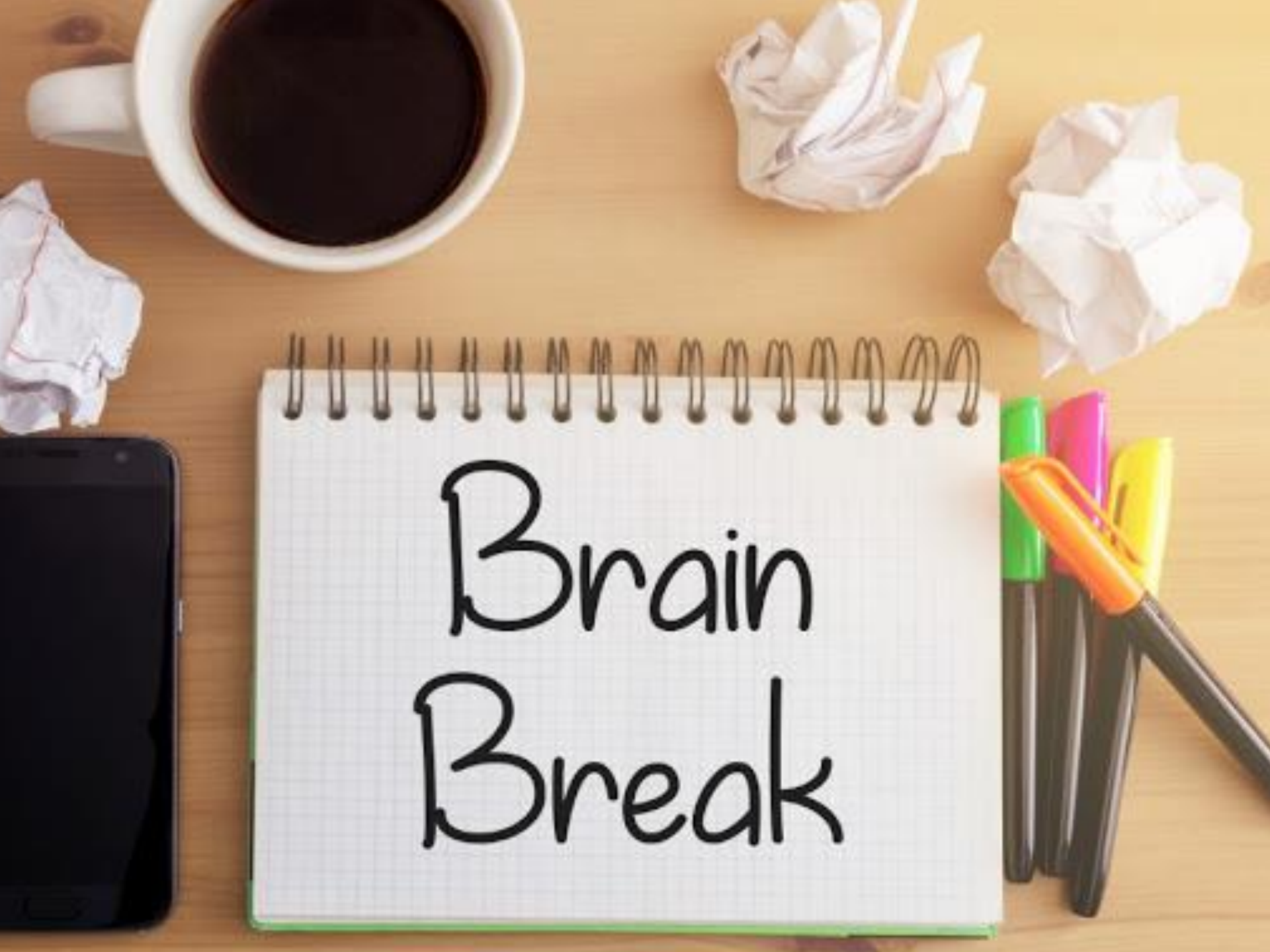
Silhouette coefficient

Join at menti.com | use code 1941 7577

"Reading" the silhouette coefficient

- **How good is the assignment of *o* to its cluster?**
  - $s(o) = -1$: bad, on average closer to members of *B*
  - $s(o) = 0$: between *A* and *B*
  - $s(o) = +1$: good assignment of *o* to its cluster *A*

- **Silhouette Coefficient s_C** of a clustering is **average silhouette of all objects**
  - $0.7 < s_C \leq 1.0$ **strong** structure,
  - $0.5 < s_C \leq 0.7$ **medium** structure
  - $0.25 < s_C \leq 0.5$ **weak** structure,
  - $s_C \leq 0.25$ **no** structure
- **Absolute** values of S_C of limited value → difficult to compare across data and algorithms
- **Used for:** Determining the best value of k

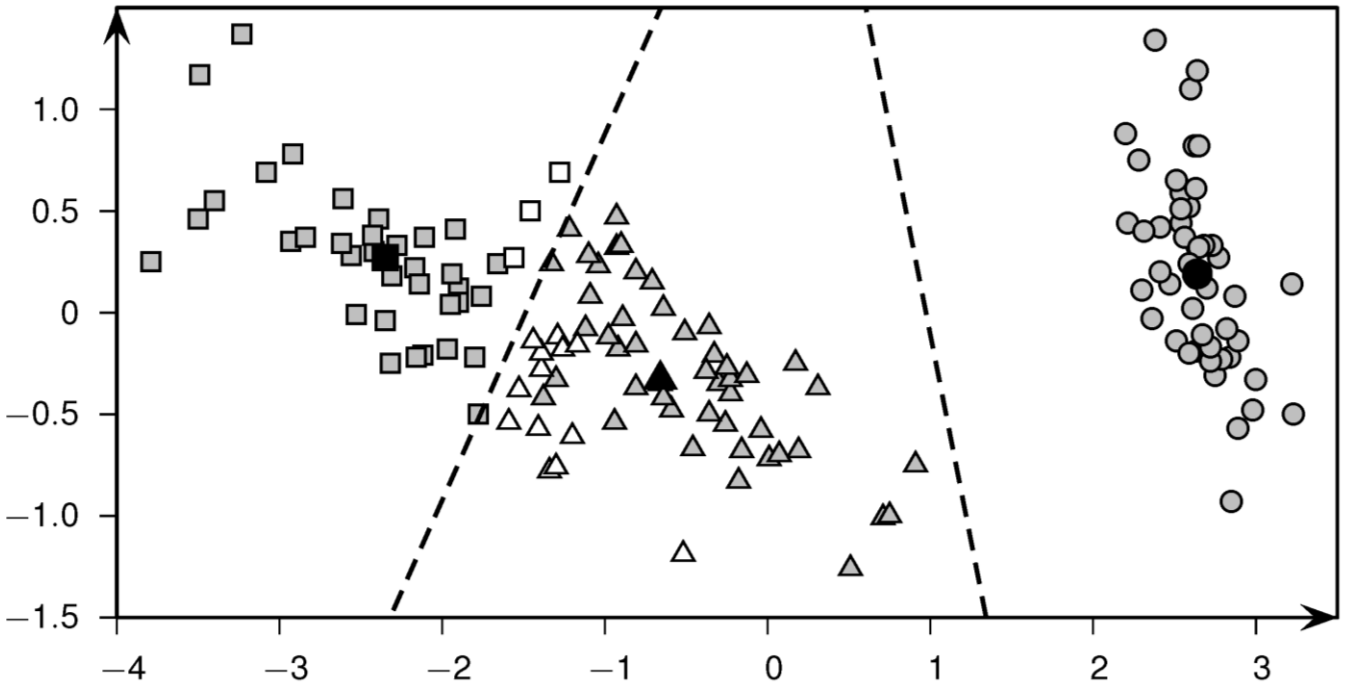

Brain
Break

Kernel k-means

… why Euclidean space?

Linearity of k-means

Recall: K-means assume the boundaries between the clusters are lines ☹



What if the boundaries are NOT lines?

Kernel k-means

- **Idea:** Use **kernels** to "project" points into another space
- Separate the points with –means in the new space
- Recall k-means objective

$$\min_C \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \|\mathbf{x}_j - \mu_i\|^2$$

- Map every point $\mathbf{x}_j$ into a different space $\phi(\mathbf{x}_j)$
- Recall that a **kernel is a dot-product** $\mathrm{K}(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i})^\top \phi(\mathbf{x_j})$

$$\min_C \sum_{i=1}^{k} \sum_{x_j \in C_i} \|\phi(\mathbf{x}_j) - \phi(\mu_i)\|^2 \approx \sum_{j=1}^{n} K(\mathbf{x}_j, \mathbf{x}_j) - \sum_{i=1}^{k} \frac{1}{n_i} \sum_{\mathbf{x}_a \in C_i} \sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b)$$

Expanded form for centroid $\sum\|\mu_i\|^2$

Only expressed in terms of kernels!

Kernel K-means: Cluster reassignment

- Cluster distance

$$\|\phi(x_j) - \phi(\mu_i)\|^2$$
$$= \|\phi(x_j)\|^2 - 2\phi(x_j)^T\phi(\mu_i) + \|\phi(\mu_i)\|^2$$
$$= K(x_j, x_j) - \frac{2}{n_i}\sum_{(x_a \in C_i)} K(x_a, x_j) + \frac{1}{n_i^2}\sum_{x_a \in C_i}\sum_{x_b \in C_i} K(x_a, x_b)$$

Constant!

