29 Gaussian Mixture Models — When One Bell Curve Isn’t Enough

29.1 From One Bell Curve to Many

29.1.1 What a Gaussian Really Means

A Gaussian Mixture Model (GMM) is a probabilistic model used for clustering and density estimation. It assumes that data are generated from a combination of several Gaussian (normal) distributions, each representing a hidden subgroup or cluster in the data. Every Gaussian component is defined by a mean vector, which determines its position, and a covariance matrix, which determines its spread and shape.

As shown in Figure 1, a univariate Gaussian distribution is fully determined by its mean and variance. Changing either parameter alters the height and width of the probability density function (PDF), but the familiar bell-shaped curve remains.

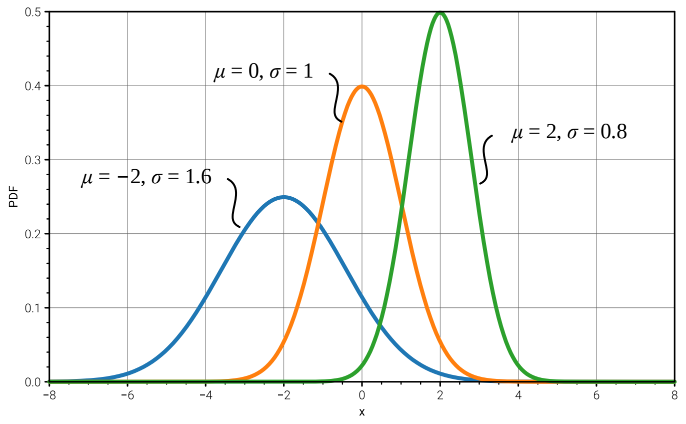


Figure 1. Three univariate Gaussian distributions with different means and variances.

29.1.2 Why One Gaussian Isn’t Enough

However, in real-world data, a single Gaussian distribution is often too simple to capture the complexity of the data’s structure. For example, in the Iris dataset, if we only look at the marginal distribution of sepal length (Figure 2 (a)), fitting one Gaussian (Figure 2 (b)) fails to represent the multimodal nature of the data. In contrast, using a mixture of multiple Gaussians (Figure 2 (c)) better approximates the true distribution. This intuition is closely related to kernel density estimation (KDE), which also relies on Gaussian kernels to approximate data density.

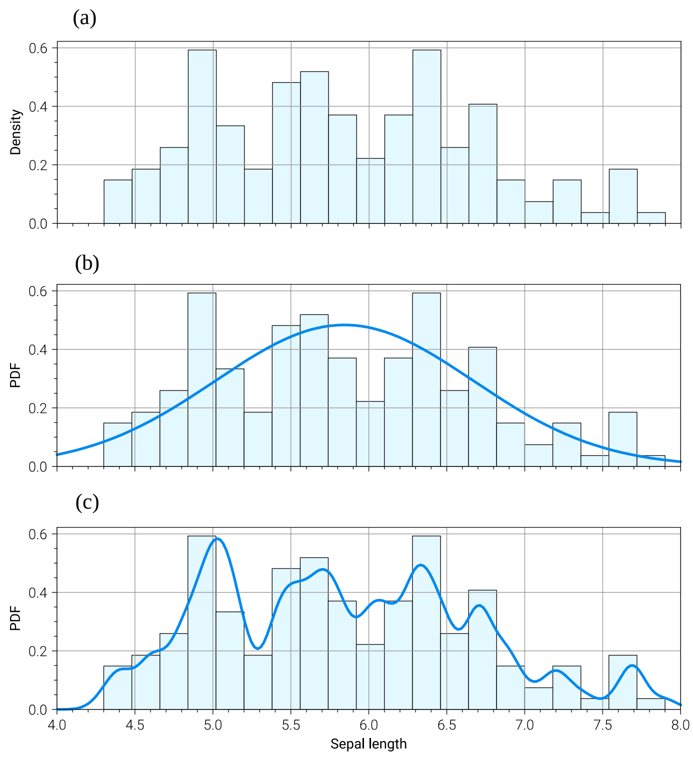


Figure 2. Comparison of (a) histogram, (b) single Gaussian fit, and (c) Gaussian kernel density estimation.

29.1.3 The Idea of a Mixture

The key idea behind a GMM is to represent the overall data distribution as a mixture of several Gaussian components, each corresponding to a potential cluster. In the univariate case (Figure 3), if we assume that the sepal length can be divided into three clusters, we can model it as the sum of three Gaussian components, each weighted by its relative importance.

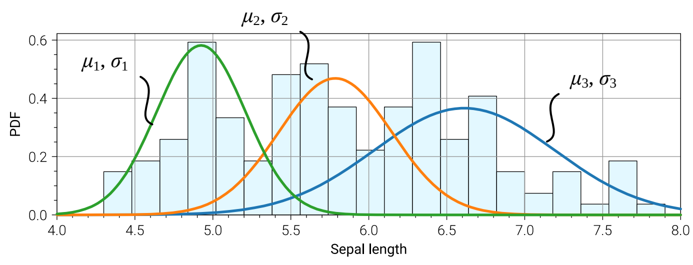


Figure 3. A Gaussian mixture model with three univariate components.

