

Fall 2022 ECEN 748

Data Mining & Analysis

High Dimensional Data & Clustering

Nick Duffield

Department of Electrical & Computer Engineering

Texas A&M Institute of Data Science



TEXAS A&M UNIVERSITY

Department of Electrical
& Computer Engineering

Texas A&M Institute of Data Science <https://tamids.tamu.edu>



TEXAS A&M

Institute of
Data Science

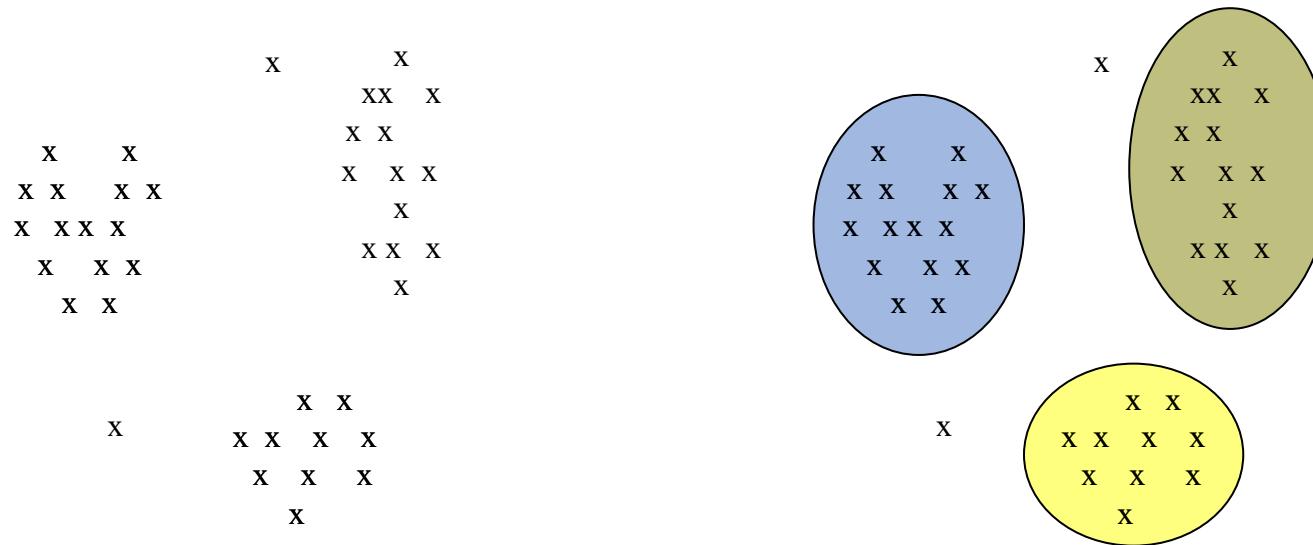
High Dimensional Data

- Given a cloud of data points we want to understand its structure



Clustering

- In many data sets, the points evidently disjoin into groups
 - Different subpopulations within data, each with own data characteristic



- Want to automate identification of clusters in data
- Different notions of clustering
 - Representative: cluster points close to a central representative
 - Density based: separated groups of contiguous points
 - Hierarchical clustering: grouping at multiple levels

The Problem of Clustering

- **Given a set of points, with a notion of distance between points, group the points into some number of *clusters*, so that**
 - Members of a cluster are close/similar to each other
 - Members of different clusters are dissimilar
- **Usually:**
 - Points are in a high-dimensional space
 - Similarity is defined using a distance measure
 - Euclidean, Cosine, Jaccard, edit distance, ...



Why is it hard?

- Clustering in two dimensions looks easy
- Clustering small amounts of data looks easy
- Many applications involve not 2, but 10 or 10,000 dimensions
- High-dimensional spaces look different: Almost all pairs of points are at about the same distance
 - Assuming random points within a bounding box, e.g., values between 0 and 1 in each dimension.
 - Differences in individual coordinates get averaged away in high dimensions.
 - See [ZM] Chapter 6 for further details



Representative-based Clustering

Given a dataset with n points in a d -dimensional space, $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$, and given the number of desired clusters k , the goal of representative-based clustering is to partition the dataset into k groups or clusters, which is called a *clustering* and is denoted as $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$.

For each cluster C_i there exists a representative point that summarizes the cluster, a common choice being the mean (also called the *centroid*) μ_i of all points in the cluster, that is,

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{x_j \in C_i} \mathbf{x}_j$$

where $n_i = |C_i|$ is the number of points in cluster C_i .

A brute-force or exhaustive algorithm for finding a good clustering is simply to generate all possible partitions of n points into k clusters, evaluate some optimization score for each of them, and retain the clustering that yields the best score. However, this is clearly infeasible, since there are $O(k^n/k!)$ clusterings of n points into k groups.

K-means Algorithm: Objective

The *sum of squared errors* scoring function is defined as

$$SSE(\mathcal{C}) = \sum_{i=1}^k \sum_{x_j \in C_i} \|x_j - \mu_i\|^2$$

The goal is to find the clustering that minimizes the SSE score:

$$\mathcal{C}^* = \arg \min_{\mathcal{C}} \{SSE(\mathcal{C})\}$$

K-means employs a greedy iterative approach to find a clustering that minimizes the SSE objective. As such it can converge to a local optima instead of a globally optimal clustering.



k -means Algorithm(s)

- Assumes Euclidean space/distance
- Start by picking k , the number of clusters
- Initialize clusters by picking one point per cluster
 - Example: Pick one point at random, then $k-1$ other points, each as far away as possible from the previous points



K-Means Algorithm

K-means (D, k, ϵ):

```
1  $t = 0$ 
2 Randomly initialize  $k$  centroids:  $\mu_1^t, \mu_2^t, \dots, \mu_k^t \in \mathbb{R}^d$ 
3 repeat
4    $t \leftarrow t + 1$ 
5    $C_j \leftarrow \emptyset$  for all  $j = 1, \dots, k$ 
     // Cluster Assignment Step
6   foreach  $x_j \in D$  do
7      $i^* \leftarrow \arg \min_i \left\{ \|x_j - \mu_i^t\|^2 \right\}$  // Assign  $x_j$  to closest
      centroid
8      $C_{i^*} \leftarrow C_{i^*} \cup \{x_j\}$ 
     // Centroid Update Step
9   foreach  $i = 1$  to  $k$  do
10     $\mu_i^t \leftarrow \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$ 
11 until  $\sum_{i=1}^k \|\mu_i^t - \mu_i^{t-1}\|^2 \leq \epsilon$ 
```



K-means Algorithm: Objective

K-means initializes the cluster means by randomly generating k points in the data space. Each iteration of K-means consists of two steps: (1) cluster assignment, and (2) centroid update.

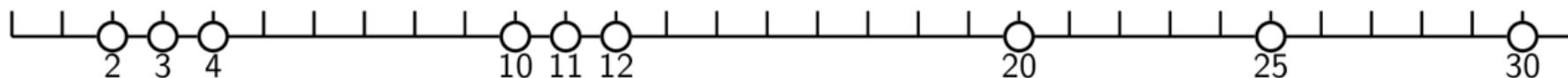
Given the k cluster means, in the cluster assignment step, each point $\mathbf{x}_j \in \mathcal{D}$ is assigned to the closest mean, which induces a clustering, with each cluster C_i comprising points that are closer to μ_i than any other cluster mean. That is, each point \mathbf{x}_j is assigned to cluster C_{i^*} , where

$$i^* = \arg \min_{i=1}^k \left\{ \|\mathbf{x}_j - \mu_i\|^2 \right\}$$

Given a set of clusters C_i , $i = 1, \dots, k$, in the centroid update step, new mean values are computed for each cluster from the points in C_i .

The cluster assignment and centroid update steps are carried out iteratively until we reach a fixed point or local minima.