- Cluster assignment

$$C^*(x_j) = \arg\min_i \left\{ \|\phi(x_j) - \phi(\mu_i)\|^2 \right\}$$

Algorithm

KERNEL-KMEANS$(\mathbf{K}, k, \epsilon)$:

1 $t \leftarrow 0$

2 $\mathcal{C}^t \leftarrow \{C_1^t, \ldots, C_k^t\}$ // Randomly partition points into k clusters

3 **repeat**

4 $\quad t \leftarrow t + 1$

5 $\quad$ **foreach** $C_i \in \mathcal{C}^{t-1}$ **do** // Compute squared norm of cluster means

6 $\quad\quad \text{sqnorm}_i \leftarrow \frac{1}{n_i^2} \sum_{\mathbf{x}_a \in C_i} \sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b)$

Centroid computation

7 $\quad$ **foreach** $\mathbf{x}_j \in \mathbf{D}$ **do** // Average kernel value for $\mathbf{x}_j$ and C_i

8 $\quad\quad$ **foreach** $C_i \in \mathcal{C}^{t-1}$ **do**

9 $\quad\quad\quad \text{avg}_{ji} \leftarrow \frac{1}{n_i} \sum_{\mathbf{x}_a \in C_i} K(\mathbf{x}_a, \mathbf{x}_j)$

$\quad$ // Find closest cluster for each point

10 $\quad$ **foreach** $\mathbf{x}_j \in \mathbf{D}$ **do**

11 $\quad\quad$ **foreach** $C_i \in \mathcal{C}^{t-1}$ **do**

12 $\quad\quad\quad d(\mathbf{x}_j, C_i) \leftarrow \text{sqnorm}_i - 2 \cdot \text{avg}_{ji}$

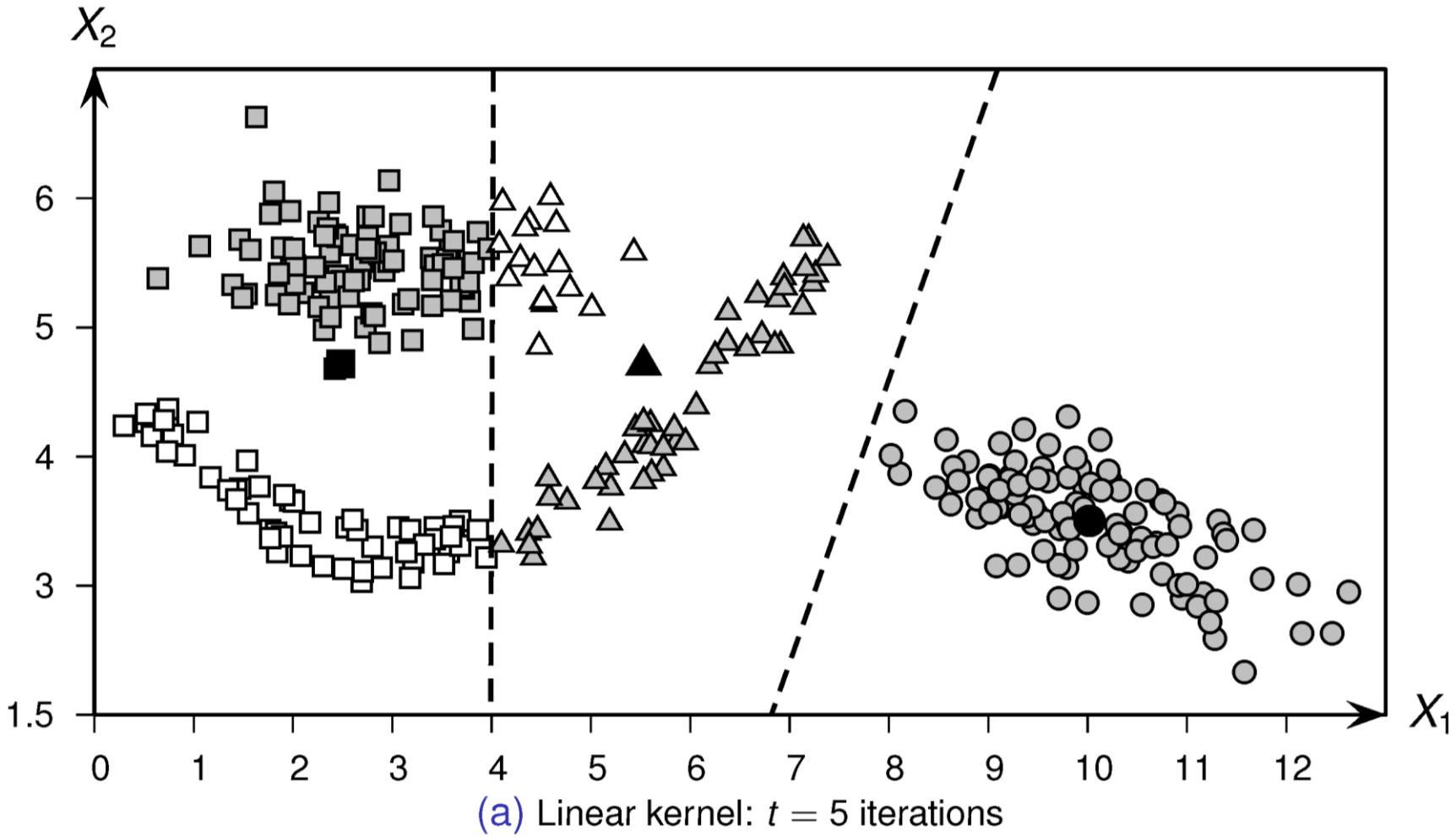
13 $\quad\quad j^* \leftarrow \arg\min_i \{d(\mathbf{x}_j, C_i)\}$

14 $\quad\quad C_{j^*}^t \leftarrow C_{j^*}^t \cup \{\mathbf{x}_j\}$ // Cluster reassignment

Assignment phase

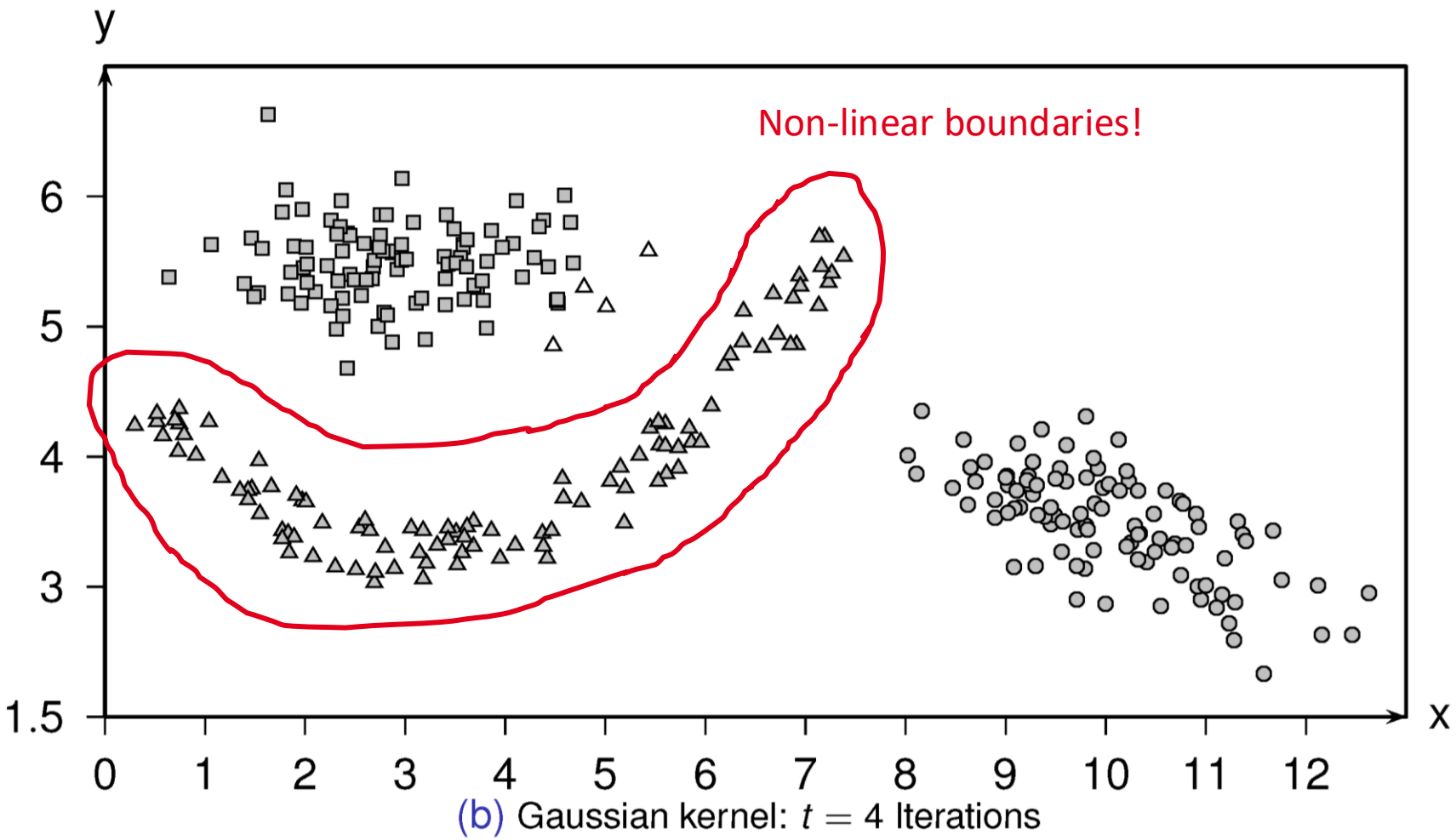
15 $\quad \mathcal{C}^t \leftarrow \{C_1^t, \ldots, C_k^t\}$