29.1.4 From Lines to Ellipses: Extending to Higher Dimensions

In the two-dimensional case (Figure 4), each Gaussian component is characterized by a mean vector (the cluster centroid) and a covariance matrix (the shape and orientation of the cluster). Together, these parameters describe ellipsoid-shaped contours in the data space.

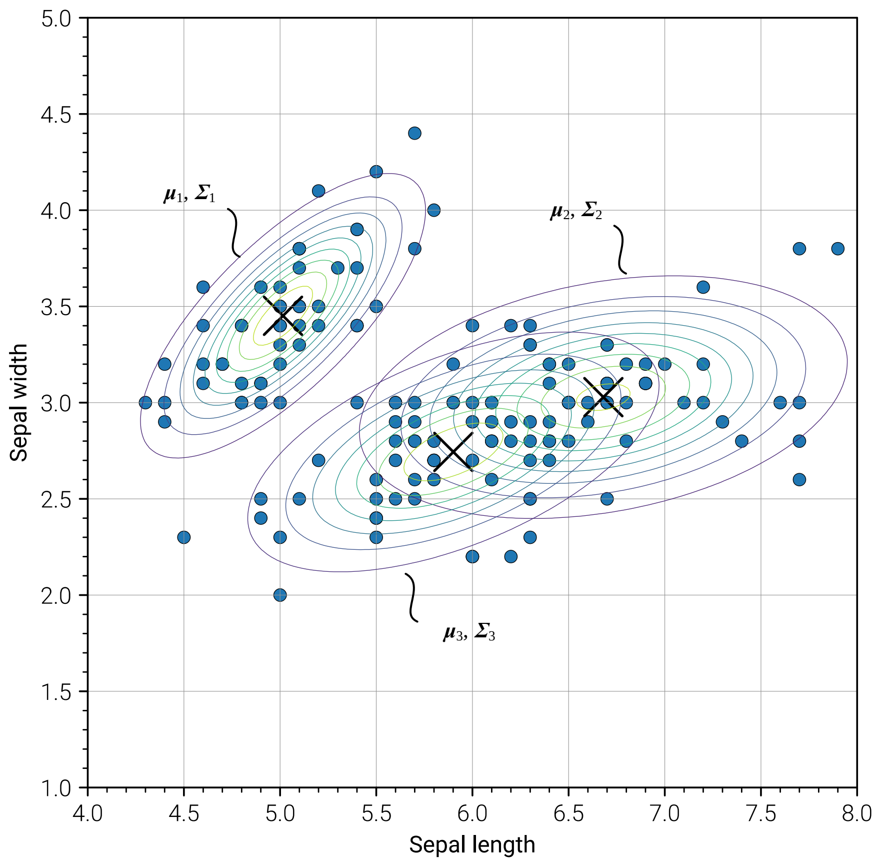


Figure 4. A Gaussian mixture model with three bivariate components.

In unsupervised learning settings, where data labels are unknown, GMMs infer the hidden cluster structure through an iterative estimation process. Each component has its own mean and covariance (defining its position and shape), and an associated mixing weight, which reflects how much of the total data it represents. In other words, fitting a GMM means estimating not only where each cluster is and how spread out it is, but also how important each cluster is in explaining the overall dataset.

It is important to distinguish Gaussian Mixture Models (GMMs) from Kernel Density Estimation (KDE). Although both rely on Gaussian distributions, their approaches differ fundamentally. KDE is a non-parametric method that places one Gaussian kernel on every data point and averages them equally to form a smooth density estimate. Its complexity grows with the number of samples, and its shape depends heavily on the choice of bandwidth.

In contrast, GMM is a parametric model that assumes data come from a finite number of Gaussian components. Each component corresponds to a cluster, and its parameters (mean, covariance, and weight) are learned using maximum likelihood estimation or the Expectation-Maximization (EM) algorithm.

In short, KDE can be thought of as “one Gaussian per data point,” while GMM represents “one Gaussian per cluster.”

29.2 Learning Hidden Structure — The EM Algorithm in Action

29.2.1 Why We Need EM?

Maximization (EM) algorithm. The EM algorithm is an iterative optimization method designed for problems involving incomplete or hidden data—in our case, the unknown cluster labels of each sample.

In the previous chapter, we learned that a GMM represents data as a weighted sum of multiple Gaussian distributions. Each Gaussian component has its own parameters: the mean and variance for the univariate case, or the mean vector and covariance matrix for the multivariate case. In addition, each component has a mixing coefficient, which indicates its relative contribution (or prior probability) within the overall model. Unfortunately, there is no closed-form analytical solution for estimating these parameters directly.

The EM algorithm provides a practical solution. It alternates between two main steps—Expectation (E-step) and Maximization (M-step)—to refine parameter estimates until convergence.