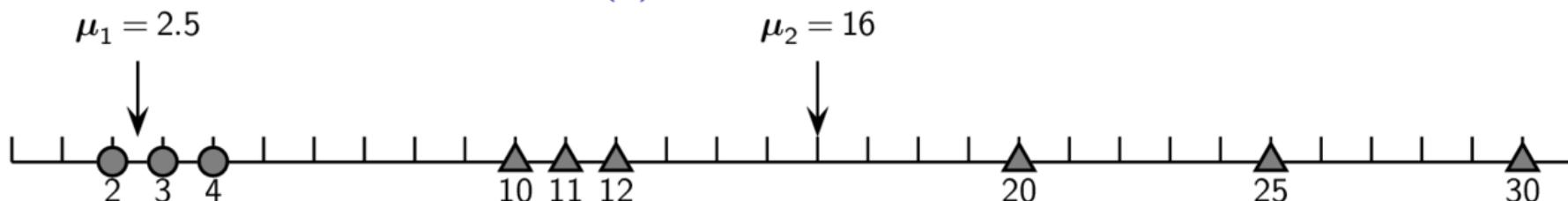
K-means in One Dimension



(a) Initial dataset

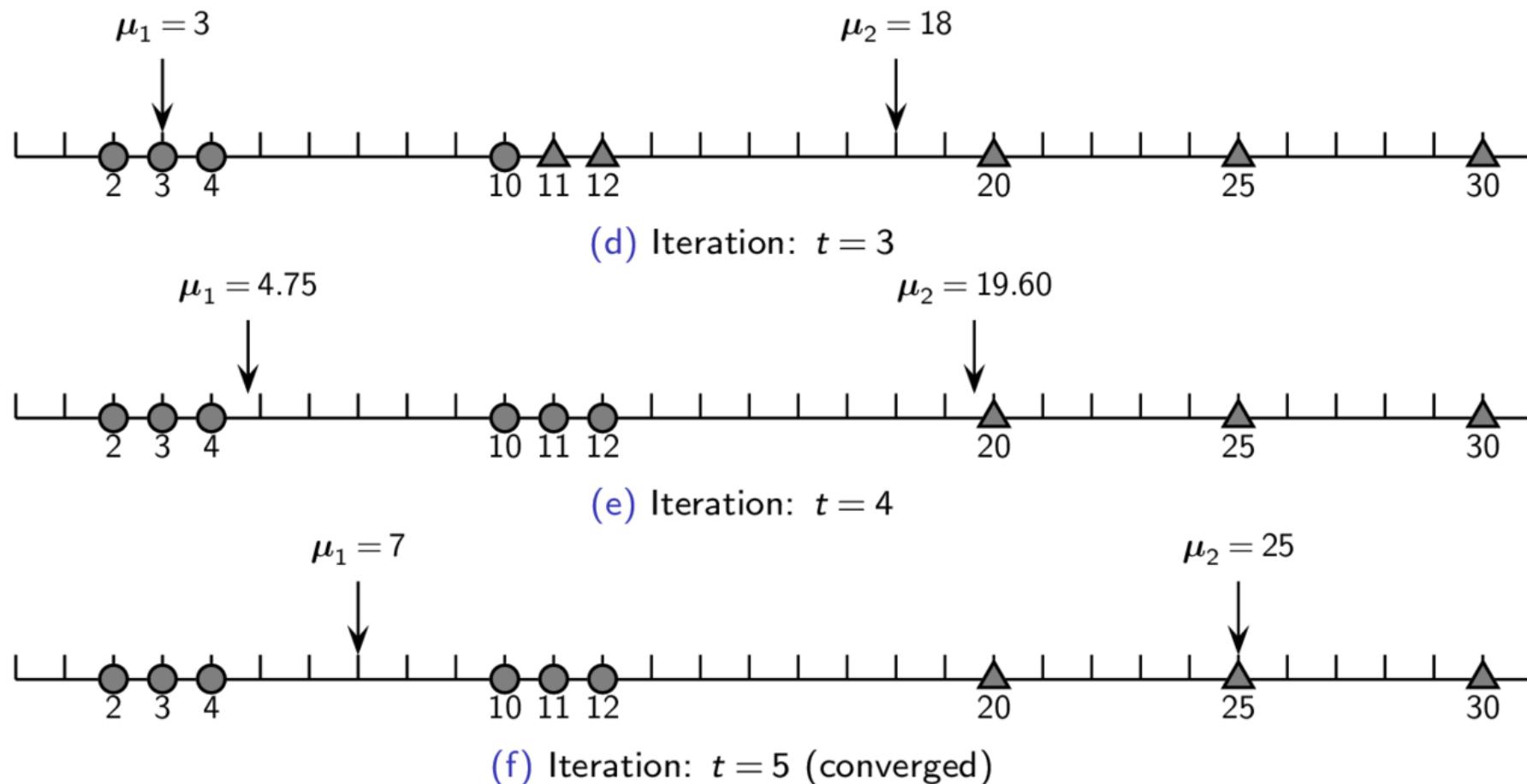


(b) Iteration: $t = 1$

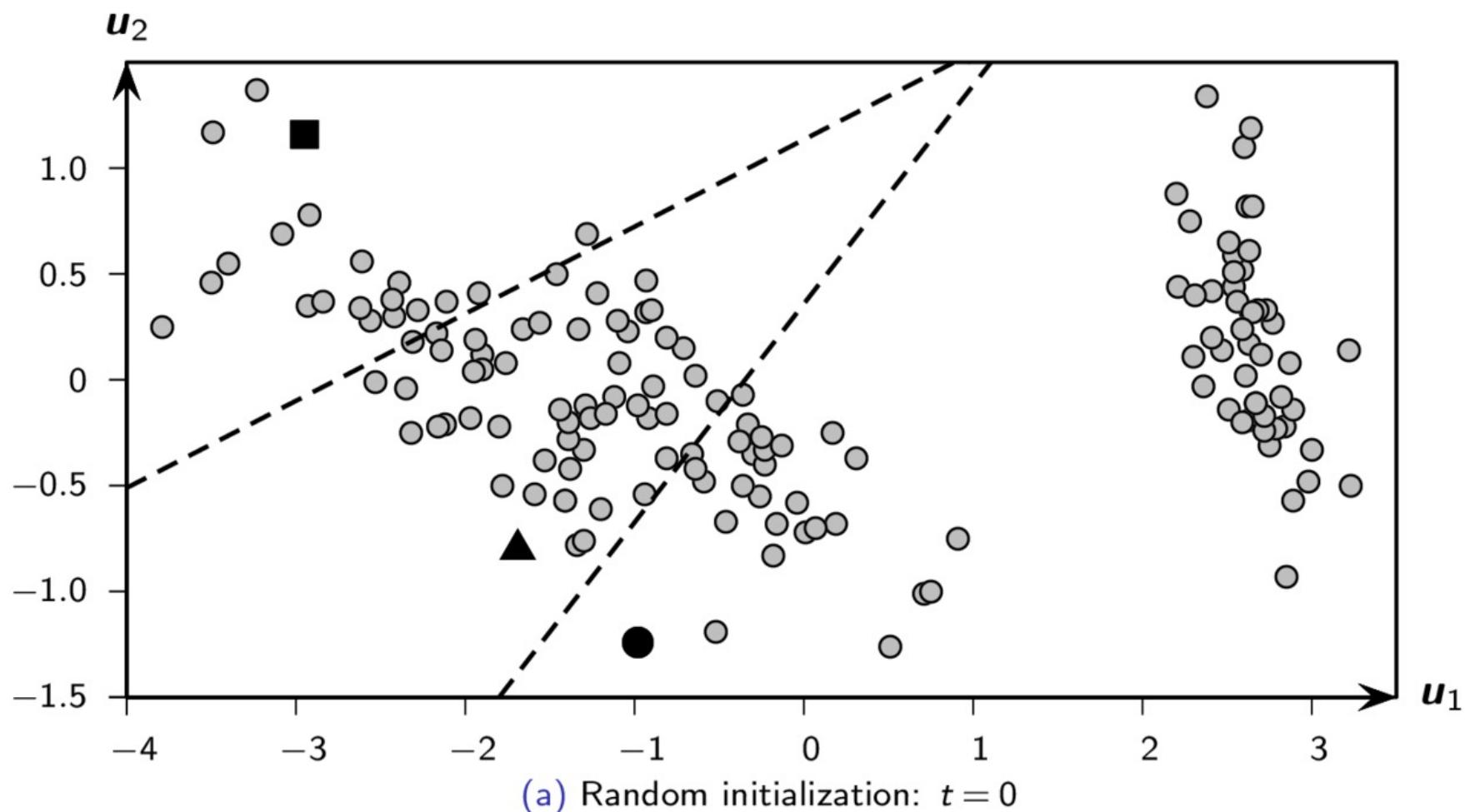


(c) Iteration: $t = 2$

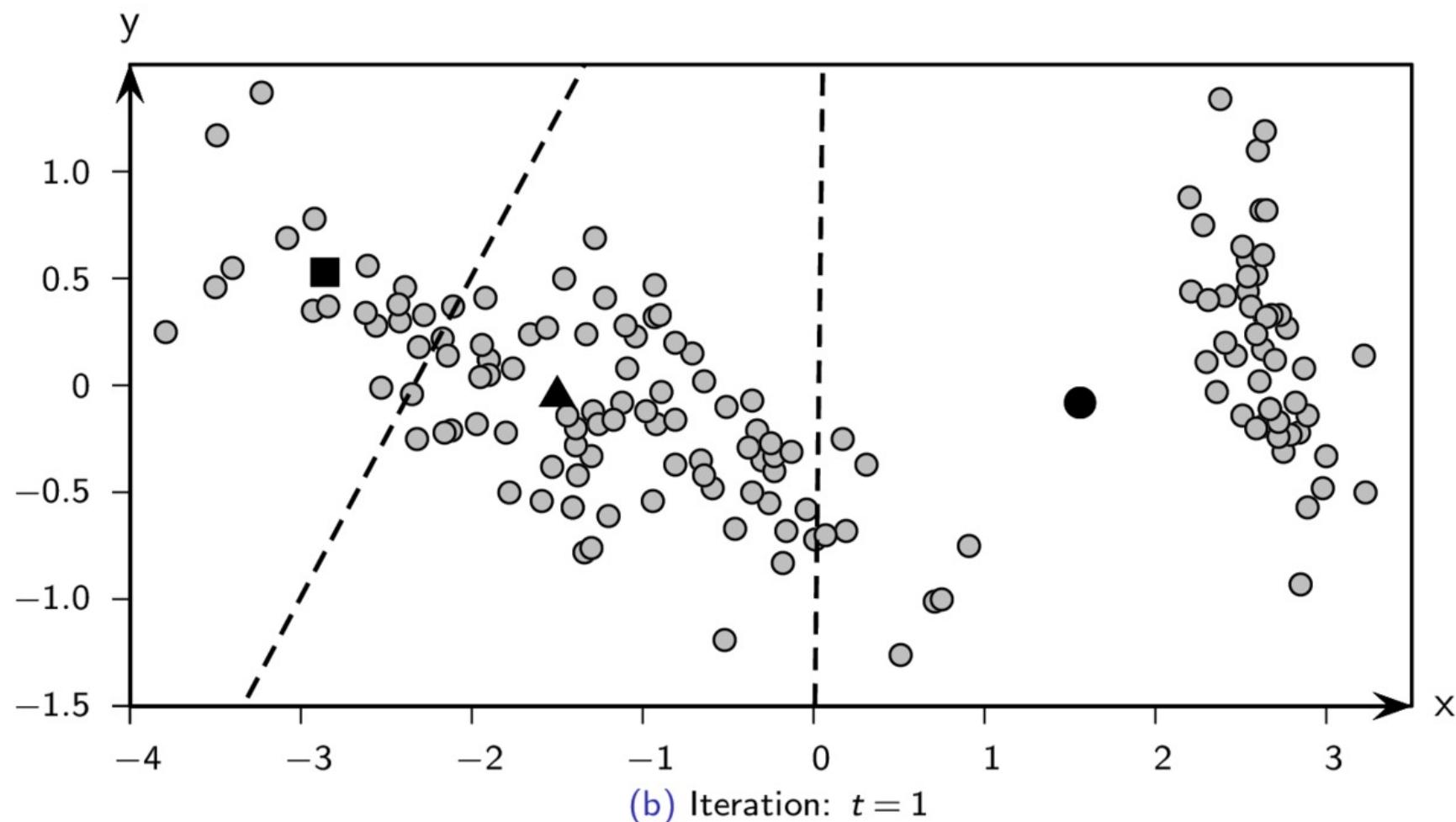
K-means in One Dimension (contd.)