K-means or kernel K-means with linear kernel



(a) Linear kernel: $t = 5$ iterations

Kernel k-means: Gaussian kernel



Kernel K-Means: Discussion

Advantages

- Allows detection of clusters with arbitrary shape
- Fits any possible kernel

Disadvantages

- Inefficient (requires the computation of a square kernel matrix)
- Like *k*-means: need to specify the number of clusters *k* in advance, and clusters are forced to have convex shapes
- Requires the choice of a kernel

d = # iterations
k = # clusters
n = # objects

No. of Objects (n)	No. of operations	
	K-means	Kernel K-means
	$O(nkd)$	$O(n^2k)$
1M	10^{11} (3.2)	10^{15}
10M	10^{12} (34.9)	10^{17}
100M	10^{13}(5508.5)	10^{19}
1B	10^{14}	10^{21}

d = 100; k=10

No. of Objects (n)	No. of operations	
	K-means	Kernel K-means
	$O(nkd)$	$O(n^2k)$
1M	10^{13} (6412.9)	10^{16}
10M	10^{14}	10^{18}
100M	10^{15}	10^{20}
1B	10^{16}	10^{22}

d = 10,000; k=10

k-medoid

No fake-points added

K-medoid motivation

- k-means assumes **Euclidean distance**
  - Minimizing distance to mean

$$TD(C_j) = \sqrt{\sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mu_j)^2}$$

  - Variations exist for other distance functions
- **K-medoid more general**
  - Motivated by L_1 norm (Manhattan distance)
  - Works also in spaces, where a mean is not defined
  - Only input is the possibility to compute pairwise distances (”compare”)
- More robust to noise 

Recall L_p norms

- For standardized numerical attributes, i.e., vectors $\mathbf{x} = (x_1, \ldots, x_d)$ and $\mathbf{y} = (y_1, \ldots, y_d)$ from a d-dimensional vector space:
  - General L_p-Metric (Minkowski-Distance)

$$d_p(\mathbf{x}, \mathbf{y}) = \sqrt[p]{\sum_{i=1}^{d} |x_i - y_i|^p}$$

  - $p = 2$: Euclidean Distance (used in k-means)

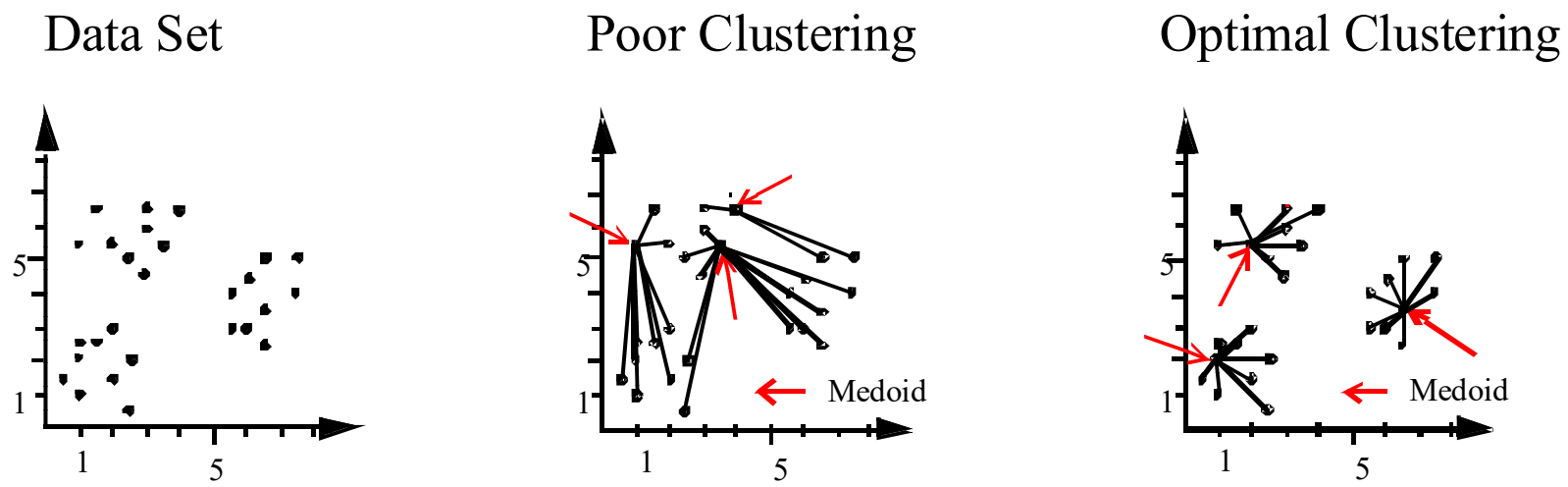
$$d_p(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} |x_i - y_i|^2}$$

  - $p = 1$: Manhattan-Distance (used in k-medoids)

$$d_p(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} |x_i - y_i|$$

k-Medoid Clustering: Basic Idea

- **Objective**:
  - Find *k* representatives in the dataset so that, the sum of the distances between representatives and objects which are assigned to them is minimal.
- **Medoid**: representative object "in the middle" (cf. median)



k-Medoid Clustering: Basic Notions

- Requires arbitrary objects and a distance function
- ***Medoid*** m_C: representative object in a cluster C
- Measure for the **compactness of a cluster** C:

$$TD(C) = \sum_{\mathbf{x} \in C} dist(\mathbf{x}, m_C)$$

- Measure for the **compactness of a clustering**

$$TD(C) = \sum_{i=1}^{k} TD(C_i)$$

K-Medoid Clustering: PAM Algorithm

Partitioning Around Medoids [Kaufman and Rousseeuw, 1990]

- Given *k*, the *k*-medoid algorithm is:
  1. **Select** *k* objects arbitrarily as medoids (representatives); assign each remaining (non-medoid) object to the cluster with the nearest representative, and compute $TD_{current}$.
  2. **For each pair** (medoid *M*, non-medoid *N*)
     - compute the value $TD_{N\leftrightarrow M}$, (TD for the partition that results when "swapping" *M* with *N)*
  3. **Select** the non-medoid *N* for which $TD_{N\leftrightarrow M}$ is minimum
  4. **If** $TD_{N\leftrightarrow M}$ is smaller than $TD_{current}$
     - Swap *N* with *M*
     - Set $TD_{current}$:= $TD_{N\leftrightarrow M}$
     - Go back to Step 2
  5. **Else Stop**

K-Medoid Clustering: CLARA and CLARANS

- CLARA [Kaufmann and Rousseeuw,1990]
  - Additional parameter: *numlocal*
  - Draws *numlocal* samples of the data set
  - Applies PAM on each sample
  - Returns the best of these sets of medoids as output
- CLARANS [Ng and Han, 1994)
  - Two additional parameters: *maxneighbor* and *numlocal*
  - At most *maxneighbor* many pairs (medoid *M*, non-medoid *N*) are evaluated in the algorithm.
  - The first pair (*M*, *N*) for which $TD_{N \leftrightarrow M}$ is smaller than $TD_{current}$ is swapped (instead of the pair with the minimal value of $TD_{N \leftrightarrow M}$)
  - Finding the local minimum with this procedure is repeated *numlocal* times.
- Efficiency: runtime(CLARANS) < runtime(CLARA) < runtime(PAM)

CLARANS: Selection of Representatives

CLARANS (objects DB, k, dist, numlocal, maxneighbor)

for r **from** 1 **to** numlocal **do**

Randomly select k objects as medoids

Let i := 0

while i < maxneighbor **do**

Randomly select (Medoid M, Non-medoid N)

Compute *changeOfTD*:= $TD_{N \leftrightarrow M}$ - TD

if *changeOfTD* < 0 **then**

substitute M by N

TD := $TD_{N \leftrightarrow M}$

i := 0

else i:= i + 1

if TD < TD_best **then**

TD_best := TD; Store current medoids

return Medoids

K-Medoid Clustering: Discussion

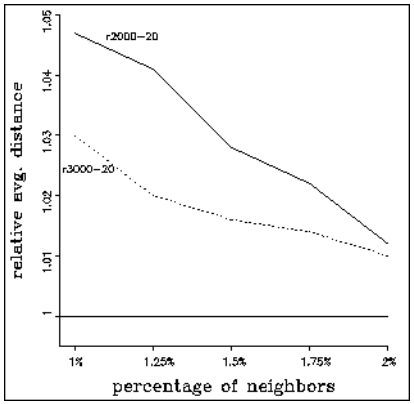
Advantages

- Applicable to arbitrary objects + distance function
- Not as sensitive to noisy data and outliers as *k*-means

