K-means and Gaussian Mixture Models

CS145: Introduction to Data Mining

Spring 2024

Contents

1 Introduction **1**

2 K-means Clustering **1**

2.1 K-means Algorithm . . . 1

2.2 Objective Function: Within-Cluster Sum of Squares (WCSS) . . . 2

2.3 Iterative Optimization of WCSS . . . 2

2.4 Choosing the Number of Clusters (K) . . . 2

2.5 Limitations of K-means . . . 3

3 Gaussian Mixture Models (GMM) **3**

3.1 Mixture of Gaussians . . . 3

3.2 Maximum Likelihood Estimation . . . 3

3.3 Expectation-Maximization (EM) Algorithm . . . 4

3.4 Advantages of GMM over K-means . . . 5

3.5 Limitations of GMM . . . 5

4 Discussions **5**

A Rationale of K-means Algorithms **6**

B Derivation of E-step and M-step for GMM **6**

1 Introduction

Clustering is an unsupervised learning technique that aims to partition a dataset into groups (clusters) of similar objects. Two popular clustering algorithms are K-means and Gaussian Mixture Models (GMM). In this document, we will discuss these algorithms in detail, including their mathematical formulations, optimization techniques, and practical considerations.

2 K-means Clustering

K-means is a centroid-based clustering algorithm that aims to partition n data points into K clusters, where each data point belongs to the cluster with the nearest centroid (mean).

2.1 K-means Algorithm

The K-means algorithm can be summarized as follows:

Algorithm 1 K-means Algorithm

```
1: Initialize $K$ cluster centroids randomly or using a specific initialization scheme.
2: repeat
3:     Assignment Step: Assign each data point to the nearest centroid.
4:     Update Step: Update the centroids as the mean of the data points assigned to each cluster.
5: until Convergence or maximum iterations reached
```

2.2 Objective Function: Within-Cluster Sum of Squares (WCSS)

The objective of K-means is to minimize the Within-Cluster Sum of Squares (WCSS), also known as the squared error function:

$$\text{WCSS} = \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in C_k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2 \tag{1}$$

where C_k is the set of data points assigned to cluster k, and $\boldsymbol{\mu}_k$ is the centroid of cluster k.

2.3 Iterative Optimization of WCSS

K-means optimizes the WCSS objective function iteratively through the assignment step and the update step.

Assignment Step: In this step, each data point is assigned to the nearest centroid based on the Euclidean distance:

$$\text{Assignment :} \quad \underset{k}{\text{argmin}} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2 \tag{2}$$

By assigning each data point to the nearest centroid, the assignment step minimizes the WCSS for fixed centroids.

Update Step: In this step, the centroids are updated as the mean of the data points assigned to each cluster:

$$\text{Update :} \quad \boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i \tag{3}$$

where $|C_k|$ is the number of data points in cluster k. Updating the centroids to the mean of the assigned data points minimizes the WCSS for fixed assignments.

The assignment step and the update step are repeated until convergence (i.e., no change in assignments) or a maximum number of iterations is reached. Each iteration is guaranteed to decrease the WCSS, and the algorithm converges to a local minimum of the WCSS objective function.

2.4 Choosing the Number of Clusters (K)

The choice of the number of clusters (K) is a crucial hyperparameter in K-means. Some common methods for selecting K include:

- **Elbow Method:** Plot the WCSS as a function of K and choose the value of K at the "elbow" point, where the rate of decrease in WCSS slows down significantly.
- **Silhouette Analysis:** Compute the silhouette coefficient for different values of K and choose the K that maximizes the average silhouette coefficient. The silhouette coefficient measures how well a data point fits into its assigned cluster compared to other clusters.
- **Domain Knowledge:** Use prior knowledge or expert insight about the data to determine a suitable number of clusters.

2.5 Limitations of K-means

While K-means is a simple and widely used clustering algorithm, it has some limitations:

- K-means assumes that the clusters are spherical and have equal variances, which may not always be true for real-world data.
- The algorithm is sensitive to the initial placement of centroids and may converge to different local minima depending on the initialization.
- K-means requires the number of clusters (K) to be specified in advance, which may not be known or easily determined.
- The algorithm is not well-suited for discovering clusters with non-convex shapes or clusters of different sizes and densities.