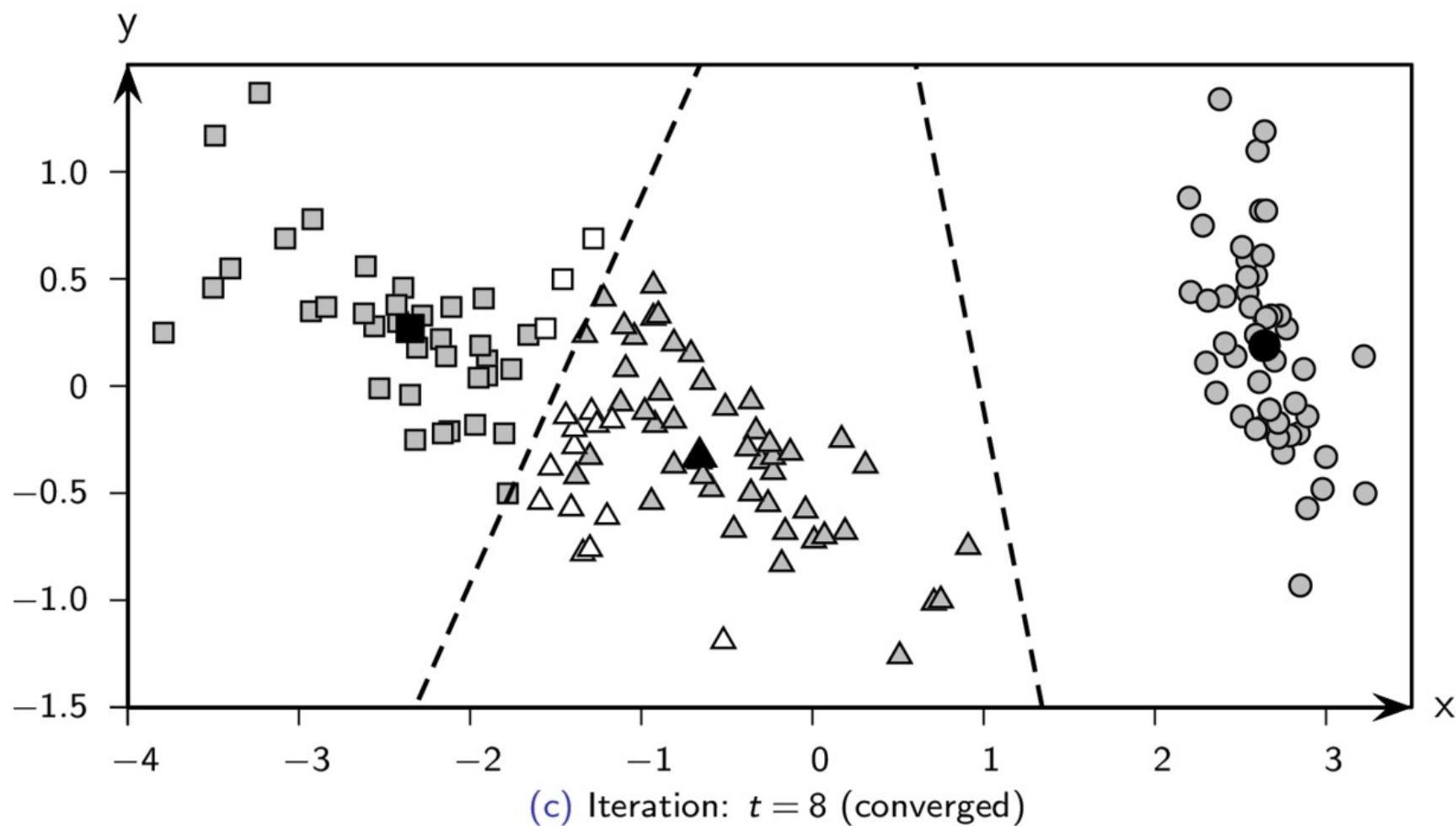
K-means in 2D: Iris Principal Components



K-means in 2D: Iris Principal Components



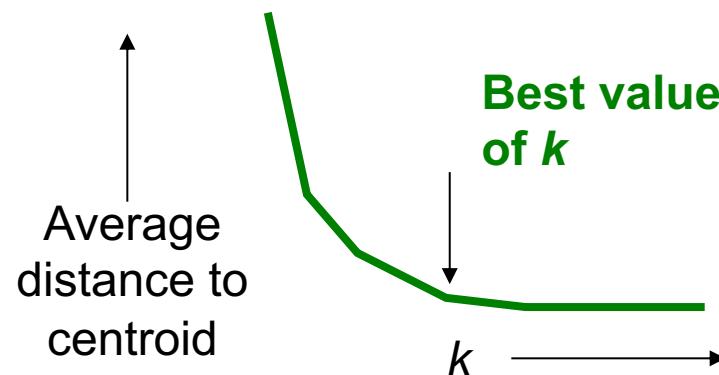
K-means in 2D: Iris Principal Components



Getting the k right

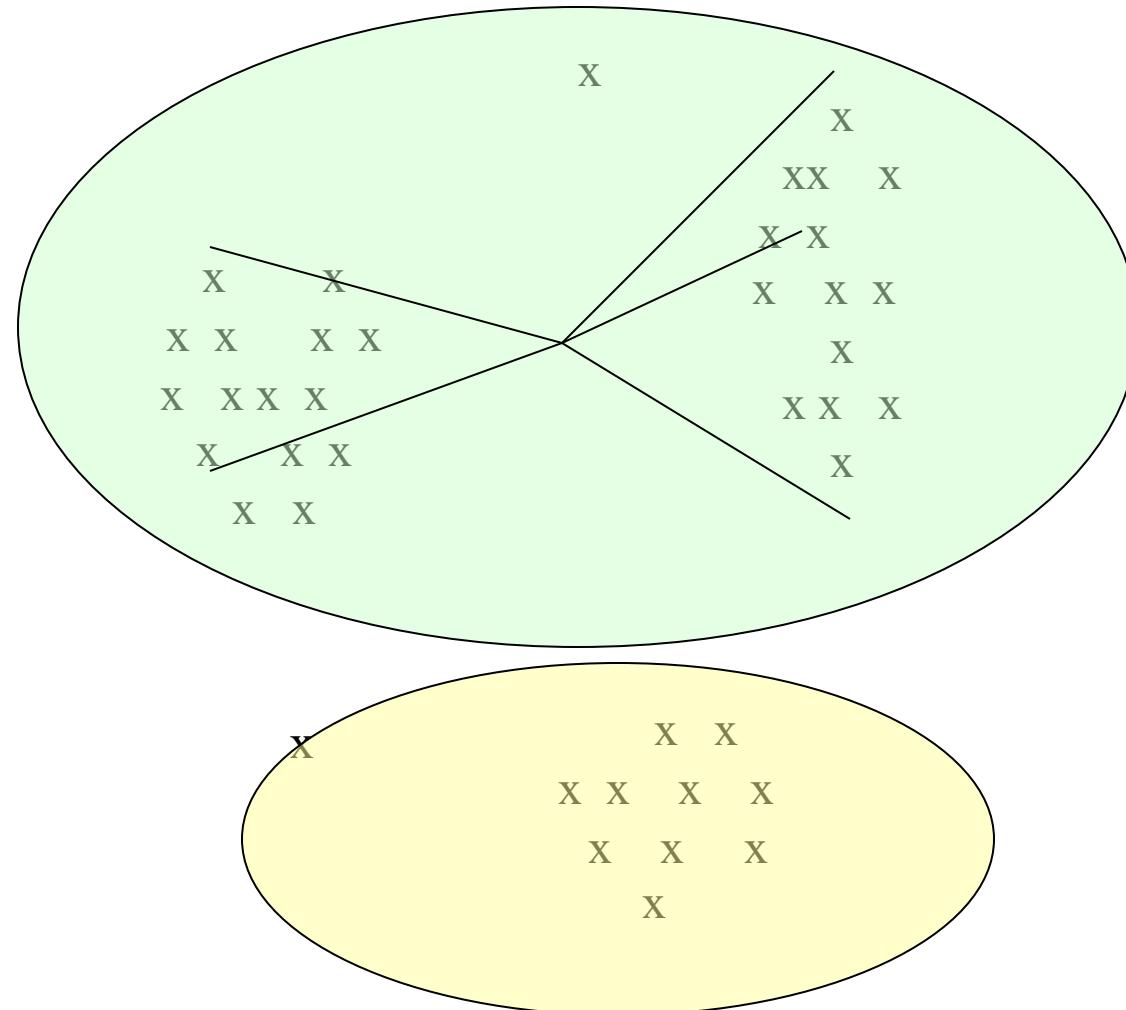
How to select k ?