Disadvantages

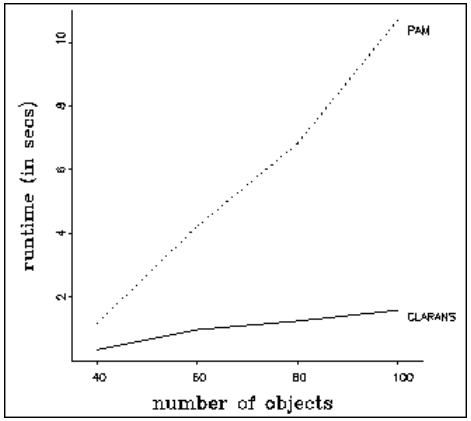
- Inefficient
- Like *k*-means: need to specify the number of clusters *k* in advance, and clusters are forced to have convex shapes
- Result and runtime for CLARA and CLARANS may vary largely due to the randomization

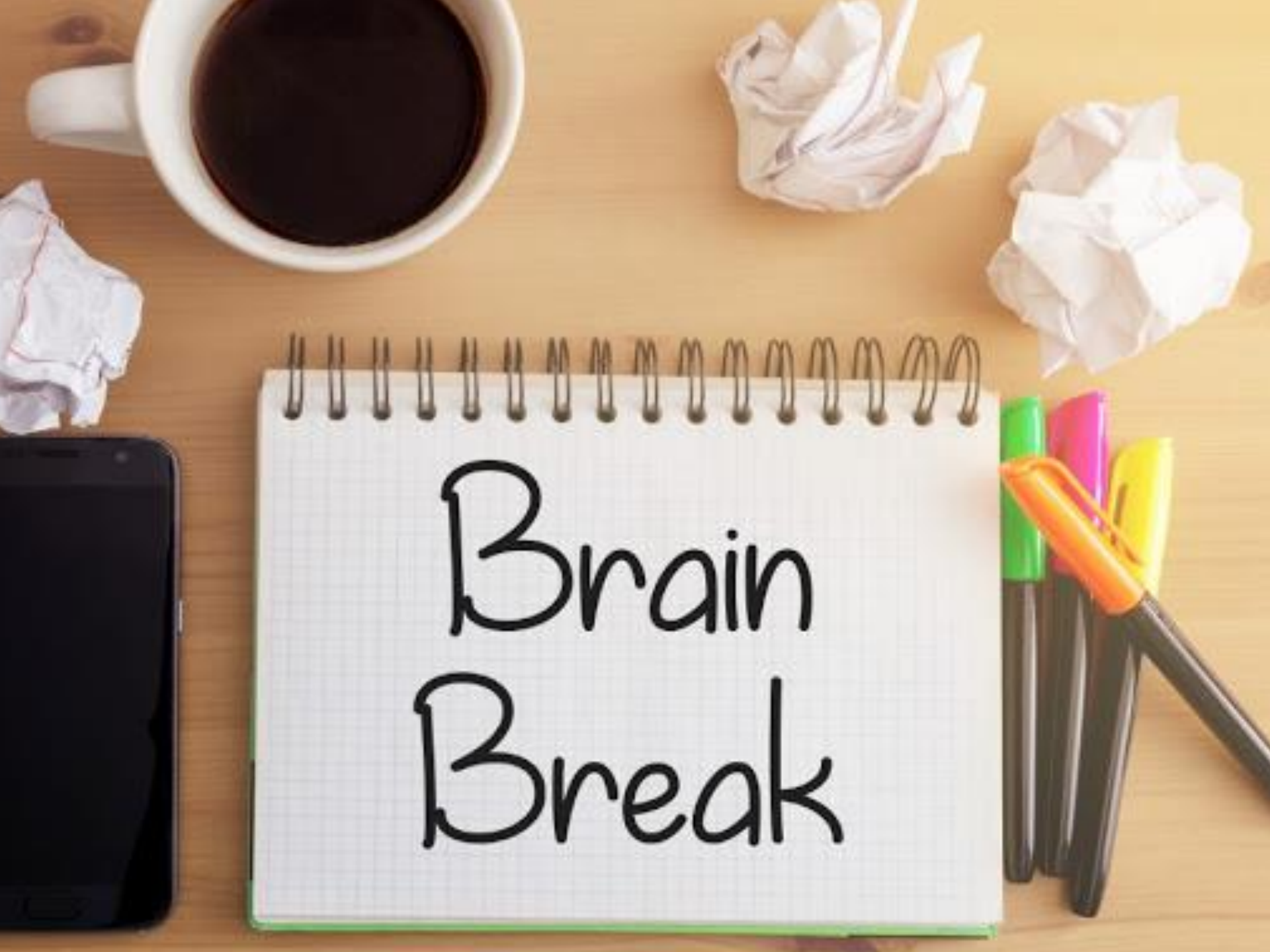
20 rectangular clusters out of
--- 2000 points
- - 3000 points



TD(CLARANS)

TD(PAM)




Brain
Break

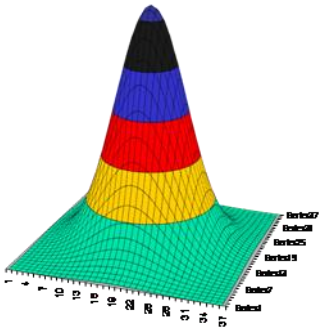
Expectation Maximization

What if cluster overlaps?

Expectation Maximization (EM)

Basic Notions [Dempster, Laird & Rubin 1977]

- Consider points $\mathbf{x}_j = (x_1, \ldots, x_d)$ from a *d*-dimensional Euclidean vector space
- Each cluster is represented by a **probability distribution**
- **Typically:** mixture of Gaussian distributions



Single distribution to represent a cluster C_i

- Center point $\boldsymbol{\mu_i}$ of all points in the cluster
- $d \; x \; d$ Covariance matrix Σ_i for the points in the cluster C_i

- Density function for cluster C_i:

$$f(\mathbf{x} \mid \boldsymbol{\mu_i}, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \cdot e^{-\frac{1}{2} \cdot (\mathbf{x} - \boldsymbol{\mu_i})^T \cdot (\Sigma_i)^{-1} \cdot (\mathbf{x} - \boldsymbol{\mu_i})}$$

Normal distribution
$\mathcal{N}(\mu_C, \Sigma_C)$

Model: Gaussian Mixtures

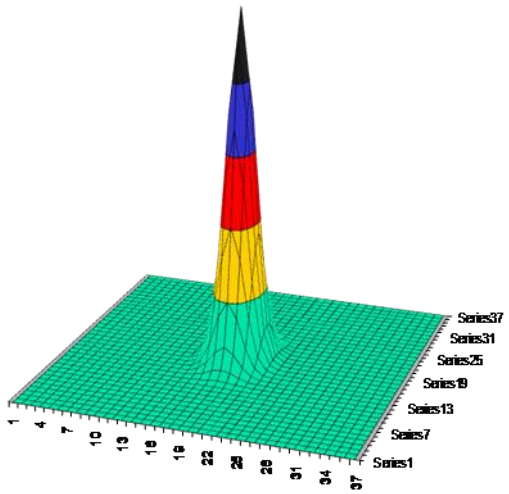
- Having the probability for each cluster, we can define the density for the clustering

$$f(\mathbf{x}) = \sum_{i=1}^{k} f(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) P(C_i)$$

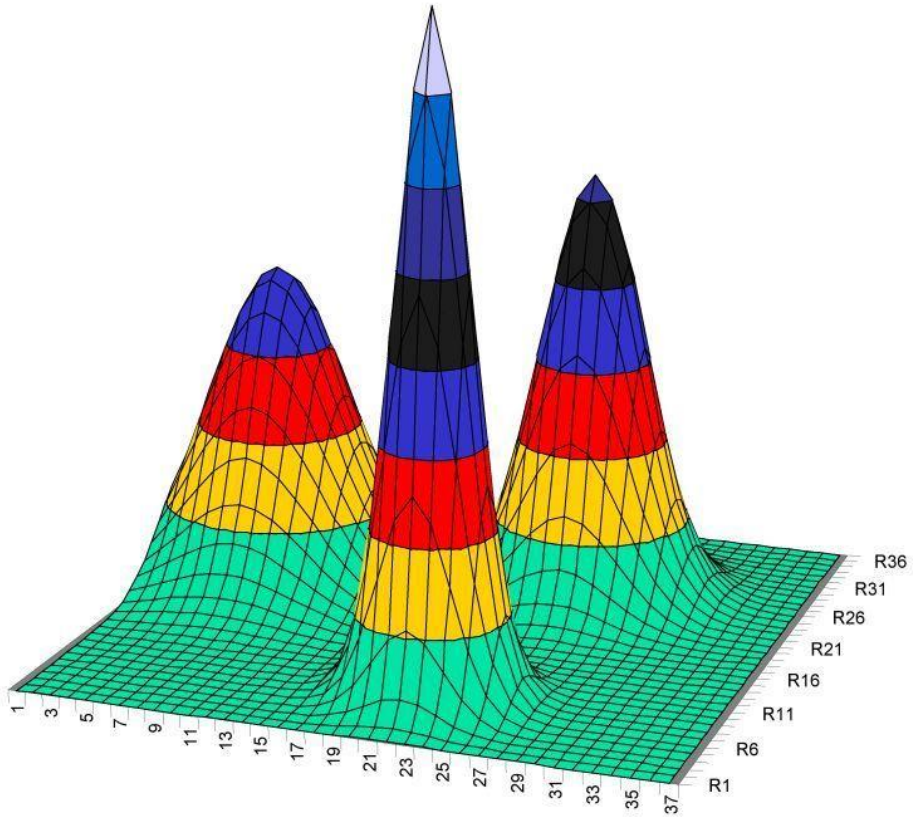
where $\sum_{i=1}^{k} P(C_i) = 1$

EM: Gaussian Mixture – 2D examples

A single Gaussian density function



A Gaussian mixture model, $k = 3$



Gaussian Mixtures

- Having the probability for each cluster, we can define the density for the clustering

$f(\mathbf{x}) = \sum_{i=1}^{k} f(\mathbf{x}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}) P(C_i)$ where $\sum_{i=1}^{k} P(C_i) = 1$

How do we find the parameters $\theta_i = \mu_i, \Sigma_i, P(C_i)$?