29.2.2 The Logic Behind Expectation and Maximization

At its core, EM seeks to maximize the likelihood function, which measures how probable the observed data are under the current model parameters. Because we do not know which component generated each sample, EM introduces the concept of latent variables representing these hidden cluster assignments.

- E-step (Expectation): Using the current parameter estimates $\boldsymbol{\theta}$, the algorithm computes the posterior probability that each data point belongs to each Gaussian component. Intuitively, this step estimates the degree of membership of each sample in every cluster.
- M-step (Maximization): Using those posterior probabilities as soft labels, the algorithm updates the parameters of each Gaussian—its mean, variance (or covariance), and mixing weight—by maximizing the expected log-likelihood.

These two steps are repeated until the parameter estimates or the log-likelihood value stops changing significantly, meaning the algorithm has converged.

29.2.3 A Step-by-Step Example of EM Learning

To make this concrete, consider a simple one-dimensional dataset that appears to contain two clusters (Figure 5). We set K = 2 to model it with two Gaussian components.

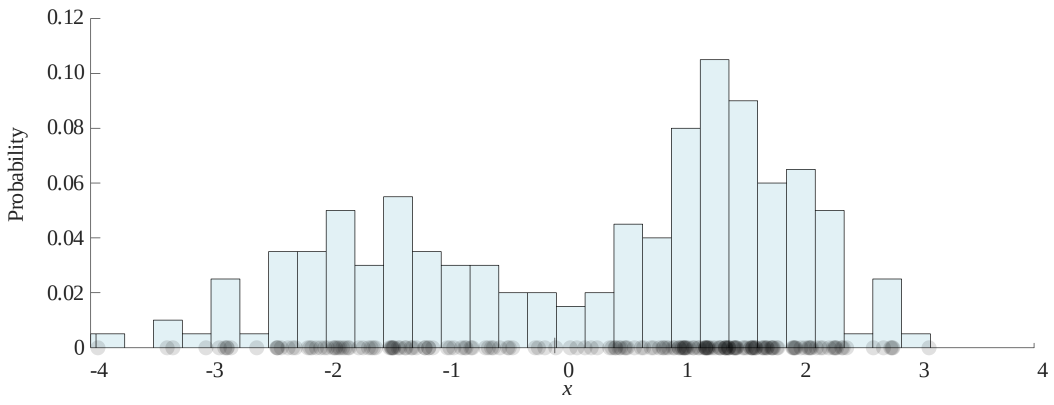


Figure 5. One-dimensional dataset to be clustered with GMM.

We begin by initializing the parameters $\boldsymbol{\theta}^{(0)}$, which include the mixing weights α_1, α_2, the means μ_1, μ_2 and the standard deviations σ_1, σ_2. These parameters define two initial Gaussian curves, as shown in Figure 6.

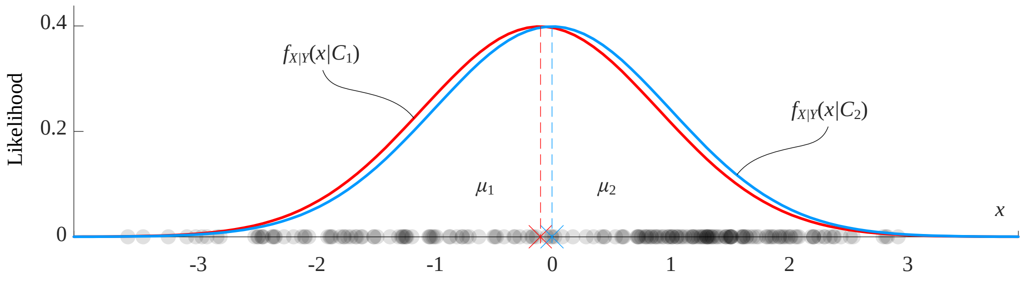


Figure 6. Initial Gaussian probability density functions $f_{X|Y}(x \mid C_1)$ and $f_{X|Y}(x \mid C_2)$

Next, we compute the marginal likelihood of each data point by summing the weighted likelihoods of both components. This gives us the overall probability density $f_X(x \mid \boldsymbol{\theta}^{(0)})$, shown in Figure 7, along with the joint probabilities $f_{X,Y}(x, C_1)$ and $f_{X,Y}(x, C_2)$. The colors along the x-axis indicate the most likely cluster assignment at this stage.

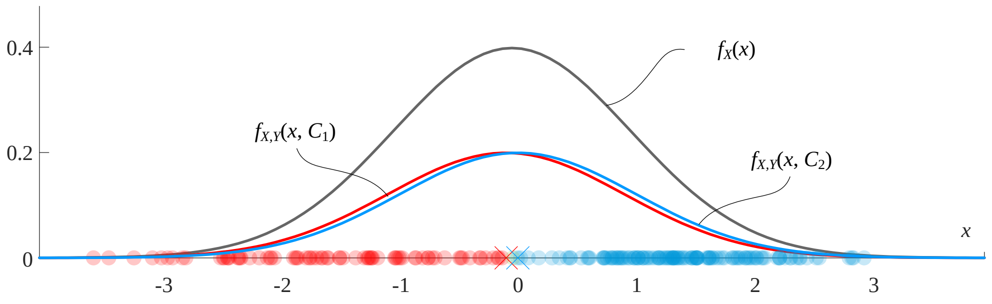


Figure 7. Initial joint probabilities and overall likelihood under parameters $\boldsymbol{\theta}^{(0)}$.