- Try different k , looking at the change in the average distance to centroid as k increases
- Average falls rapidly until right k , then changes little



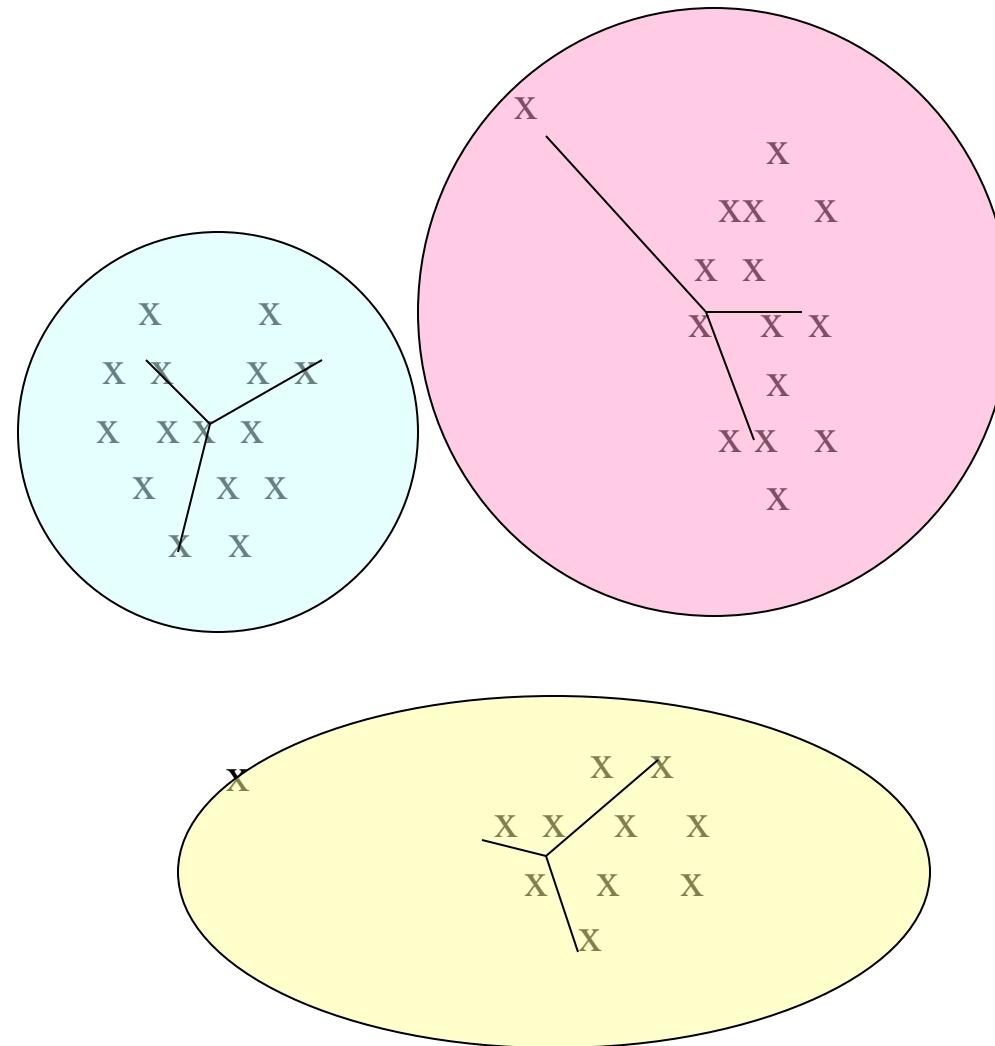
Example: Picking k

Too few;
many long
distances
to centroid.



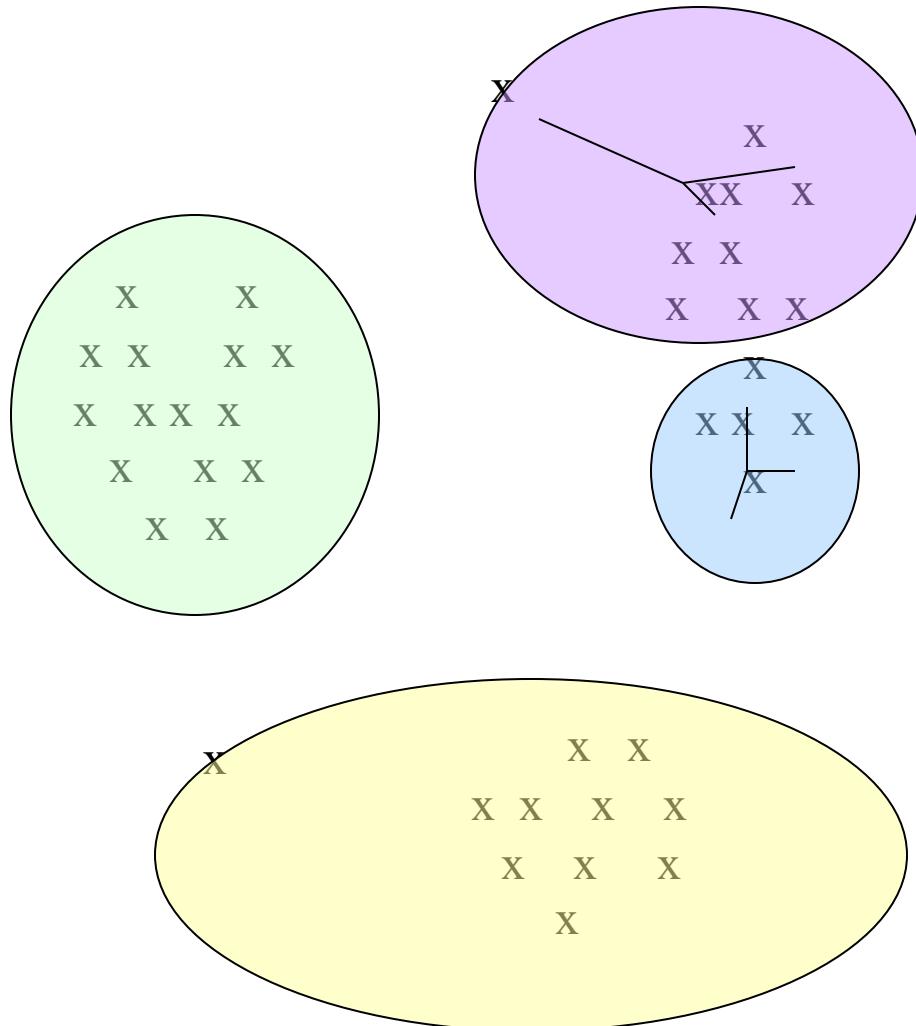
Example: Picking k

Just right;
distances
rather short.



Example: Picking k

Too many;
little improvement
in average
distance.



Distribution-Based Clustering

- **Representative-based clustering**
 - Based on distances between data points, #clusters
 - Iterate to find best cluster centroids
- **Distribution-based clustering**
 - Parameterized model of point distribution in clusters
 - E.g. Gaussian Distribution: mean, covariance matrix
 - Iteratively fit to find best cluster model parameters

Expectation-Maximization Clustering: Gaussian Mixture Model

Let X_a denote the random variable corresponding to the a th attribute. Let $\mathbf{X} = (X_1, X_2, \dots, X_d)$ denote the vector random variable across the d -attributes, with \mathbf{x}_j being a data sample from \mathbf{X} .

We assume that each cluster C_i is characterized by a multivariate normal distribution

$$f_i(\mathbf{x}) = f(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{\Sigma}_i|^{\frac{1}{2}}} \exp \left\{ -\frac{(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)}{2} \right\}$$

where the cluster mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and covariance matrix $\boldsymbol{\Sigma}_i \in \mathbb{R}^{d \times d}$ are both unknown parameters.

The probability density function of \mathbf{X} is given as a *Gaussian mixture model* over all the k clusters

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

where the prior probabilities $P(C_i)$ are called the *mixture parameters*, which must satisfy the condition

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

Expectation-Maximization Clustering: Maximum Likelihood Estimation

We write the set of all the model parameters compactly as

$$\theta = \{\mu_1, \Sigma_1, P(C_1), \dots, \mu_k, \Sigma_k, P(C_k)\}$$

Given the dataset \mathbf{D} , we define the *likelihood* of θ as the conditional probability of the data \mathbf{D} given the model parameters θ

$$P(\mathbf{D}|\theta) = \prod_{j=1}^n f(\mathbf{x}_j)$$

The goal of maximum likelihood estimation (MLE) is to choose the parameters θ that maximize the likelihood. We do this by maximizing the log of the likelihood function

$$\theta^* = \arg \max_{\theta} \{\ln P(\mathbf{D}|\theta)\}$$

where the *log-likelihood* function is given as

$$\ln P(\mathbf{D}|\theta) = \sum_{j=1}^n \ln f(\mathbf{x}_j) = \sum_{j=1}^n \ln \left(\sum_{i=1}^k f(\mathbf{x}_j | \mu_i, \Sigma_i) P(C_i) \right)$$

Expectation-Maximization Clustering

Directly maximizing the log-likelihood over θ is hard. Instead, we can use the expectation-maximization (EM) approach for finding the maximum likelihood estimates for the parameters θ .

EM is a two-step iterative approach that starts from an initial guess for the parameters θ . Given the current estimates for θ , in the *expectation step* EM computes the cluster posterior probabilities $P(C_i|\mathbf{x}_j)$ via the Bayes theorem:

$$P(C_i|\mathbf{x}_j) = \frac{P(C_i \text{ and } \mathbf{x}_j)}{P(\mathbf{x}_j)} = \frac{P(\mathbf{x}_j|C_i)P(C_i)}{\sum_{a=1}^k P(\mathbf{x}_j|C_a)P(C_a)} = \frac{f_i(\mathbf{x}_j) \cdot P(C_i)}{\sum_{a=1}^k f_a(\mathbf{x}_j) \cdot P(C_a)}$$

In the *maximization step*, using the weights $P(C_i|\mathbf{x}_j)$ EM re-estimates θ , that is, it re-estimates the parameters μ_i , Σ_i , and $P(C_i)$ for each cluster C_i . The re-estimated mean is given as the weighted average of all the points, the re-estimated covariance matrix is given as the weighted covariance over all pairs of dimensions, and the re-estimated prior probability for each cluster is given as the fraction of weights that contribute to that cluster.



EM in One Dimension: Expectation Step

Let \mathbf{D} comprise of a single attribute X , with each point $x_j \in \mathbb{R}$ a random sample from X . For the mixture model, we use univariate normals for each cluster:

$$f_i(x) = f(x|\mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left\{-\frac{(x - \mu_i)^2}{2\sigma_i^2}\right\}$$

with the cluster parameters μ_i , σ_i^2 , and $P(C_i)$.

Initialization: For each cluster C_i , with $i = 1, 2, \dots, k$, we can randomly initialize the cluster parameters μ_i , σ_i^2 , and $P(C_i)$.