Idea: Maximize the log-likelihood (MLE)

$$\arg \max_{(\theta_1, \dots \theta_k)} \ln P(\mathbf{D}|\, \theta_1, \dots \theta_k)$$

Too hard to maximize directly ☹

where $\ln P(\mathbf{D}|\, \theta_1, \dots \theta_k) = \ln \prod_{j=1}^{n} f(\boldsymbol{x}_j) = \sum_{j=1}^{n} \ln f(\mathbf{x}_j)$

MLE for Gaussian Mixtures

$$\arg\max_{(\theta_1,\ldots\theta_k)} \ln P(\mathbf{D}|\,\theta_1,\ldots\theta_k)$$

where

$$\ln P(\mathbf{D}|\,\theta_1,\ldots\theta_k)$$

$$= \ln \prod_{j=1}^{n} f(\boldsymbol{x}_j)$$

$$= \sum_{j=1}^{n} \ln f(\mathbf{x}_j)$$

$$= \sum_{j=1}^{n} ln \sum_{i=1}^{k} f(\mathbf{x_j}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}) P(C_i)$$

Hard! Depends on unknown parameters $\mu, \Sigma, \mathrm{P(C)}$ and the data ☹

EM algorithm: The recipe

1. Assume you have already computed the parameters $\theta_1, \ldots, \theta_k$

2. Compute the probability of each cluster conditioned to each point (posterior)

3. Update the parameters $\theta_1, \ldots, \theta_k$

EM Algorithm

ClusteringByExpectationMaximization (point set *D*, int *k*)

Generate an initial model $M' = (C_1', \ldots, C_k')$

repeat

// assign points to clusters – **expectation step**

For each object $\mathbf{x}_j$ from D and for each cluster (= Gaussian) C_i

Compute $P(C_i|\mathbf{x_j}) = \mathrm{W_{ij}}$,

// compute the model - **maximization step**

For each Cluster $\boldsymbol{C_i}$

Compute a new model $M = \{C_1, \ldots, C_k\}$ by **recomputing** $\boldsymbol{P(C_i), \mu_{C_i}, \Sigma_{C_i}}$

$M' \leftarrow M$

until $\sum_{i=1}^{k} \left\| \mu_{C_i}^{t} - \mu_{C_i}^{t-1} \right\| \leq \varepsilon$

return M

Expectation step: If we had the parameters θ …

Idea: Assume you have computed the parameters $\theta_1, \ldots, \theta_k$

You could „assign“ points to clusters by computing the posterior

- A point $\mathbf{x_j}$ may belong to several clusters with different probabilities $P(\mathbf{x_j}|C_i)$

we call them weights W_{ij}

$$P(C_i|\mathbf{x_j}) = \frac{P(\mathbf{x}_j \mid C_i)P(C_i)}{P(\mathbf{x}_j)} = \frac{P(\mathbf{x}_j \mid C_i)P(C_i)}{\sum_a P(\mathbf{x}_j \mid C_a)P(C_a)}$$

where $P(\mathbf{x}_j \mid C_i) \simeq 2\epsilon \cdot f(\mathbf{x_j}|\boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}) = 2\epsilon \cdot f(\mathbf{x}_j)$

… and then you could update the parameters θ

Source: Designed by Freepik at flaticon.com

Maximization step: Recomputation of Parameters

- **Center μ_i of cluster C_i**
  - Re-estimate the mean as the weighted average of all points

$$\mu_i = \frac{\sum_{j=1}^{n} \mathbf{x}_j \cdot W_{ij}}{\sum_{j=1}^{n} W_{ij}}$$

- **Covariance matrix Σ_i of cluster C_i**
  - Re-estimate the covariance matrix as the weighted covariance over all pairs of dimensions

$$\Sigma_i = \frac{\sum_{j=1}^{n} W_{ij} \left(\mathbf{x_j} - \mu_i\right)\left(\mathbf{x_j} - \mu_i\right)^T}{\sum_{j=1}^{n} W_{ij}}$$

- **Prior probability on cluster C_i**
  - Re-estimate the prior probablity for each cluster as the fraction of weights that contribute to that cluster

$$P(C_i) = \frac{1}{n}\sum_{j=1}^{n} P\left(C_i \mid x_j\right) = \frac{\sum_{j=1}^{n} W_{ij}}{n}$$

Curious on the motivation?

- You can find a very thorough explanation in these slides:
  - https://www.cs.toronto.edu/~rgrosse/courses/csc411_f18/slides/lec16-slides.pdf

EM: Why?

There is a long analysis in Section 13.3.3/13.3.4 in the book … but let's have a taste of it!

- Maximize the log-likelihood $\mathcal{L}(\Theta) = \ln P(\mathbf{D}|\, \theta_1, \ldots \theta_k)$
  1. Compute the gradients $\frac{\partial \mathcal{L}(\Theta)}{\partial \Theta}$
  2. Solve $\frac{\partial \mathcal{L}(\Theta)}{\partial \Theta} = 0$
- This is hard! So assume we know the only cluster C_i from which each point $\mathbf{x}_j$ comes from …
  1. The clustering assignment is a vector of Bernulli variables $\mathbf{C} = (C_1, C_2, \ldots C_k)$
  2. Only one of the $C_i = 1$, the others are zero
  3. The probability of a point to belong to the cluster is a simple product of the probabilities of each cluster
  4. This simplifies the log-likelihood
- **Expectation step:** Compute the expected value of $\mathcal{L}(\Theta)$
- **Maximization step:** Maximize the expected value … now it's easier to solve the gradients ☺

Covariance

Join at menti.com | use code 1941 7577

EM covariance matrix

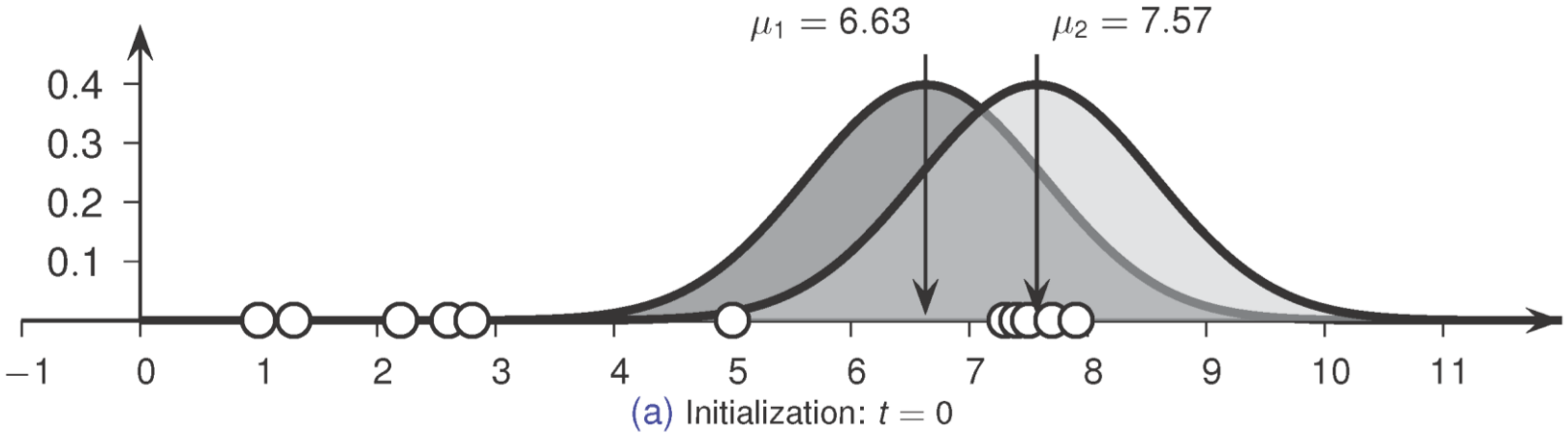
- **Full covariance matrix**
  - Detailed model
  - Requires estimation of $O(d^2)$ parameters
  - Often not enough data for reliable estimation
  - Costly

$$\mathbf{\Sigma}_i = \begin{pmatrix} (\sigma_1^i)^2 & \sigma_{12}^i & \cdots & \sigma_{1d}^i \\ \sigma_{21}^i & (\sigma_2^i)^2 & \cdots & \sigma_{2d}^i \\ \vdots & \vdots & \ddots & \\ \sigma_{d1}^i & \sigma_{d2}^i & \cdots & (\sigma_d^i)^2 \end{pmatrix}$$