Using Bayes' theorem, we then compute the posterior probabilities $f_{Y|X}(C_1 \mid x)$ and $f_{Y|X}(C_2 \mid x)$ for each sample (Figure 8). These posteriors express how likely each point is to belong to each cluster. When these values are close together, cluster assignments are uncertain; when they differ sharply, the model is confident.

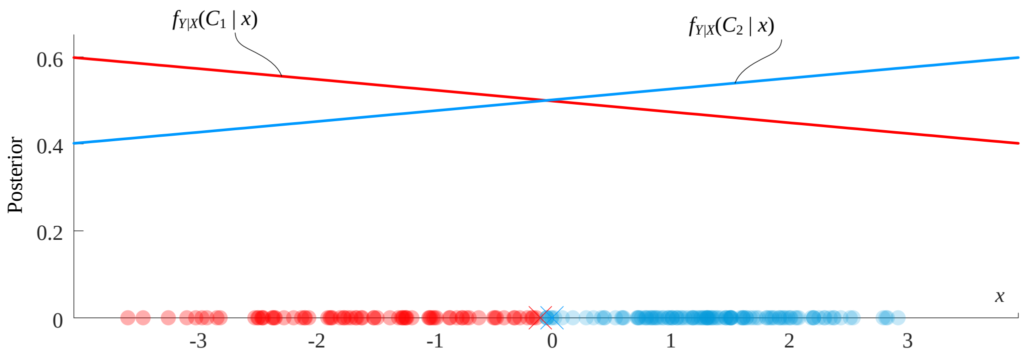


Figure 8. Posterior probabilities $f_{Y|X}(C_1 \mid x)$ and $f_{Y|X}(C_2 \mid x)$ under initial parameters.

In the M-step, we use these posterior probabilities as weights to update the parameters.

- Mixing coefficients α_1 and α_2 are updated based on the average membership of all samples in each cluster. They represent the proportion of data points attributed to each component.
- Means μ_1 and μ_2 are recalculated as weighted averages of the samples, using the posterior probabilities as weights.
- Variances (or standard deviations) σ_1, and σ_2 are similarly updated using weighted deviations from the new means.

After this update, we obtain a new parameter set $\boldsymbol{\theta}^{(1)}$. The algorithm then computes the log-likelihood function, which measures how well the new parameters explain the data. The E-step and M-step repeat, gradually improving the fit.

In this example, the algorithm runs multiple rounds of updates. After 12 iterations, the parameters $\boldsymbol{\theta}^{(12)}$ and the corresponding likelihood functions are shown in Figure 9 ~ Figure 11. The log-likelihood value increases to $L(\boldsymbol{\theta}^{(12)}) = -1.7344$.

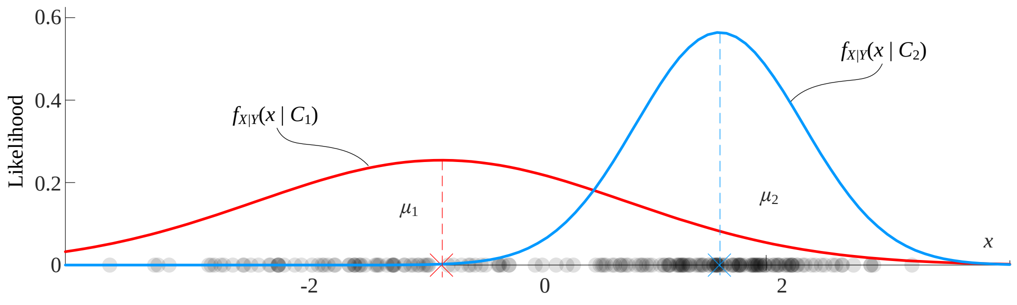


Figure 9. Likelihood functions $f_{X|Y}(x \mid C_1)$ and $f_{X|Y}(x \mid C_2)$ after 12 iterations.

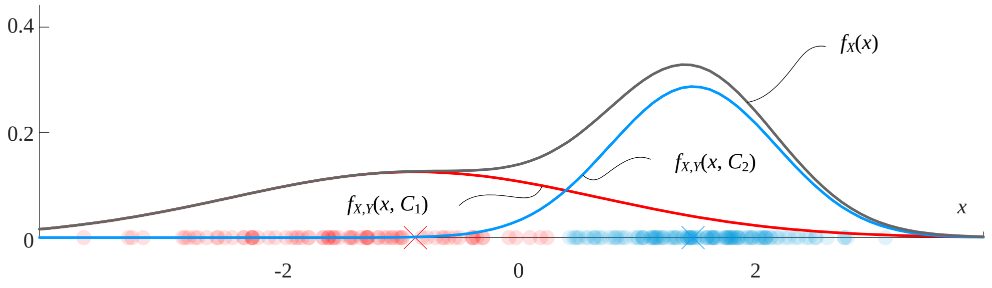


Figure 10. Joint and marginal probability densities after 12 iterations.