Expectation Step: Given the mean μ_i , variance σ_i^2 , and prior probability $P(C_i)$ for each cluster, the cluster posterior probability is computed as

$$w_{ij} = P(C_i|x_j) = \frac{f(x_j|\mu_i, \sigma_i^2) \cdot P(C_i)}{\sum_{a=1}^k f(x_j|\mu_a, \sigma_a^2) \cdot P(C_a)}$$

EM in One Dimension: Maximization Step

Given w_{ij} values, the re-estimated cluster mean is

$$\mu_i = \frac{\sum_{j=1}^n w_{ij} \cdot x_j}{\sum_{j=1}^n w_{ij}}$$

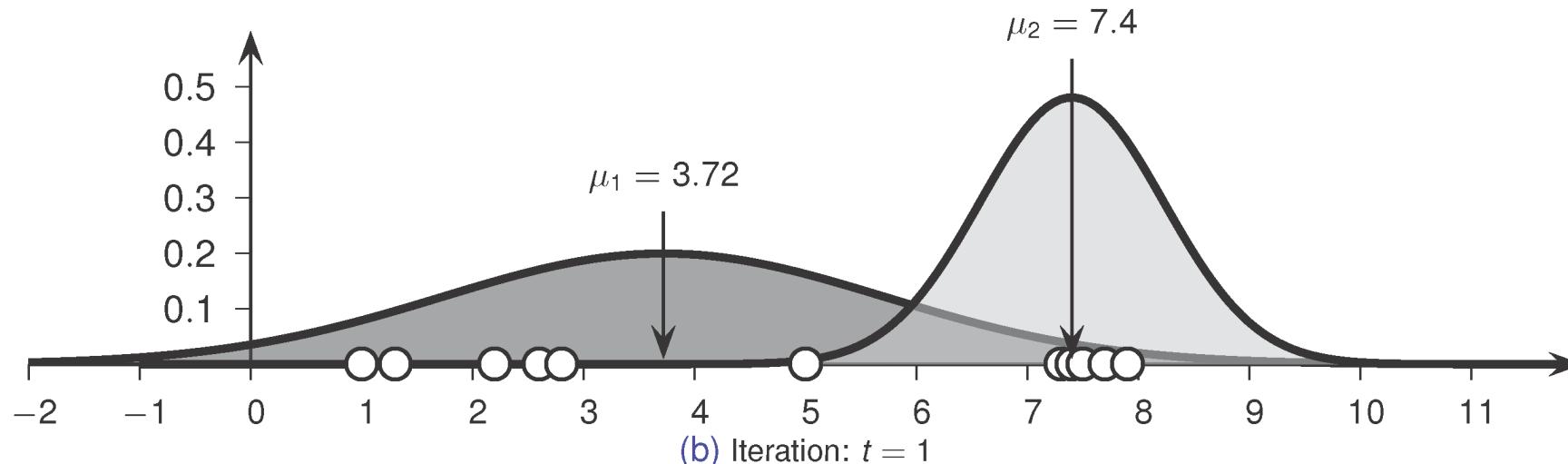
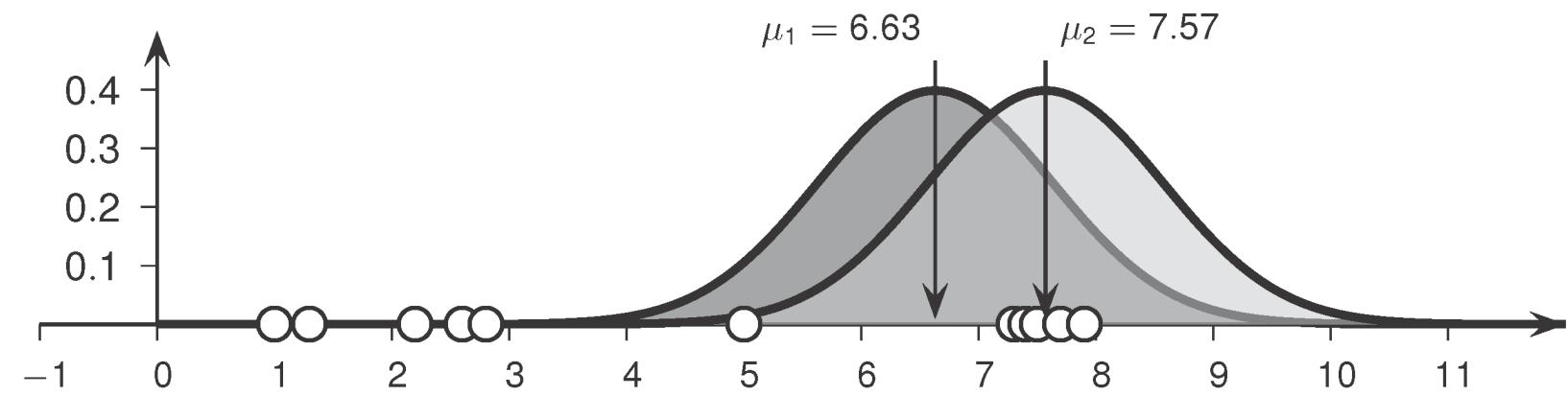
The re-estimated value of the cluster variance is computed as the weighted variance across all the points:

$$\sigma_i^2 = \frac{\sum_{j=1}^n w_{ij}(x_j - \mu_i)^2}{\sum_{j=1}^n w_{ij}}$$

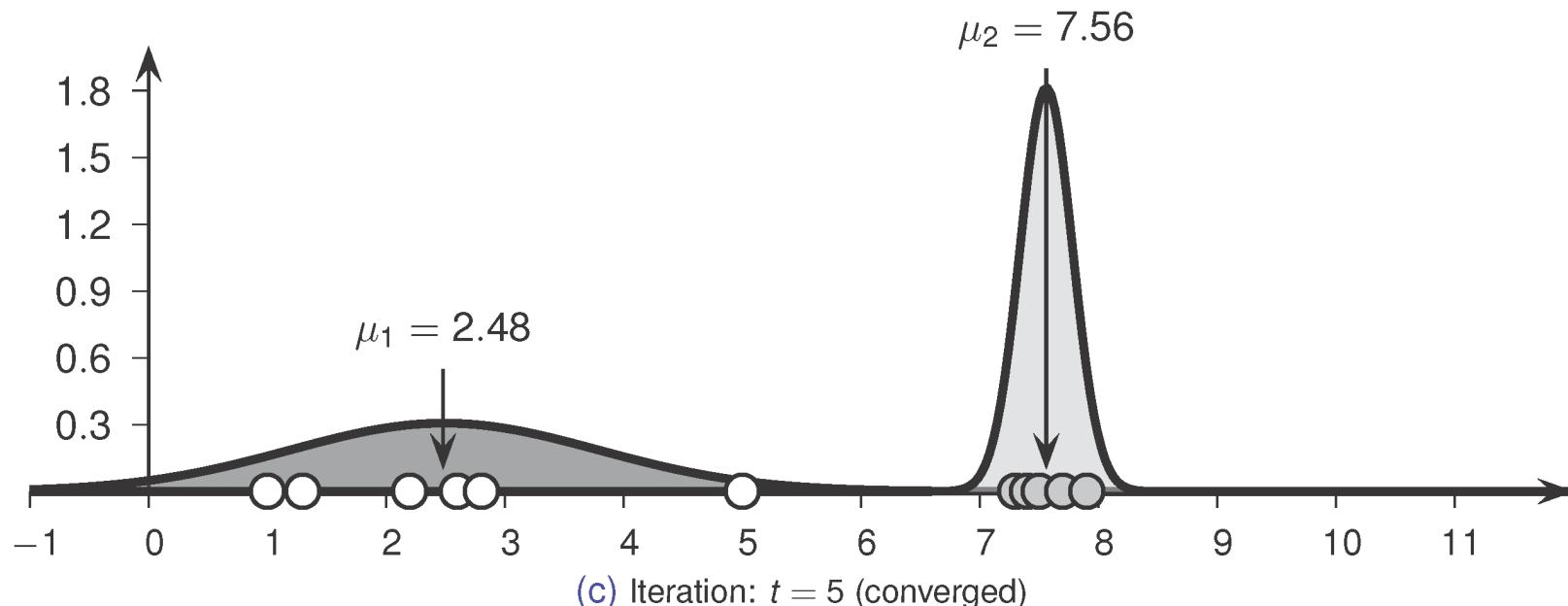
The prior probability of cluster C_i is re-estimated as

$$P(C_i) = \frac{\sum_{j=1}^n w_{ij}}{n}$$

EM in One Dimension



EM in One Dimension: Final Clusters



EM in d Dimensions

Each cluster we have to reestimate the $d \times d$ covariance matrix:

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

It requires $O(d^2)$ parameters, which may be too many for reliable estimation.
A simplification is to assume that all dimensions are independent, which leads
to a diagonal covariance matrix:

$$\Sigma_i = \begin{pmatrix} (\sigma_1^i)^2 & 0 & \dots & 0 \\ 0 & (\sigma_2^i)^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & \dots & (\sigma_d^i)^2 \end{pmatrix}$$

EM in d Dimensions

Expectation Step: Given μ_i , Σ_i , and $P(C_i)$, the posterior probability is given as

$$w_{ij} = P(C_i | \mathbf{x}_j) = \frac{f_i(\mathbf{x}_j) \cdot P(C_i)}{\sum_{a=1}^k f_a(\mathbf{x}_j) \cdot P(C_a)}$$

Maximization Step: Given the weights w_{ij} , in the maximization step, we re-estimate Σ_i , μ_i and $P(C_i)$.