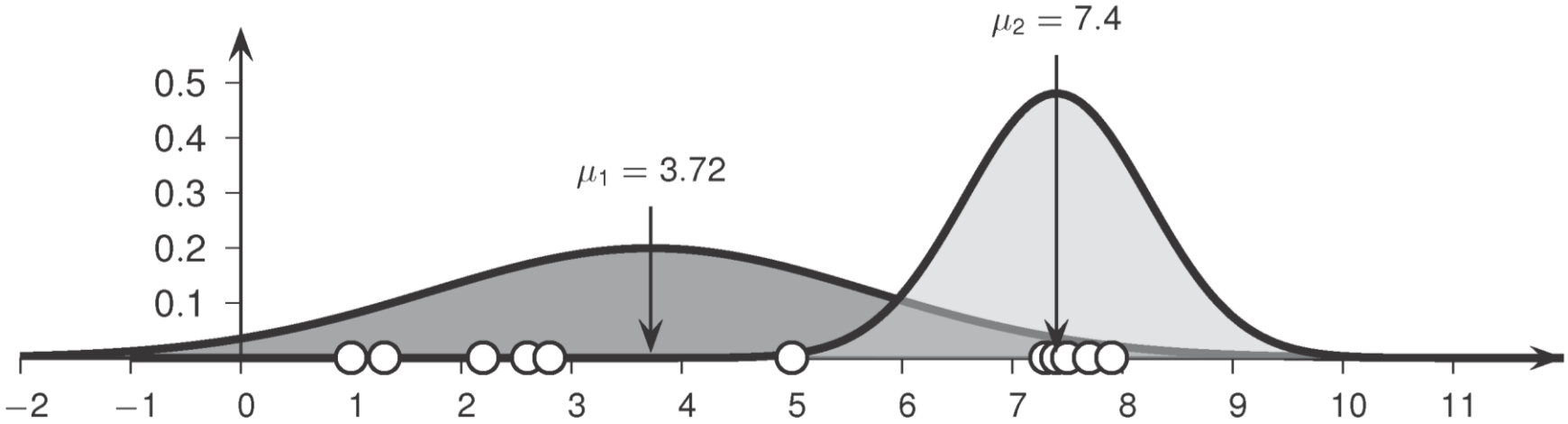
- **Diagonal covariance matrix**
  - Simplified model
  - Assume all dimensions are independent
  - Only estimate d parameters

$$\mathbf{\Sigma}_i = \begin{pmatrix} (\sigma_1^i)^2 & 0 & \cdots & 0 \\ 0 & (\sigma_2^i)^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & \cdots & (\sigma_d^i)^2 \end{pmatrix}$$