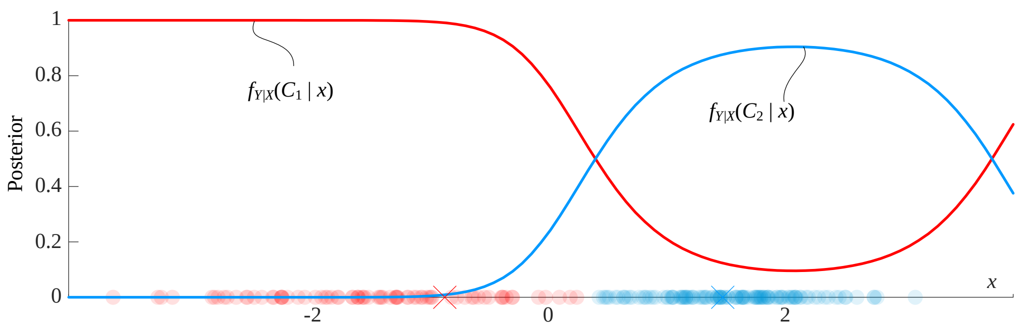


Figure 11. Posterior probabilities after 12 iterations.

29.2.4 When the Bells Settle — Convergence and Stability

After 36 iterations, the model further refines the parameters (Figure 12 ~ Figure 14), reaching $L(\boldsymbol{\theta}^{(12)}) = -1.7232$.

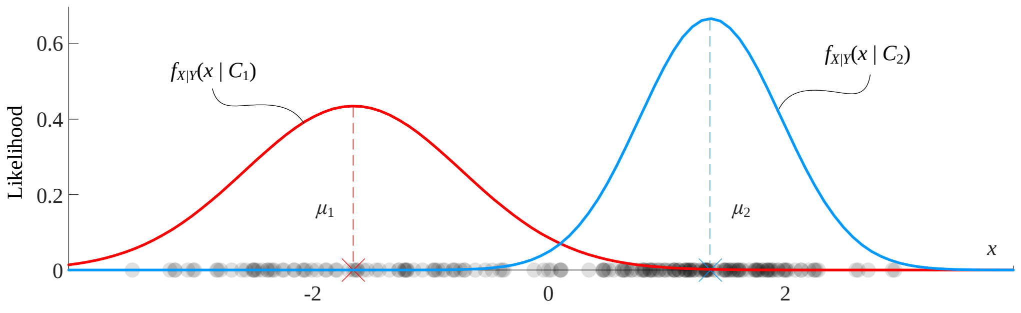


Figure 12. Likelihood functions $f_{X|Y}(x \mid C_1)$ and $f_{X|Y}(x \mid C_2)$ after 36 iterations.

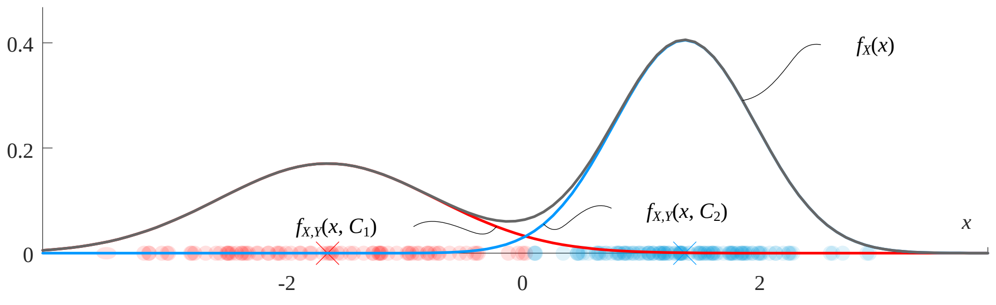


Figure 13. Joint and marginal probability densities after 36 iterations.

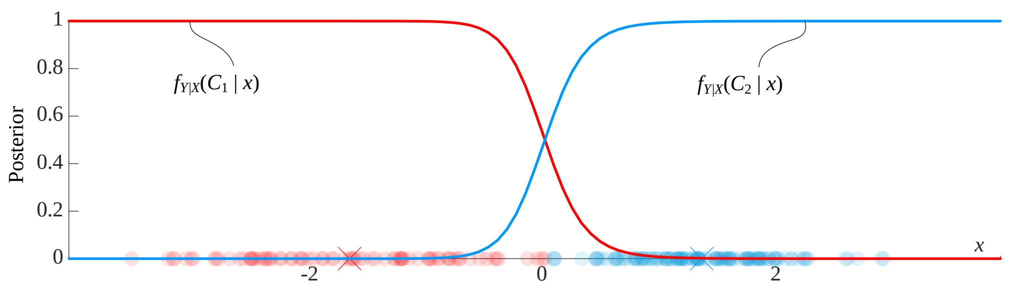


Figure 14. Posterior probabilities after 36 iterations.