3 Gaussian Mixture Models (GMM)

Gaussian Mixture Models (GMM) are probabilistic models that assume the data is generated from a mixture of K Gaussian distributions. GMM provides a more flexible and probabilistic approach to clustering compared to K-means.

3.1 Mixture of Gaussians

In GMM, the probability density function of a data point $\boldsymbol{x}$ is modeled as a weighted sum of K Gaussian densities:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{4}$$

where π_k is the mixing coefficient (prior probability) of the k-th component, $\boldsymbol{\mu}_k$ is the mean vector, and $\boldsymbol{\Sigma}_k$ is the covariance matrix. The mixing coefficients satisfy the constraints: $\sum_{k=1}^{K} \pi_k = 1$ and $\pi_k \geq 0$ for all k.

The multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ has the following probability density function:

$$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right) \tag{5}$$

where D is the dimensionality of the data, $|\boldsymbol{\Sigma}|$ denotes the determinant of the covariance matrix, and $\boldsymbol{\Sigma}^{-1}$ is the inverse of the covariance matrix.

3.2 Maximum Likelihood Estimation

Given a dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, the goal is to estimate the parameters of the GMM $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ that maximize the log-likelihood:

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right) \tag{6}$$

However, GMM cannot be directly optimized using gradient-based approaches due to the following reasons:

1. **Latent Variables:** The component assignments of data points are considered latent variables, which are not directly observed. The presence of latent variables makes the optimization problem more challenging, as the true assignments are unknown, and we need to marginalize over all possible assignments to compute the log-likelihood.

2. **Non-convexity:** The log-likelihood function of GMM is non-convex with respect to the parameters $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$. Non-convexity means that the log-likelihood function may have multiple local maxima, and gradient-based optimization methods can get stuck in suboptimal solutions depending on the initialization.

3. **Intractable Gradients:** Computing the gradients of the log-likelihood function with respect to the parameters is intractable due to the summation inside the logarithm, which prevents the direct calculation of the gradients, as the logarithm and summation operations do not commute.

4. **Identifiability Issues:** GMM has identifiability issues, meaning that different parameter values can lead to the same probability distribution. This can cause difficulties for gradient-based optimization methods, as they may not be able to distinguish between equivalent solutions.

The Expectation-Maximization (EM) algorithm is used to estimate the parameters iteratively, which circumvents these challenges by introducing latent variables and iteratively optimizing a lower bound on the log-likelihood function. The E-step computes the posterior probabilities of the latent variables given the current parameters, while the M-step updates the parameters by maximizing the expected complete-data log-likelihood. The EM algorithm guarantees a monotonic increase in the log-likelihood at each iteration and converges to a local maximum of the log-likelihood function.

3.3 Expectation-Maximization (EM) Algorithm

The Expectation-Maximization (EM) algorithm is an iterative approach for estimating the parameters of models with latent variables, such as Gaussian Mixture Models (GMM). The EM algorithm introduces latent variables $\boldsymbol{z}_i = (z_{i1}, \ldots, z_{iK})$ for each data point $\boldsymbol{x}_i$, where $z_{ik} = 1$ if $\boldsymbol{x}_i$ belongs to the k-th component, and $z_{ik} = 0$ otherwise.

In the standard formulation of GMM, the latent variables z_{ik} are assumed to be sampled from a categorical distribution, which is the prior distribution over $\boldsymbol{z}_i$. The categorical distribution is parameterized by the mixing coefficients π_k:

$$p(\boldsymbol{z}_i \mid \boldsymbol{\pi}) = \prod_{k=1}^{K} \pi_k^{z_{ik}} \tag{7}$$

where $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$ are the mixing coefficients satisfying $\sum_{k=1}^K \pi_k = 1$ and $\pi_k \geq 0$.