EM example in one dimension

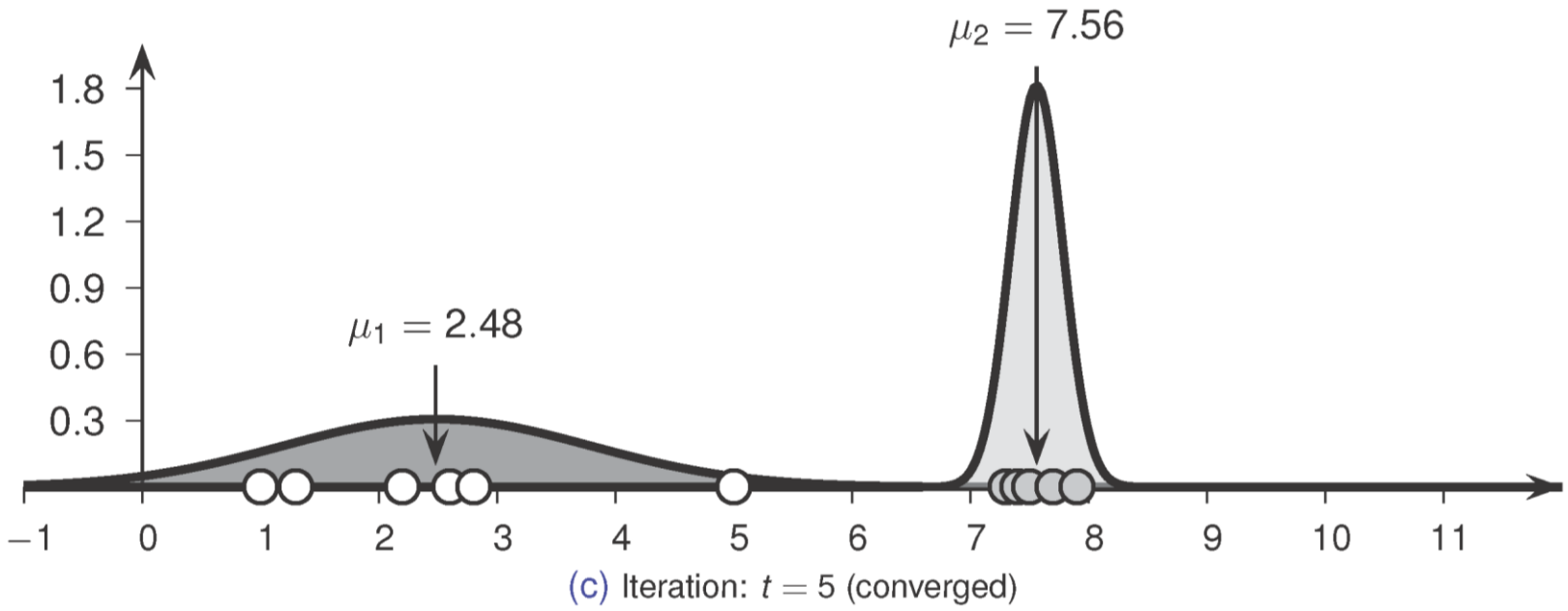


(a) Initialization: $t = 0$



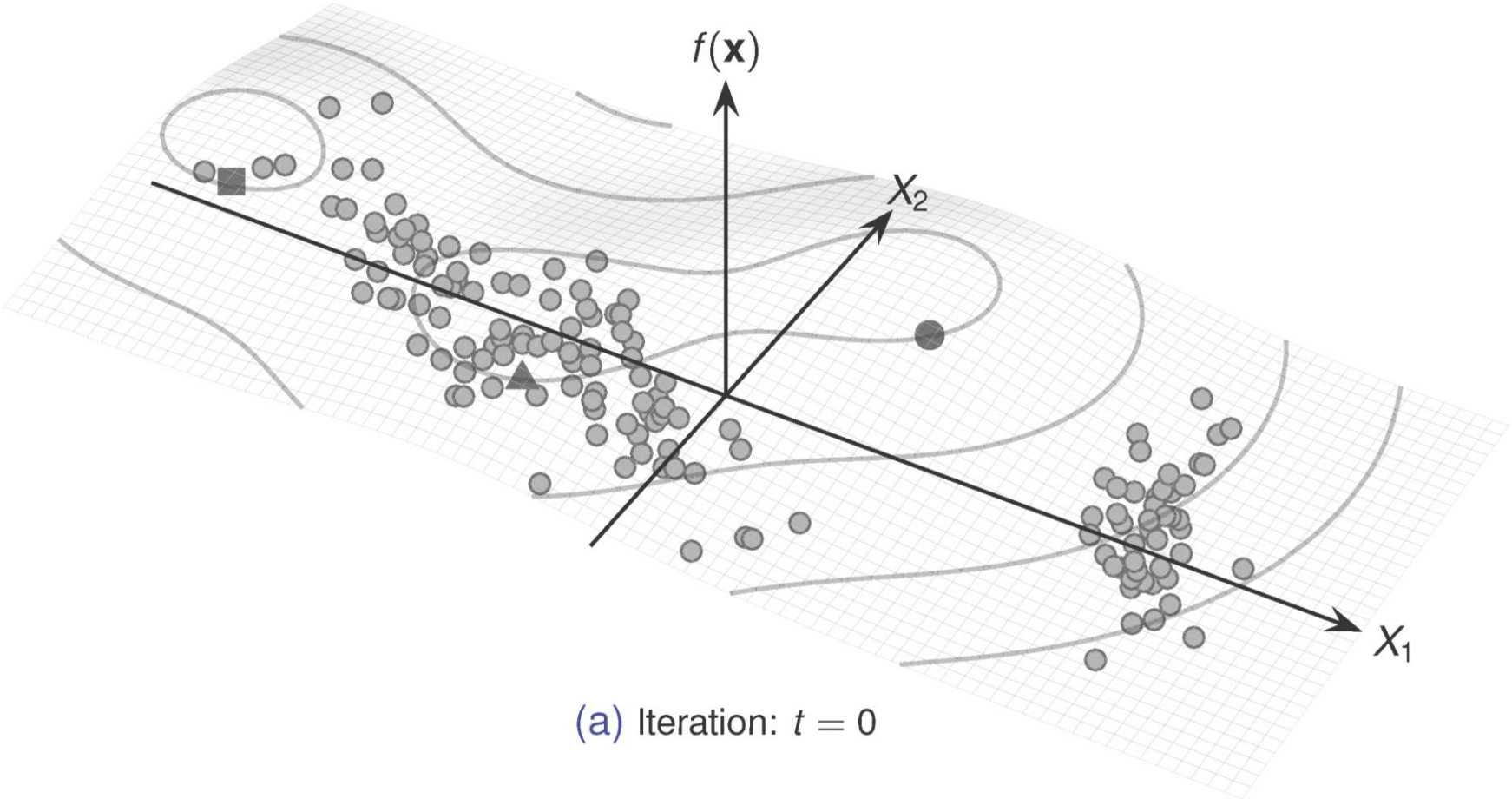
(b) Iteration: $t = 1$

EM example in one dimension



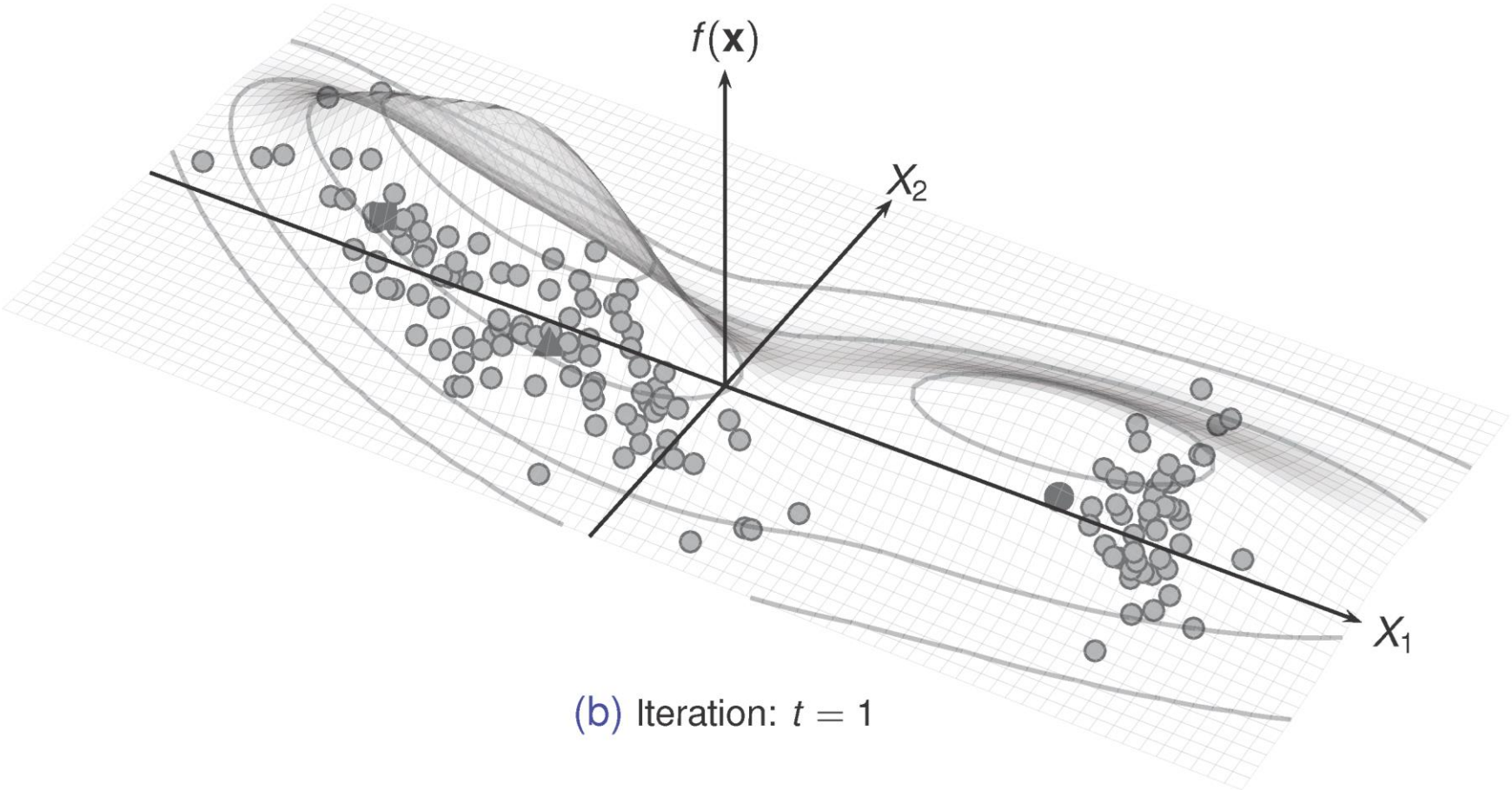
(c) Iteration: $t = 5$ (converged)