Figure 15 shows how the log-likelihood increases monotonically during 36 iterations and eventually stabilizes after about 29 iterations, indicating convergence.

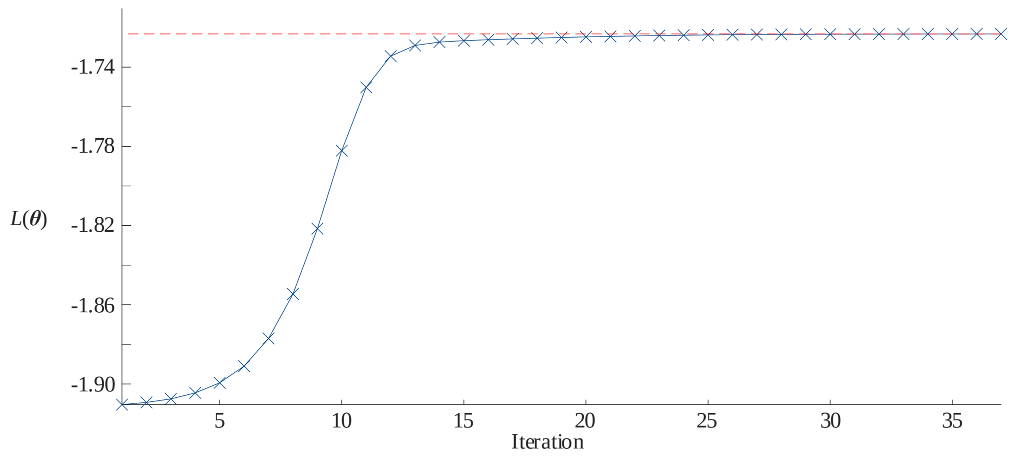


Figure 15. Convergence of the log-likelihood function over 36 iterations.

The EM algorithm refines the parameters of a Gaussian Mixture Model by alternating between estimating cluster memberships (E-step) and updating model parameters (M-step). Each iteration increases the log-likelihood, moving the model closer to the best fit for the observed data. Convergence occurs when both the parameters and the likelihood stabilize.

In essence, EM is a process of soft assignment and refinement: rather than assigning each point to a single cluster immediately, it gradually learns the most likely structure of the data by letting probabilities guide the learning.

29.3 Understanding Shape and Spread — The Role of the Covariance Matrix

29.3.1 From Circles to Ellipses: Geometry of Covariance

In a Gaussian Mixture Model (GMM), the overall data distribution can be viewed as a weighted combination of several Gaussian components. Each component represents a cluster and has its own mean, which defines the cluster’s center; a covariance matrix, which determines its shape and spread; and a mixing weight, which reflects how much that cluster contributes to the entire model.

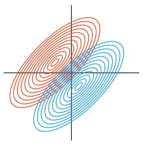
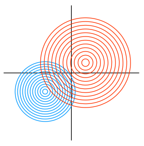
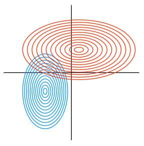
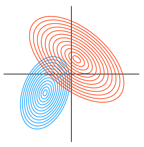
Unlike hard-clustering algorithms such as *k*-means, which assign each point to a single cluster, GMMs perform soft clustering. This means that every data point has a probability of belonging to each cluster. Some points may clearly belong to one cluster, while others lie between clusters and are assigned fractional memberships. This probabilistic nature allows GMMs to model overlapping and irregularly shaped clusters much more flexibly than *k*-means.

The covariance matrix plays a crucial role in defining the geometry of each Gaussian component. In multivariate Gaussian distributions, the covariance matrix determines how features vary together, shaping the orientation and elongation of the cluster’s density contour. A larger variance along one direction stretches the distribution, while correlations between features tilt it away from the axes.

29.3.2 Four Ways to Shape the Bells — Covariance Structures in GMMs

In the scikit-learn implementation of GMMs (sklearn.mixture.GaussianMixture), the parameter covariance_type controls how covariance matrices are modeled. Four types are supported—tied, spherical, diag, and full—each offering a different balance between flexibility and computational cost. Table 1 summarizes their main characteristics.

Table 1. Types of covariance structures in Gaussian Mixture Models

Type	Covariance matrix	Characteristics	PDF Contour Shape	Decision Boundary
tied	Shared (non-diagonal) across all components	Common covariance structure	Equal-sized rotated ellipses 	Linear
spherical	Independent, diagonal with equal variances	Isotropic (same variance in all directions)	Perfect circles 	Circular arcs
diag	Independent, diagonal with unequal variances	Features are uncorrelated but vary in scale	Axis-aligned ellipses 	Elliptic curves
full	Independent, full covariance matrix	No constraints (features may correlate)	Arbitrary ellipses 	General conic curves

Let’s briefly interpret each case:

- Tied covariance (tied) assumes that all clusters share a single covariance matrix that may include correlations between features. Since the shared matrix is non-diagonal, each component has an elliptical contour of equal shape and size. Because the same covariance applies everywhere, the quadratic terms in the decision function cancel out, leading to linear decision boundaries between clusters.