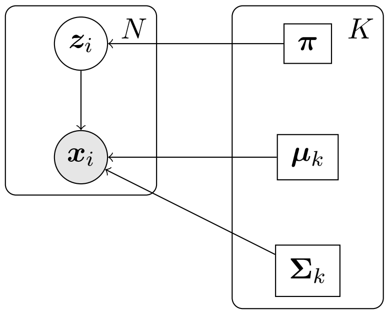

Figure 1: A graphical plate diagram of GMM.

However, it is worth noting that other prior distributions can be considered for $\boldsymbol{z}_i$ depending on the specific problem and assumptions. For example, in some variants of GMM, such as the Dirichlet Process Gaussian Mixture Model (DP-GMM), the prior distribution over $\boldsymbol{z}_i$ is modeled using a Dirichlet process prior, allowing for an infinite number of potential components. Another example is the Pitman-Yor process prior, which can capture power-law behavior in the cluster sizes. The choice of the prior distribution over $\boldsymbol{z}_i$

can impact the flexibility and complexity of the resulting GMM. The standard categorical prior distribution is the most commonly used choice in the basic formulation of GMM.

The EM algorithm consists of two main steps:

E-step: Compute the posterior probabilities (responsibilities) of each data point belonging to each component, given the current parameter estimates:

$$\gamma(z_{ik}) = p(z_{ik} = 1 \mid \boldsymbol{x}_i, \boldsymbol{\theta}^{\text{old}}) = \frac{\pi_k^{\text{old}} \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k^{\text{old}}, \boldsymbol{\Sigma}_k^{\text{old}})}{\sum_{j=1}^{K} \pi_j^{\text{old}} \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j^{\text{old}}, \boldsymbol{\Sigma}_j^{\text{old}})} \tag{8}$$

M-step: Update the parameters by maximizing the expected complete-data log-likelihood:

$$\pi_k^{\text{new}} = \frac{1}{N} \sum_{i=1}^{N} \gamma(z_{ik}) \tag{9}$$

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{\sum_{i=1}^{N} \gamma(z_{ik}) \boldsymbol{x}_i}{\sum_{i=1}^{N} \gamma(z_{ik})} \tag{10}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{\sum_{i=1}^{N} \gamma(z_{ik}) (\boldsymbol{x}_i - \boldsymbol{\mu}_k^{\text{new}})(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{\text{new}})^{\mathsf{T}}}{\sum_{i=1}^{N} \gamma(z_{ik})} \tag{11}$$

The EM algorithm alternates between the E-step and M-step until convergence or a maximum number of iterations is reached. Each iteration is guaranteed to increase the log-likelihood of the observed data.

3.4 Advantages of GMM over K-means

GMM offers several advantages compared to K-means:

- GMM provides a probabilistic assignment of data points to clusters, allowing for soft clustering and uncertainty quantification.
- GMM can model clusters with different sizes, shapes, and orientations by using different covariance matrices for each component.
- GMM is more flexible and can capture complex data distributions that K-means may struggle with.

3.5 Limitations of GMM

Despite its advantages, GMM also has some limitations:

- GMM requires the specification of the number of components (K), similar to K-means.
- The EM algorithm for GMM can be sensitive to initialization and may converge to suboptimal solutions.
- GMM assumes that the data is generated from a mixture of Gaussian distributions, which may not always be a valid assumption for real-world data.
- GMM is computationally more expensive than K-means, especially for high-dimensional data or a large number of components.

4 Discussions

K-means and Gaussian Mixture Models are two fundamental clustering algorithms with different assumptions and optimization techniques. K-means is a simple and widely used algorithm that aims to minimize the within-cluster sum of squares, while GMM is a probabilistic model that assumes the data is generated from a mixture of Gaussian distributions and provides a more flexible and probabilistic approach to clustering.

When choosing between K-means and GMM, consider the following factors:

- The shape, size, and orientation of the clusters in the data.
- The desired level of flexibility and probabilistic interpretation.
- The computational resources available and the scalability requirements.
- The validity of the Gaussian assumption for the data.

It is often beneficial to experiment with both algorithms and compare their results using evaluation metrics and domain knowledge. Preprocessing the data, handling missing values, and scaling the features are important steps before applying clustering algorithms.