The mean μ_i for cluster C_i can be estimated as

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

The covariance matrix Σ_i is re-estimated via the outer-product form

$$\Sigma_i = \frac{\sum_{j=1}^n w_{ij} (\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \mu_i)^T}{\sum_{j=1}^n w_{ij}}$$

The prior probability $P(C_i)$ for each cluster is

$$P(C_i) = \frac{\sum_{j=1}^n w_{ij}}{n}$$

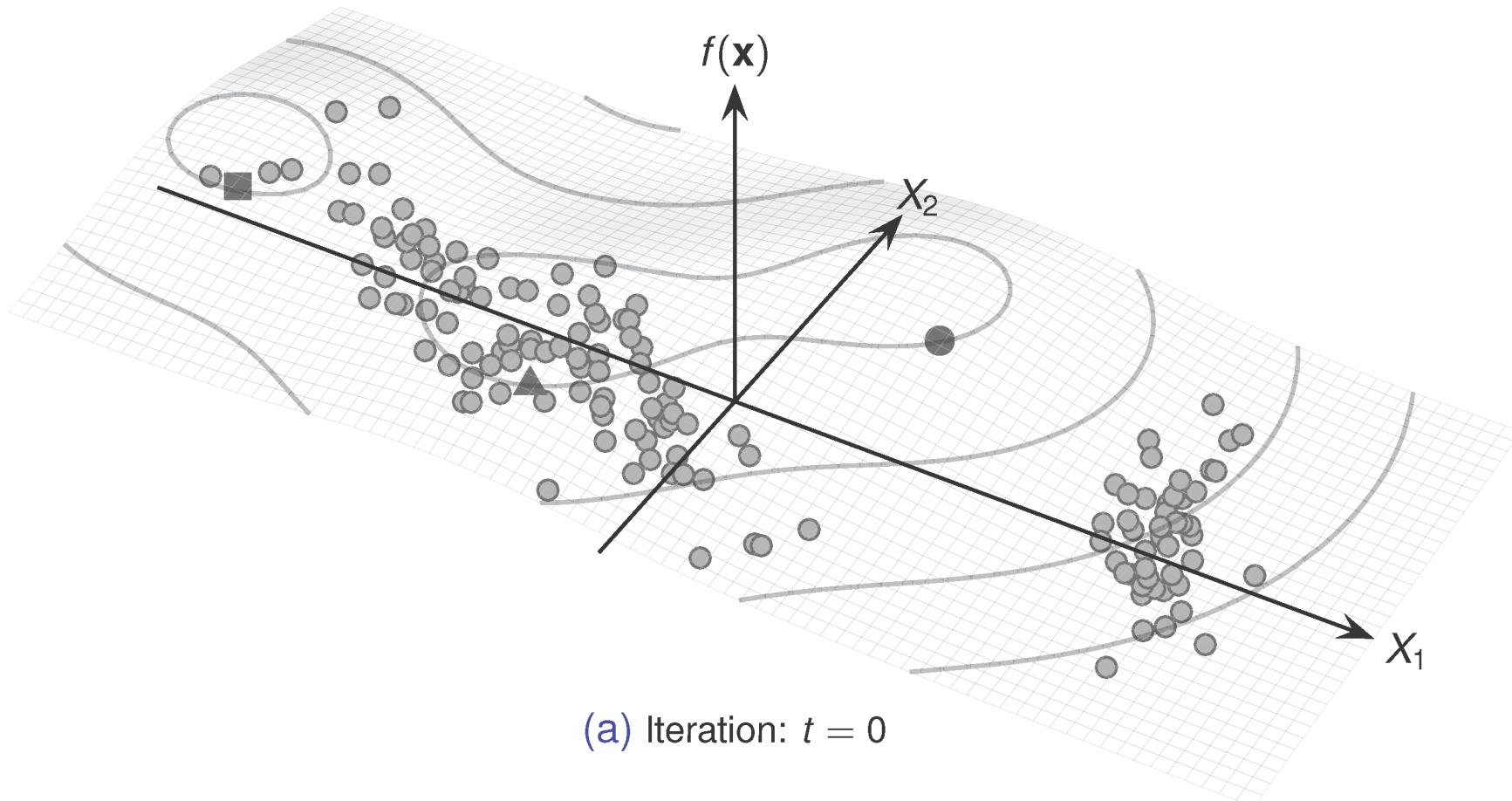
Expectation-Maximization Clustering Algorithm

EXPECTATION-MAXIMIZATION (\mathbf{D}, k, ϵ):

```
1  $t \leftarrow 0$ 
2 Randomly initialize  $\mu_1^t, \dots, \mu_k^t$ 
3  $\Sigma_i^t \leftarrow \mathbf{I}, \forall i = 1, \dots, k$ 
4 repeat
5    $t \leftarrow t + 1$ 
6   for  $i = 1, \dots, k$  and  $j = 1, \dots, n$  do
7      $w_{ij} \leftarrow \frac{f(\mathbf{x}_j | \mu_i, \Sigma_i) \cdot P(C_i)}{\sum_{a=1}^k f(\mathbf{x}_j | \mu_a, \Sigma_a) \cdot P(C_a)}$  // posterior probability
8      $P^t(C_i | \mathbf{x}_j)$ 
9   for  $i = 1, \dots, k$  do
10     $\mu_i^t \leftarrow \frac{\sum_{j=1}^n w_{ij} \cdot \mathbf{x}_j}{\sum_{j=1}^n w_{ij}}$  // re-estimate mean
11     $\Sigma_i^t \leftarrow \frac{\sum_{j=1}^n w_{ij} (\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \mu_i)^T}{\sum_{j=1}^n w_{ij}}$  // re-estimate covariance
12     $P^t(C_i) \leftarrow \frac{\sum_{j=1}^n w_{ij}}{n}$  // re-estimate priors
13 until  $\sum_{i=1}^k \| \mu_i^t - \mu_i^{t-1} \|^2 \leq \epsilon$ 
```