- Spherical covariance (spherical) gives each component its own diagonal covariance matrix with equal variances along all axes. This produces circular contours, meaning each cluster is equally spread in all directions, though with potentially different radii. The resulting decision boundaries are curved arcs separating these circular regions.

- Diagonal covariance (diag) allows each component to have its own diagonal covariance matrix with different variances per feature. The resulting contours are axis-aligned ellipses, where clusters can stretch more along some dimensions than others. The corresponding decision boundaries form elliptic conic sections.

- Full covariance (full) is the most general case—each component has a full covariance matrix with no restrictions. Clusters can have any orientation or correlation pattern among features, resulting in arbitrary elliptical contours and nonlinear decision boundaries.

29.3.3 Seeing the Difference — Visualizing Covariance Effects

To illustrate these differences, Figure 16 through Figure 19 show the clustering results on the Iris dataset under the four covariance settings. The contrasts between linear, circular, and elliptical decision boundaries highlight how covariance assumptions influence model flexibility and shape interpretation.

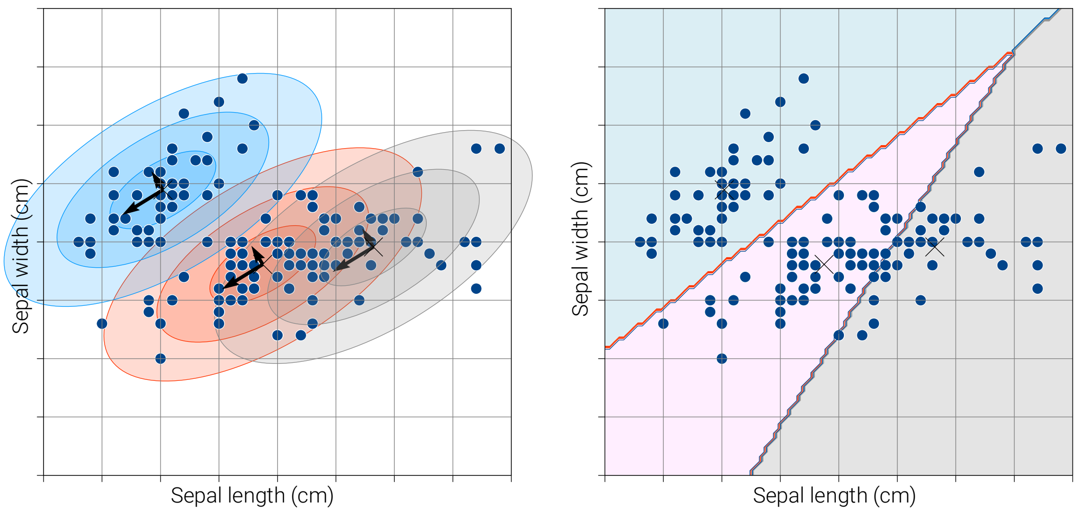


Figure 16. GMM clustering with covariance type 'tied' (shared elliptical contours and linear boundaries). Figure generated by Ch29_01_Gaussian_Mixture_Model.ipynb.

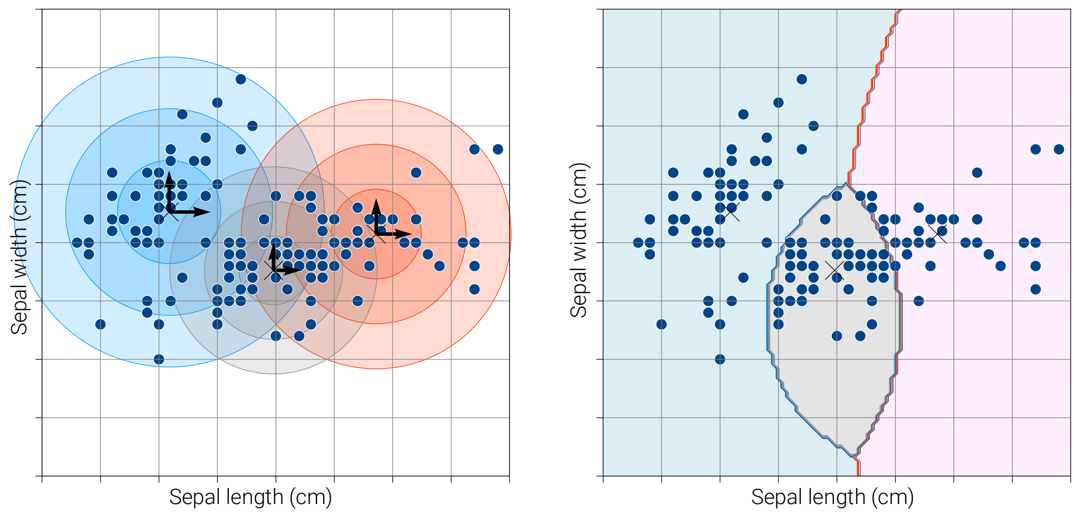


Figure 17. GMM clustering with covariance type 'spherical' (circular contours and curved boundaries). Figure generated by Ch29_01_Gaussian_Mixture_Model.ipynb.