Understanding the strengths, limitations, and optimization techniques of K-means and GMM is crucial for effectively applying them in various domains, such as customer segmentation, anomaly detection, and data compression.

A Rationale of K-means Algorithms

Assignment Step: The assignment step in K-means involves assigning each data point to the nearest centroid. This step minimizes the part of the WCSS that involves distances between data points and centroids for that iteration. Mathematically, it is represented as:

$$\text{Assignment:} \quad \underset{k}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2 \tag{12}$$

Update Step: The update step recalculates the centroids to be the mean of all data points assigned to each cluster. The derivation of this rule is based on minimizing the WCSS with respect to the centroids:

$$\frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{\boldsymbol{x}_i \in C_k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2 = 0 \tag{13}$$

$$\sum_{\boldsymbol{x}_i \in C_k} 2(\boldsymbol{x}_i - \boldsymbol{\mu}_k) = 0 \tag{14}$$

$$\sum_{\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i = \left(\sum_{\boldsymbol{x}_i \in C_k} 1 \right) \boldsymbol{\mu}_k \tag{15}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{\boldsymbol{x}_i \in C_k} \boldsymbol{x}_i}{|C_k|} \tag{16}$$

This step ensures that each centroid is located at the center of the cluster, effectively reducing the WCSS for the next iteration.

B Derivation of E-step and M-step for GMM

In the EM algorithm for GMM, the E-step and M-step are derived from the complete-data log-likelihood. The complete-data log-likelihood for GMM is given by:

$$\log p(\mathcal{D}, \boldsymbol{Z} \mid \boldsymbol{\theta}) = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} \log \left(\pi_k \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right) \tag{17}$$

where $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ is the observed data, $\boldsymbol{Z} = \{z_{ik}\}$ are the latent variables indicating the component assignments, and $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ are the parameters of the GMM.

E-step: In the E-step, we compute the posterior distribution of the latent variables $\boldsymbol{Z}$ given the observed data $\mathcal{D}$ and the current parameter estimates $\boldsymbol{\theta}^{\text{old}}$. This is done by calculating the responsibilities $\gamma(z_{ik})$, which represent the probability of data point $\boldsymbol{x}_i$ belonging to component k:

$$\begin{aligned}
\gamma(z_{ik}) &= p(z_{ik} = 1 \mid \boldsymbol{x}_i, \boldsymbol{\theta}^{\text{old}}) && (18)\\
&= \frac{p(\boldsymbol{x}_i, z_{ik} = 1 \mid \boldsymbol{\theta}^{\text{old}})}{p(\boldsymbol{x}_i \mid \boldsymbol{\theta}^{\text{old}})} && (19)\\
&= \frac{p(\boldsymbol{x}_i \mid z_{ik} = 1, \boldsymbol{\theta}^{\text{old}}) p(z_{ik} = 1 \mid \boldsymbol{\theta}^{\text{old}})}{\sum_{j=1}^{K} p(\boldsymbol{x}_i \mid z_{ij} = 1, \boldsymbol{\theta}^{\text{old}}) p(z_{ij} = 1 \mid \boldsymbol{\theta}^{\text{old}})} && (20)\\
&= \frac{\pi_k^{\text{old}} \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k^{\text{old}}, \boldsymbol{\Sigma}_k^{\text{old}})}{\sum_{j=1}^{K} \pi_j^{\text{old}} \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j^{\text{old}}, \boldsymbol{\Sigma}_j^{\text{old}})} && (21)
\end{aligned}$$

M-step: In the M-step, we maximize the expected complete-data log-likelihood with respect to the parameters $\boldsymbol{\theta}$. The expected complete-data log-likelihood is given by:

$$\begin{aligned}
Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) &= \mathbb{E}_{\boldsymbol{Z} \mid \mathcal{D}, \boldsymbol{\theta}^{\text{old}}}[\log p(\mathcal{D}, \boldsymbol{Z} \mid \boldsymbol{\theta})] && (22)\\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} \gamma(z_{ik}) \log \left(\pi_k \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) && (23)
\end{aligned}$$

To update the parameters, we maximize $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ with respect to each parameter:

Mixing Coefficients π_k: Maximizing $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ with respect to π_k subject to the constraint $\sum_{k=1}^{K} \pi_k = 1$ using the Lagrange multiplier method:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\pi}, \lambda) &= \sum_{i=1}^{N} \sum_{k=1}^{K} \gamma(z_{ik}) \log \pi_k + \lambda \left(1 - \sum_{k=1}^{K} \pi_k\right) && (24)\\
\frac{\partial \mathcal{L}}{\partial \pi_k} &= \sum_{i=1}^{N} \frac{\gamma(z_{ik})}{\pi_k} - \lambda = 0 && (25)\\
\pi_k &= \frac{1}{\lambda} \sum_{i=1}^{N} \gamma(z_{ik}) && (26)
\end{aligned}$$

Using the constraint $\sum_{k=1}^{K} \pi_k = 1$, we obtain:

$$\pi_k^{\text{new}} = \frac{\sum_{i=1}^{N} \gamma(z_{ik})}{N} \tag{27}$$

Mean Vectors $\boldsymbol{\mu}_k$: Maximizing $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ with respect to $\boldsymbol{\mu}_k$ by setting the gradient to zero:

$$\begin{aligned}
\frac{\partial Q}{\partial \boldsymbol{\mu}_k} &= \sum_{i=1}^{N} \gamma(z_{ik}) \frac{\partial}{\partial \boldsymbol{\mu}_k} \log \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) && (28)\\
&= \sum_{i=1}^{N} \gamma(z_{ik}) \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k) = 0 && (29)
\end{aligned}$$

Solving for $\boldsymbol{\mu}_k$:

$$\sum_{i=1}^{N} \gamma(z_{ik})(\boldsymbol{x}_i - \boldsymbol{\mu}_k) = 0 \tag{30}$$

$$\sum_{i=1}^{N} \gamma(z_{ik})\boldsymbol{x}_i = \left(\sum_{i=1}^{N} \gamma(z_{ik})\right) \boldsymbol{\mu}_k \tag{31}$$

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{\sum_{i=1}^{N} \gamma(z_{ik})\boldsymbol{x}_i}{\sum_{i=1}^{N} \gamma(z_{ik})} \tag{32}$$

Covariance Matrices $\boldsymbol{\Sigma}_k$: Maximizing $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}})$ with respect to $\boldsymbol{\Sigma}_k$ by setting the gradient to zero:

$$\frac{\partial Q}{\partial \boldsymbol{\Sigma}_k} = \sum_{i=1}^{N} \gamma(z_{ik}) \frac{\partial}{\partial \boldsymbol{\Sigma}_k} \log \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{33}$$

$$= \sum_{i=1}^{N} \gamma(z_{ik}) \left(-\frac{1}{2}\boldsymbol{\Sigma}_k^{-1} + \frac{1}{2}\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^{\mathsf{T}}\boldsymbol{\Sigma}_k^{-1}\right) = 0 \tag{34}$$

Solving for $\boldsymbol{\Sigma}_k$:

$$\sum_{i=1}^{N} \gamma(z_{ik})\boldsymbol{\Sigma}_k^{-1} = \sum_{i=1}^{N} \gamma(z_{ik})\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^{\mathsf{T}}\boldsymbol{\Sigma}_k^{-1} \tag{35}$$

$$\sum_{i=1}^{N} \gamma(z_{ik})\boldsymbol{\Sigma}_k = \sum_{i=1}^{N} \gamma(z_{ik})(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^{\mathsf{T}} \tag{36}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{\sum_{i=1}^{N} \gamma(z_{ik})(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{\text{new}})(\boldsymbol{x}_i - \boldsymbol{\mu}_k^{\text{new}})^{\mathsf{T}}}{\sum_{i=1}^{N} \gamma(z_{ik})} \tag{37}$$

The E-step and M-step are repeated iteratively until convergence or a maximum number of iterations is reached. The E-step computes the responsibilities $\gamma(z_{ik})$ based on the current parameter estimates, while the M-step updates the parameters $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ by maximizing the expected complete-data log-likelihood.