EM example in two dimensions, 3 Gaussians, Iris dataset



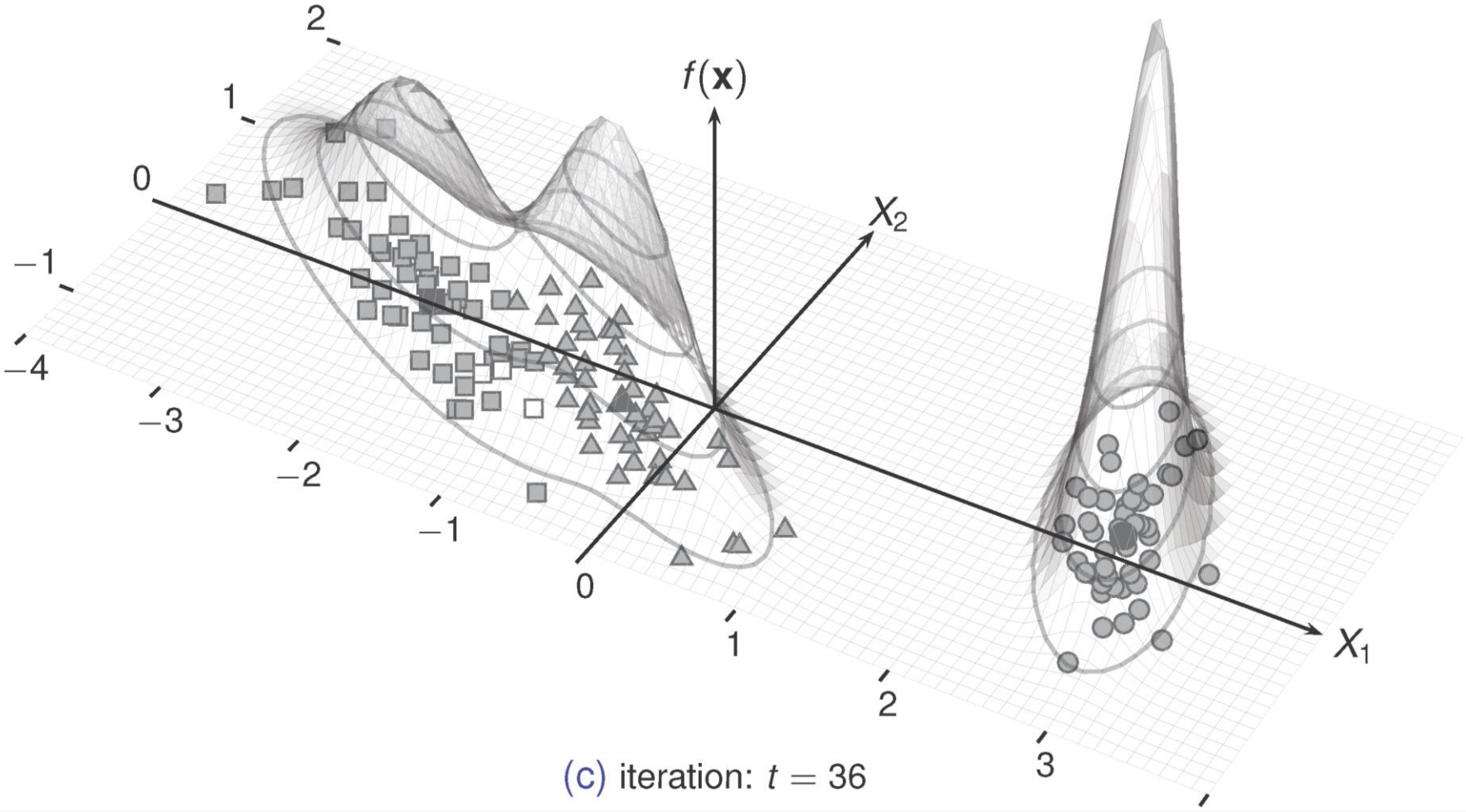
(a) Iteration: $t = 0$

EM example in two dimensions, 3 Gaussians, Iris dataset



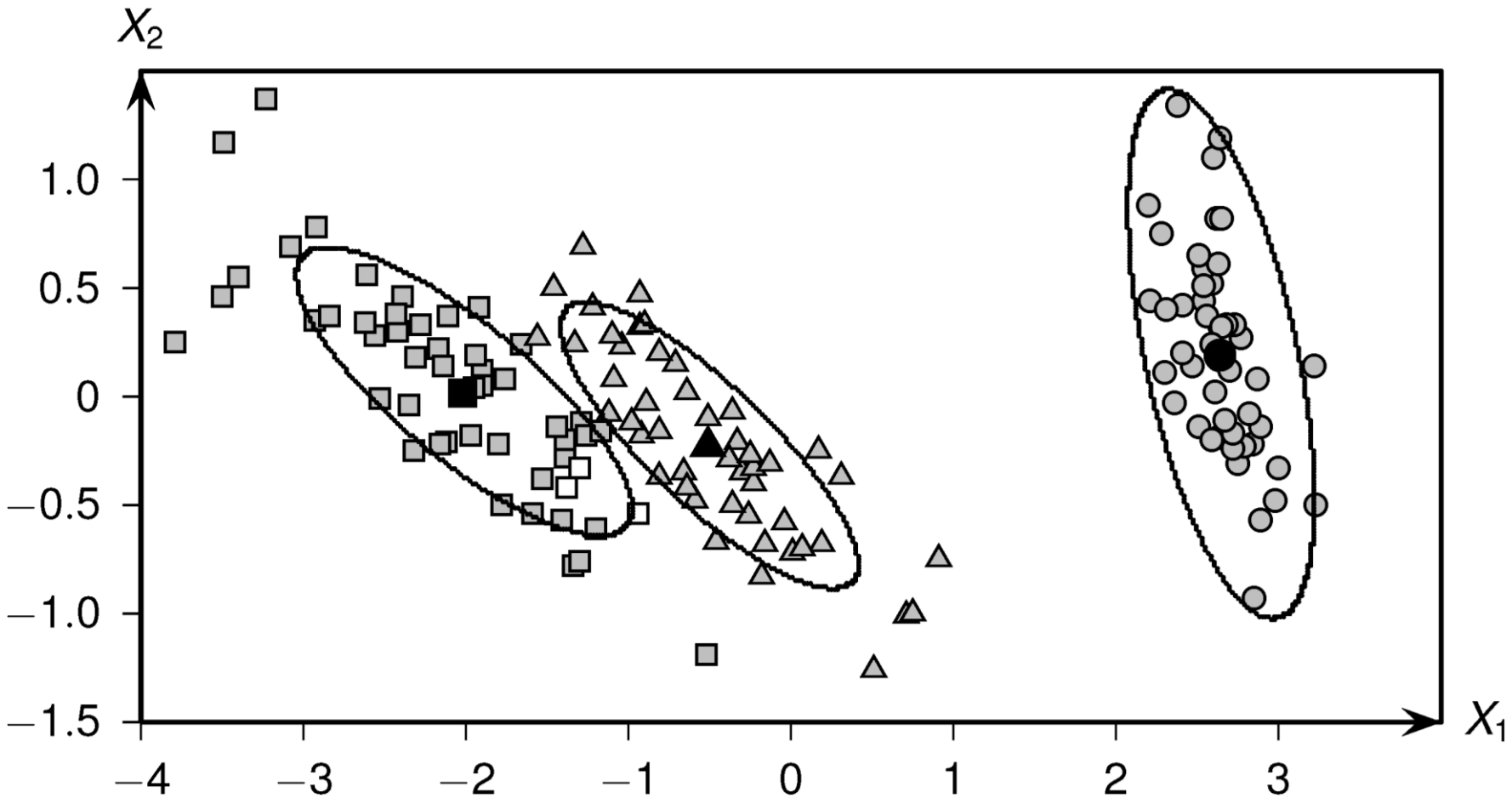
(b) Iteration: $t = 1$

EM example in two dimensions, 3 Gaussians, Iris dataset



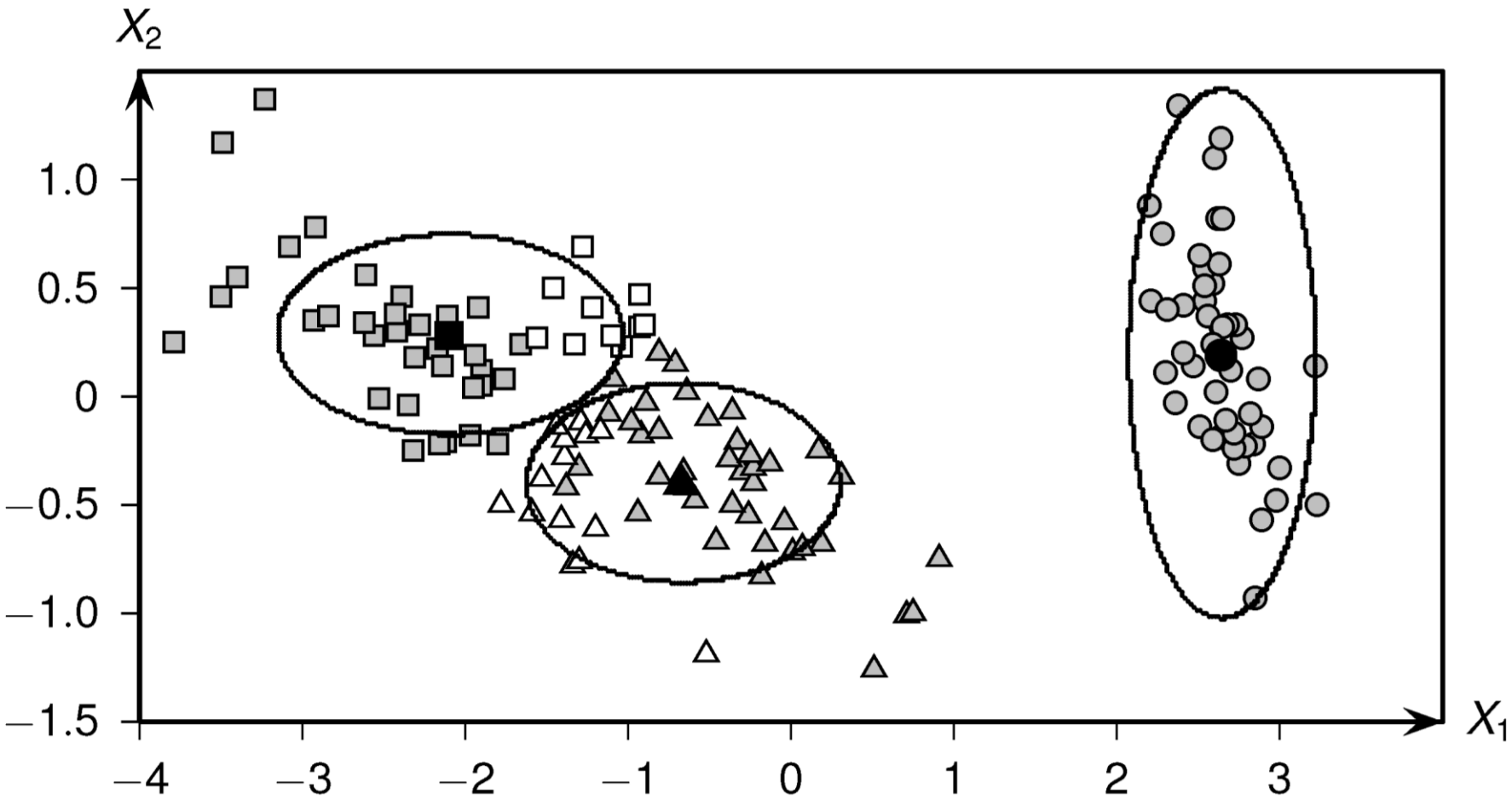
(c) iteration: $t = 36$

EM example in two dimensions, full covariance



(a) Full covariance matrix ($t = 36$)

EM in two components, three Gaussians, diagonal covariance



(b) Diagonal covariance matrix ($t = 29$)

EM – Discussion

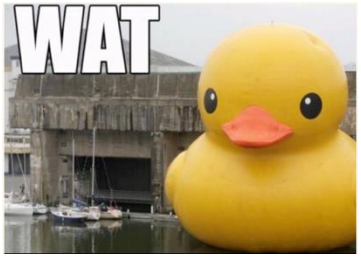
Advantages

- Flexible and powerful probabilistic model
- Captures overlapping clusters

Disadvantages

- Convergence to (possibly local) minimum
- Computational effort:
  - O(n · k · #iterations)
  - #iterations is quite high in many cases
- Both result and runtime strongly depend on
  - the initial assignment
  - a proper choice of parameter k (= desired number of clusters)
- Modification to obtain a partitioning variant
  - Objects may belong to several clusters
  - Assign each object to the cluster to which it belongs with the highest probability

A (harder?) question ….



When does EM clustering becomes k-means clustering?

- [1 minute] Small personal introduction
- [1 minute] Discuss the answer to the question!
- Add your answers in Mentimeter

Summary

- **Clustering**
  - Grouping of data based on mutual similarity
    - Similar data in the same cluster, dissimilar in different ones
- **Representation-based approaches / Partitioning approaches**
  - Characterize a cluster by a representative (mean, medoid)
  - Iterative update of clustering
  - Choice of parameter k
- **Next week**: density-based clustering

What was today's lecture about?

- Representative-based clustering has the following general idea…
- As opposed to k-means, k-medoid…
- The major difference to EM is…
- When evaluating clusters, we can…

What did you (not) grasp today?

Acknowledgements

- Slides for book Data Mining and Analysis by Mohammed J Zaki and Wagner Meira Jr
- Slides for book Data Mining: Concepts and Techniques, 2nd ed. by Jiawei Han and Micheline Kamber
- Slides for course on Knowledge Discovery in Databases by Jörg Sander and Martin Ester
- Slides for course Data Mining Algorithms by Thomas Seidl
- Slides for course Advanced Data Mining Algorithms by Ira Assent

References

- J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.
- M. H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall, 2003.
- U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.
- T. Imielinski and H. Mannila. *A database perspective on knowledge discovery*. Communications of the ACM, 39:58-64, 1996.
- G. Piatetsky-Shapiro, U. Fayyad, and P. Smith. *From data mining to knowledge discovery: An overview*. In U.M. Fayyad, et al. (eds.), Advances in Knowledge Discovery and Data Mining, 1-35. AAAI/MIT Press, 1996.
- G. Piatetsky-Shapiro and W. J. Frawley. *Knowledge Discovery in Databases*. AAAI/MIT Press, 1991.
- D. Hand, H. Mannila, P. Smyth. *Principles of Data Mining*. MIT Press, 2001.