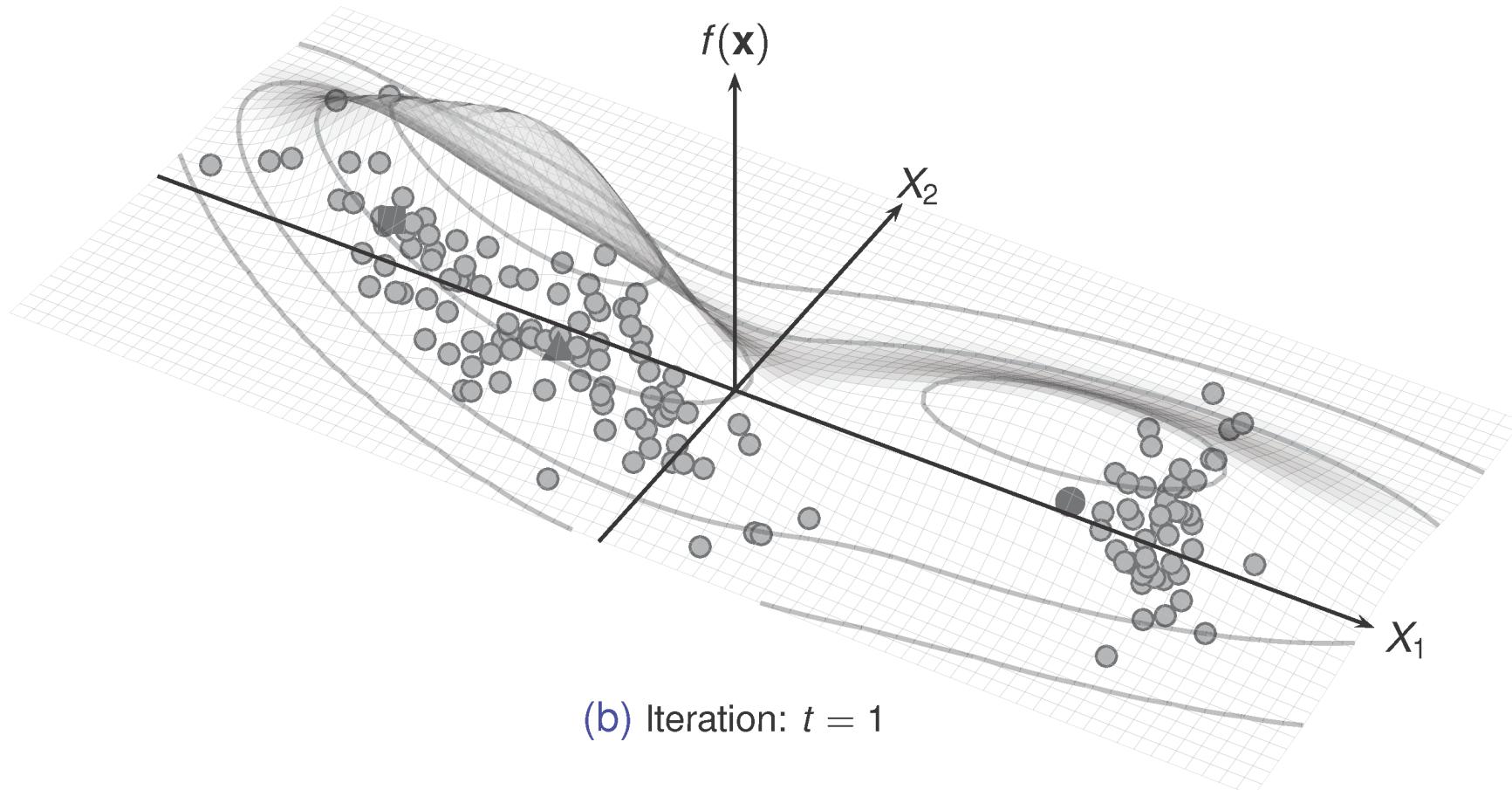
EM Clustering in 2D

Mixture of $k = 3$ Gaussians



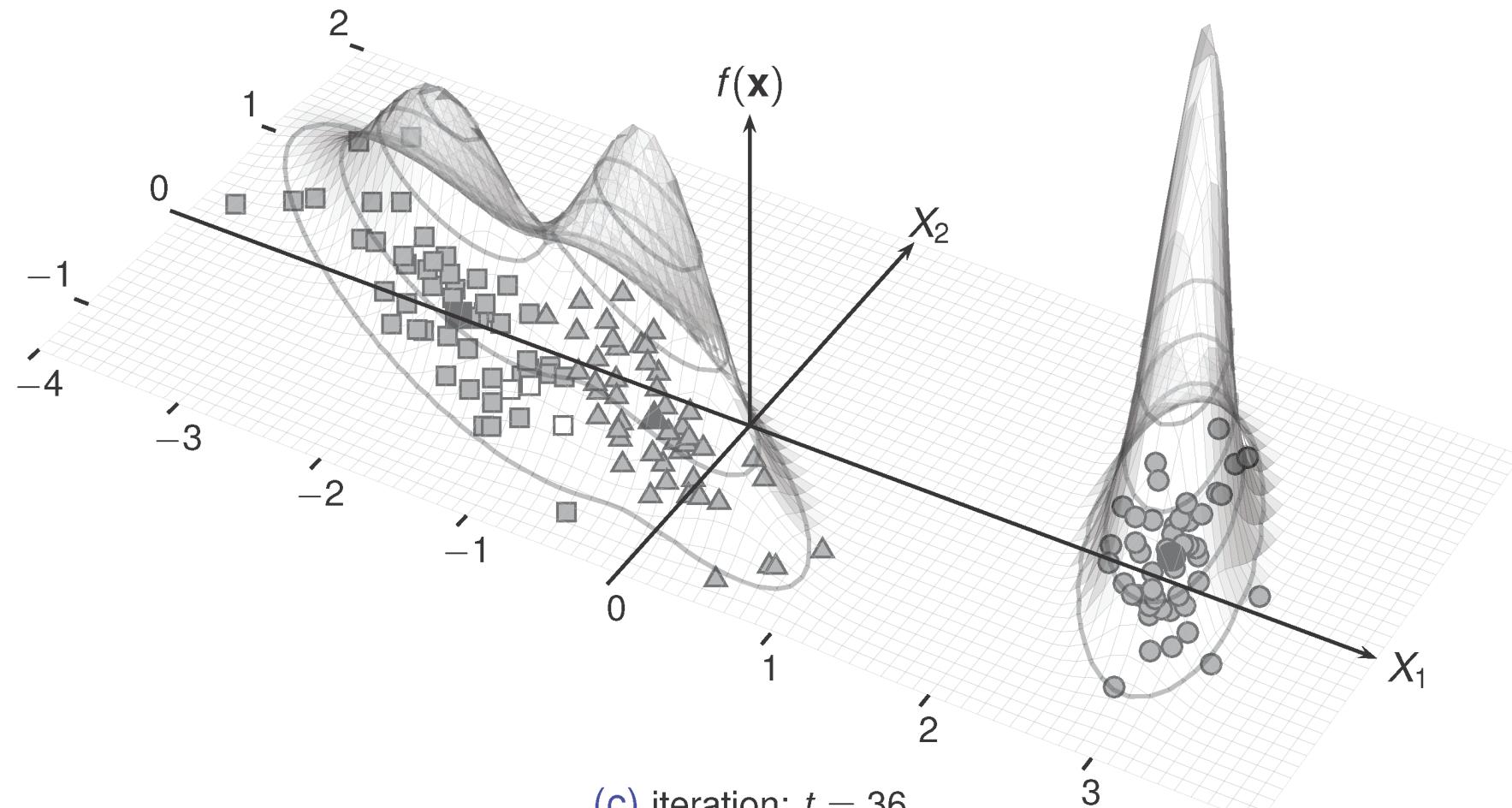
EM Clustering in 2D

Mixture of $k = 3$ Gaussians



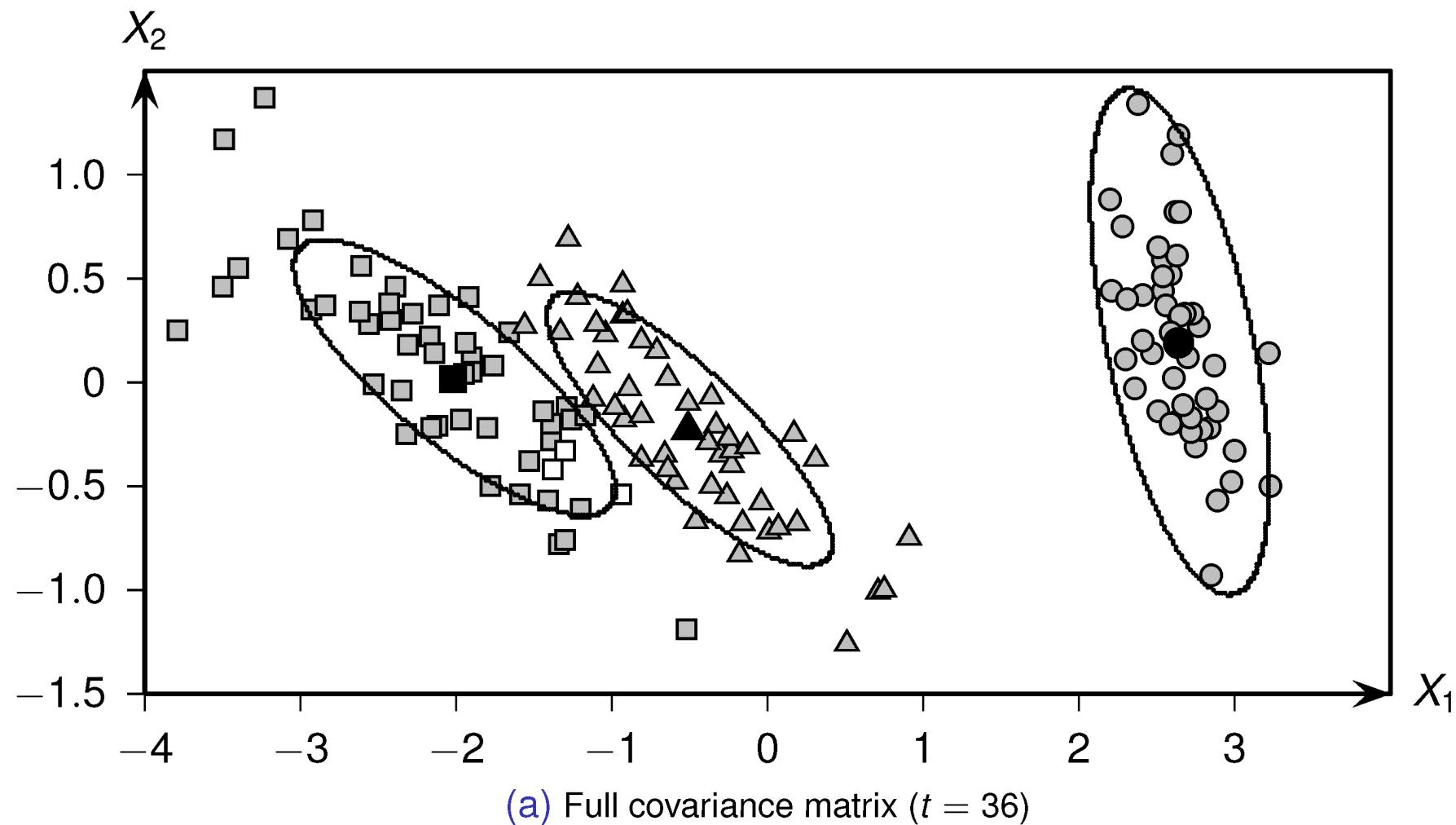
EM Clustering in 2D

Mixture of $k = 3$ Gaussians

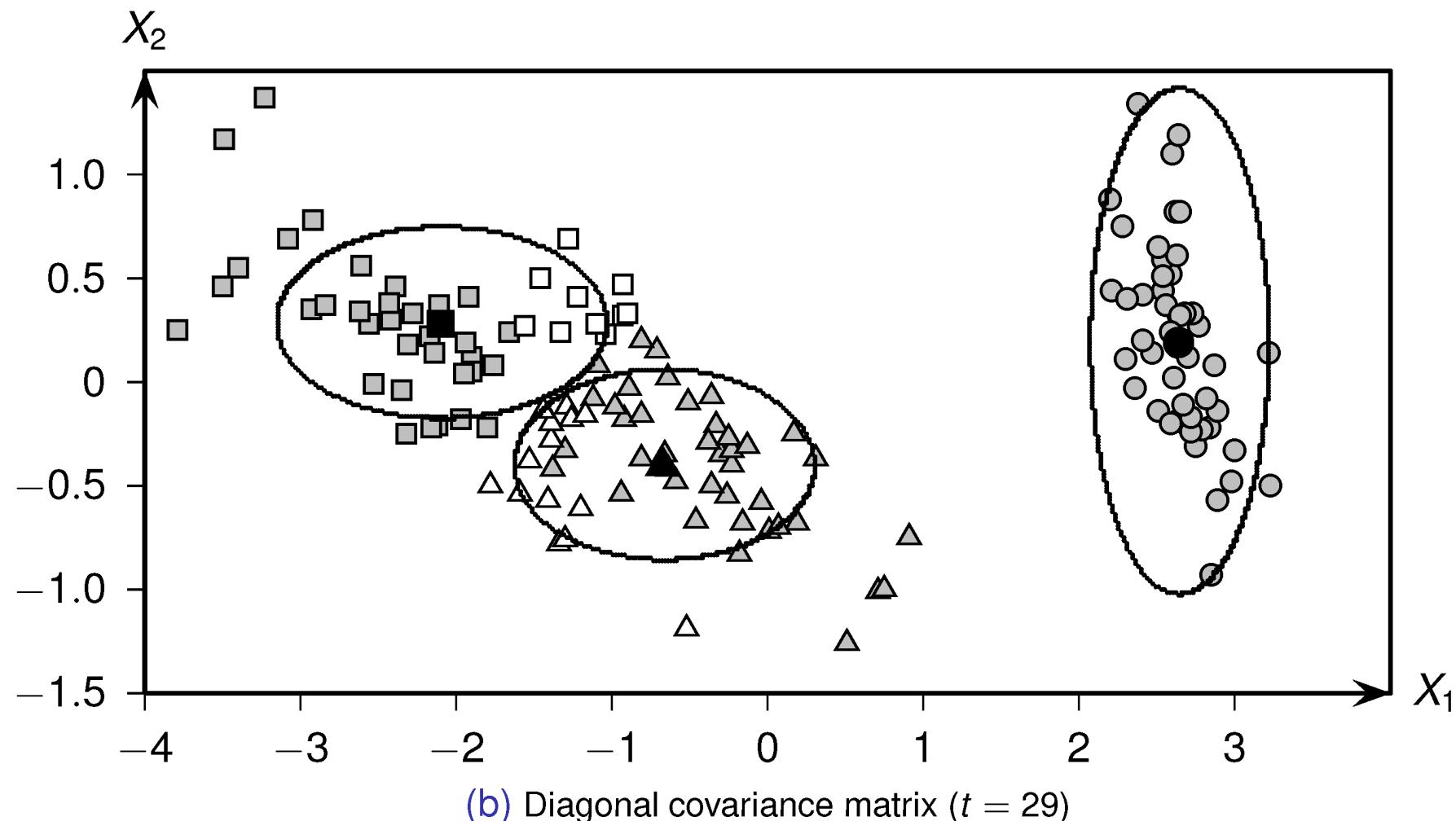


(c) iteration: $t = 36$

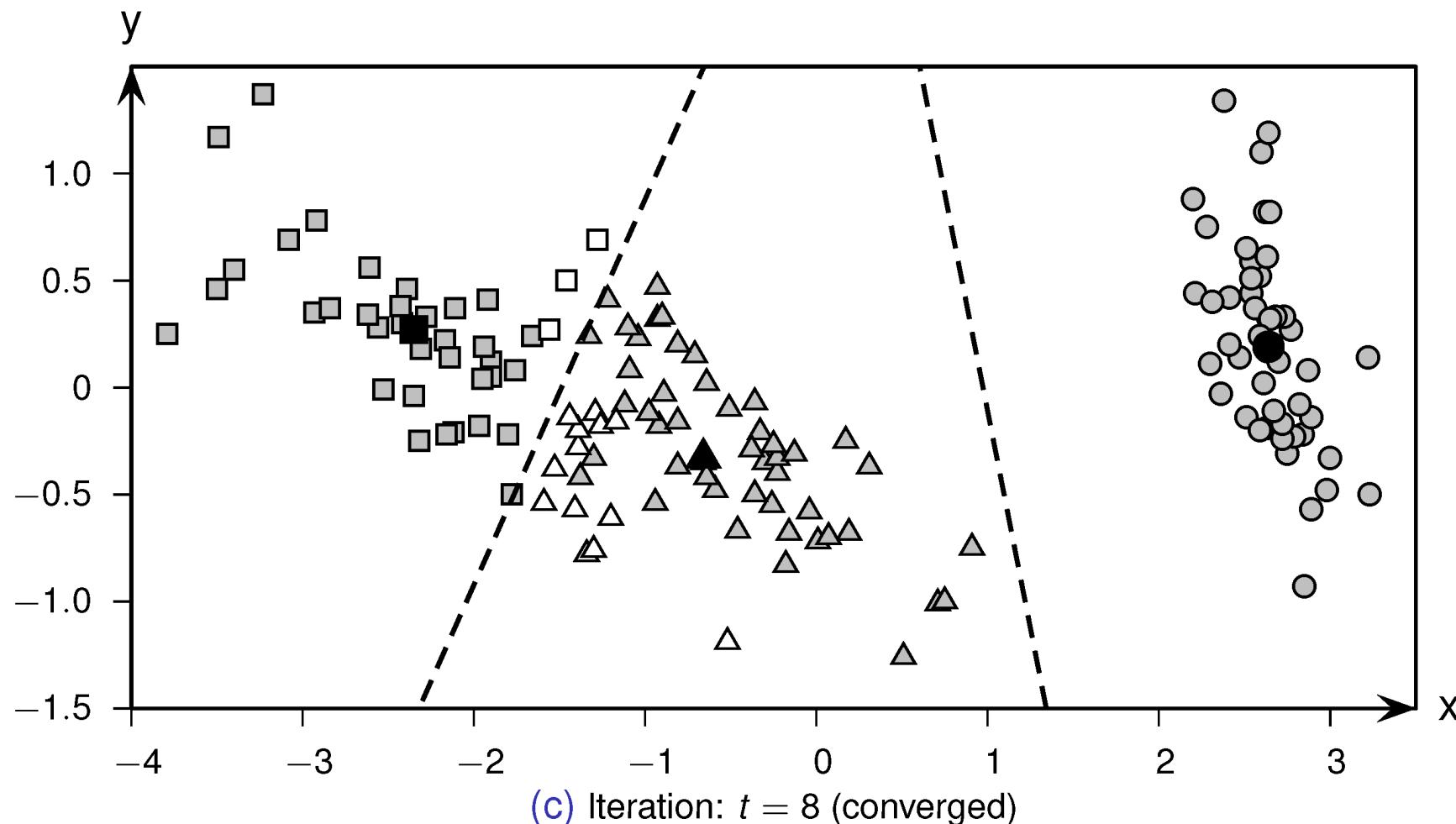
Iris Principal Components Data: Full Covariance Matrix



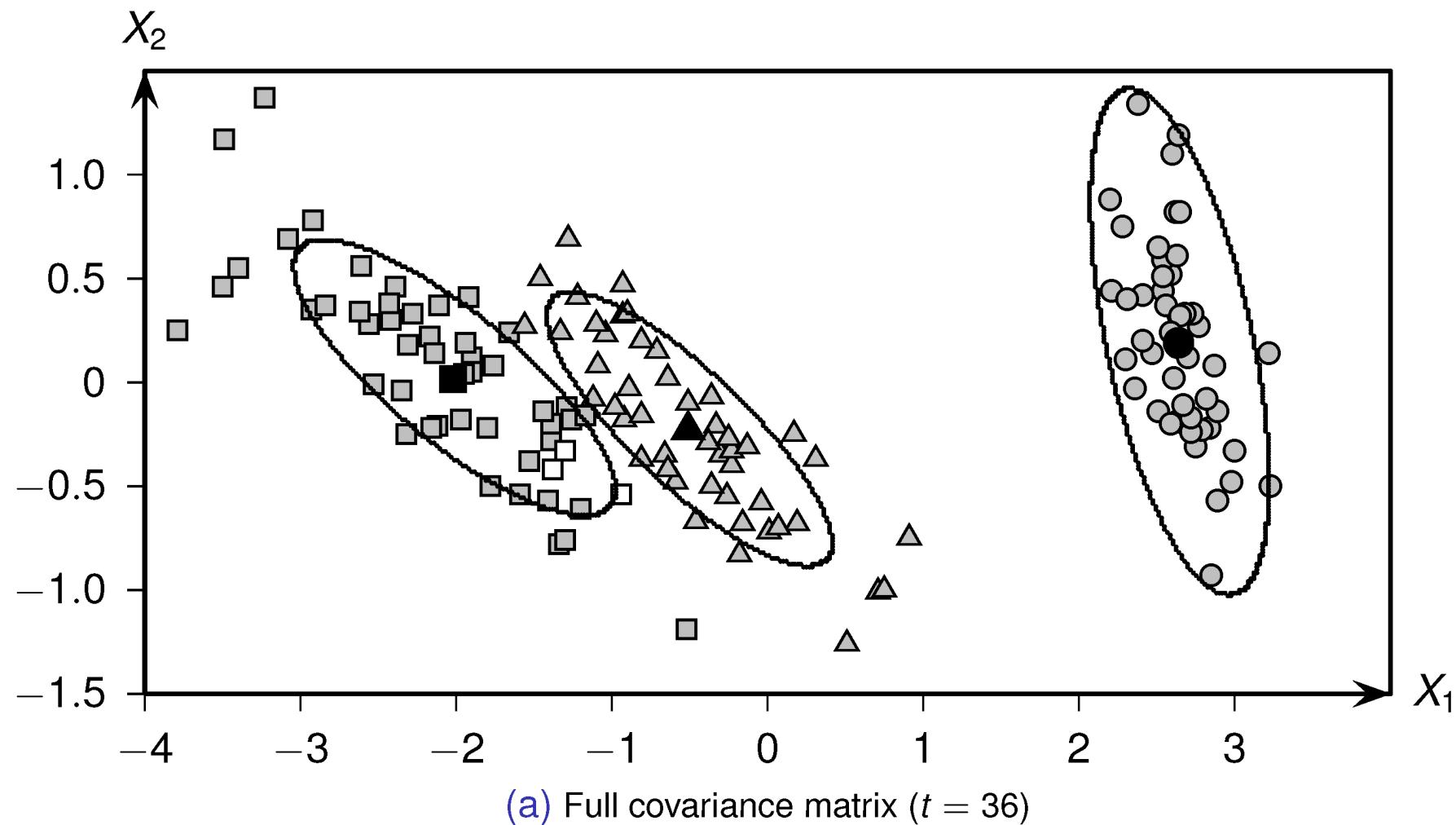
Iris Principal Components Data: Diagonal Covariance Matrix



K-means in 2D: Iris Principal Components



Iris Principal Components Data: Full Covariance Matrix



K-means as Specialization of EM

K-means can be considered as a special case of the EM algorithm, as follows:

$$P(\mathbf{x}_j|C_i) = \begin{cases} 1 & \text{if } C_i = \arg \min_{C_a} \left\{ \|\mathbf{x}_j - \boldsymbol{\mu}_a\|^2 \right\} \\ 0 & \text{otherwise} \end{cases}$$

The posterior probability $P(C_i|\mathbf{x}_j)$ is given as

$$P(C_i|\mathbf{x}_j) = \frac{P(\mathbf{x}_j|C_i)P(C_i)}{\sum_{a=1}^k P(\mathbf{x}_j|C_a)P(C_a)}$$

If $P(\mathbf{x}_j|C_i) = 0$, then $P(C_i|\mathbf{x}_j) = 0$. Otherwise, if $P(\mathbf{x}_j|C_i) = 1$, then $P(\mathbf{x}_j|C_a) = 0 \forall a \neq i$, and thus $P(C_i|\mathbf{x}_j) = \frac{1 \cdot P(C_i)}{1 \cdot P(C_i)} = 1$.

$$P(C_i|\mathbf{x}_j) = \begin{cases} 1 & \text{if } \mathbf{x}_j \in C_i, \text{i.e., if } C_i = \arg \min_{C_a} \left\{ \|\mathbf{x}_j - \boldsymbol{\mu}_a\|^2 \right\} \\ 0 & \text{otherwise} \end{cases}$$