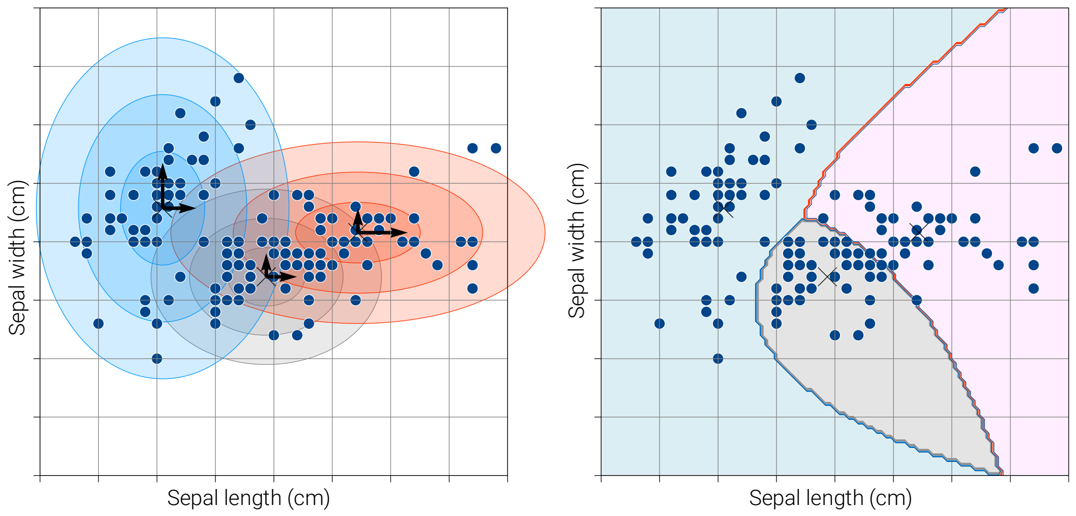


Figure 18. GMM clustering with covariance type 'diag' (axis-aligned ellipses and elliptic decision boundaries). Figure generated by Ch29_01_Gaussian_Mixture_Model.ipynb.

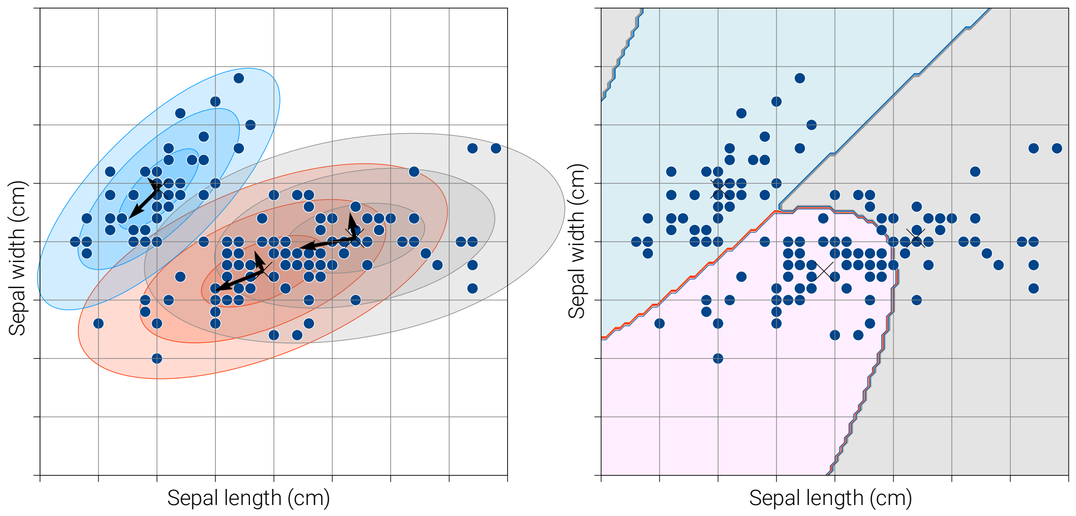


Figure 19. GMM clustering with covariance type 'full' (arbitrary ellipses and complex nonlinear boundaries). Figure generated by Ch29_01_Gaussian_Mixture_Model.ipynb.

29.4 Conclusion

A Gaussian Mixture Model is a probabilistic method for clustering and density estimation that assumes data are generated from several overlapping Gaussian distributions. Each Gaussian component has a mean, covariance, and weight, representing a potential cluster's position, shape, and importance. Unlike k-means, which performs hard assignments, GMM uses soft clustering, assigning probabilities of membership to each point.

The model parameters are learned through the Expectation-Maximization algorithm, which alternates between estimating cluster memberships and updating model parameters until convergence. The covariance matrix determines the orientation and spread of each cluster, allowing GMMs to model complex, non-spherical shapes.

Different covariance settings—tied, spherical, diagonal, and full—offer trade-offs between flexibility and computational efficiency. Overall, GMMs provide a powerful and flexible framework for discovering hidden structure in data and are widely used in pattern recognition, unsupervised learning, and probabilistic modeling.