Parameter estimation methods

🔍 What Are Parameter Estimation Methods?

Parameter Estimation Methods are statistical techniques used to determine the **values of unknown parameters** (like mean, variance, prior probabilities, etc.) of a probability model **from observed data**.

These methods help us **build models** that can:

- Predict outcomes
- Classify inputs into categories
- Recognize hidden structures in data (e.g., clusters)

📦 Goal of Parameter Estimation

🔍 Problem Setup

You are given:

- A **mathematical model** $p(x|\theta)$, which represents how data is *expected* to behave.
- A **set of observed data points**:

$$x_1, x_2, \ldots, x_n$$

The goal is to **estimate the parameters** θ of this model so that it fits the observed data as accurately as possible.

🎯 Why Estimation Is Needed

Let's say your model is:

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

Here, the parameters μ and σ^2 are **unknown**. You don't have access to the true values of these parameters, so you collect a dataset $x_1, x_2, \ldots, x_n$, and based on this dataset, you want to **guess the most suitable values** of μ and σ^2.

This "guessing" is **parameter estimation**.

🧠 Intuition

Imagine you have a duster, and you're trying to "fit" a bell curve (Gaussian) over the data points on your whiteboard.
You adjust μ (mean) and σ (spread) of the curve until the curve *matches* the data best.
That's **parameter estimation**.

✅ What Makes an Estimate "Best"?

Depending on the estimation method, "best" could mean:

- **Maximizing** the probability of seeing the data (Maximum Likelihood)
- **Minimizing** the expected loss (Bayesian Estimation)
- **Matching** sample statistics to population formulas (Method of Moments)

So:

> The **goal of parameter estimation** is to find parameter values $\hat{\theta}$ such that the model $p(x|\hat{\theta})$ describes the actual data as accurately as possible.

🎯 Why It Matters in Pattern Recognition

In pattern recognition:

- You may classify images, recognize handwriting, detect spam, etc.
- Each class (e.g., digit 0-9) has its own probability distribution (often Gaussian)
- You need to estimate the **parameters of these distributions** from training data

Without estimating parameters, **you can't construct discriminant functions**, **can't use Bayes' rule**, and **can't build accurate models**.

🔧 Common Parameter Estimation Methods

Method	Description	Use Case
Maximum Likelihood Estimation (MLE)	Finds parameters that **maximize the likelihood** of observing the given data	Most common; used for Gaussian, Poisson, Bernoulli
Bayesian Estimation	Combines **prior belief** with **observed data** to update parameter estimates	Used when prior knowledge is available or important
Method of Moments	Matches **sample moments** (mean, variance) with theoretical moments to solve for parameters	Simple but less optimal for complex models
Least Squares	Minimizes the sum of squared errors; commonly used in regression	Useful in linear/non-linear regression
Expectation-Maximization (EM)	Iteratively estimates parameters when **latent/hidden variables** are involved	Used in **Gaussian Mixture Models (GMMs)**, missing data

❓ Q1: Define Maximum Likelihood Estimation. Explain with an example.

✅ Definition:

Maximum Likelihood Estimation (MLE) is a statistical method used to **estimate the unknown parameters** of a probability distribution or model, by **maximizing the likelihood** that the observed data was generated by the model.

In other words:

MLE chooses the parameter values that make the **observed data most probable** under the assumed model.

📌 Formal Statement:

Let:

- $x_1, x_2, \ldots, x_n$ be independent observations from a probability distribution with unknown parameter θ
- The **likelihood function** is:

$$L(\theta) = \prod_{i=1}^{n} p(x_i|\theta)$$

- The **log-likelihood** is:

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log p(x_i|\theta)$$

Then, the **Maximum Likelihood Estimator (MLE)** is:

$$\hat{\theta} = \arg\max_{\theta} \ell(\theta)$$

🧮 Example: MLE for Mean of Gaussian Distribution

Suppose:

- Data $x = \{5, 6, 7\}$
- Assume data is from a Gaussian distribution:

$$x_i \sim \mathcal{N}(\mu, \sigma^2) \quad (\text{with known } \sigma^2 = 1)$$

✅ Step 1: Write the Likelihood Function

$$L(\mu) = \prod_{i=1}^{3} \frac{1}{\sqrt{2\pi}} e^{-\frac{(x_i-\mu)^2}{2}} = \left(\frac{1}{\sqrt{2\pi}}\right)^3 \cdot \exp\left(-\frac{1}{2}\sum_{i=1}^{3}(x_i-\mu)^2\right)$$

✅ Step 2: Maximize Log-Likelihood

$$\ell(\mu) = -\frac{3}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{3}(x_i-\mu)^2$$

Only the second term depends on μ, so minimize:

$$\sum(x_i-\mu)^2 \Rightarrow \hat{\mu} = \frac{1}{3}(5+6+7) = 6$$

MLE estimates the **mean** as 6 — the value that **maximizes the likelihood** of observing {5, 6, 7} under the Gaussian assumption.

Derive the Maximum Likelihood Estimators (MLE) for the Gaussian (normal) distribution

✅ Problem Setup

You are given:

- A dataset $x_1, x_2, \ldots, x_n$, assumed to be drawn i.i.d. from a **normal distribution**:

$$x_i \sim \mathcal{N}(\mu, \sigma^2)$$

Goal:

- Estimate μ and σ^2 using **Maximum Likelihood Estimation** (MLE)

🔹 Step 1: Write the PDF of Normal Distribution

For each observation x_i, the probability density function is:

$$p(x_i|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right)$$

🔹 Step 2: Construct the Likelihood Function

Since all x_i's are i.i.d., the joint likelihood is the product of individual densities:

$$L(\mu, \sigma^2) = \prod_{i=1}^{n} p(x_i|\mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right)$$

$$= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2\right)$$

🔹 Step 3: Take Log of the Likelihood (Log-Likelihood)

$$\ell(\mu, \sigma^2) = \log L(\mu, \sigma^2) = -\frac{n}{2} \log(2\pi) - \frac{n}{2} \log(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2$$

This is the **log-likelihood function**, which we will now **maximize** with respect to μ and σ^2.

🔹 Step 4: Derivative w.r.t. μ

Differentiate log-likelihood with respect to μ:

$$\frac{\partial \ell}{\partial \mu} = -\frac{1}{2\sigma^2} \cdot \frac{\partial}{\partial \mu} \left[\sum (x_i - \mu)^2\right] = -\frac{1}{2\sigma^2} \cdot \sum (-2)(x_i - \mu) = \frac{1}{\sigma^2} \sum (x_i - \mu)$$

Set derivative to zero:

$$\frac{1}{\sigma^2} \sum (x_i - \mu) = 0 \Rightarrow \sum x_i = n\mu \Rightarrow \boxed{\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i}$$

🔹 Step 5: Derivative w.r.t. σ^2

Differentiate log-likelihood with respect to σ^2:

$$\frac{\partial \ell}{\partial \sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2(\sigma^2)^2} \sum (x_i - \mu)^2$$

Set to zero:

$$-\frac{n}{2\sigma^2} + \frac{1}{2(\sigma^2)^2} \sum (x_i - \mu)^2 = 0$$

Multiply both sides by $2(\sigma^2)^2$:

$$-n\sigma^2 + \sum (x_i - \mu)^2 = 0 \Rightarrow \sigma^2 = \frac{1}{n} \sum (x_i - \mu)^2$$

Substitute $\hat{\mu}$:

$$\boxed{\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})^2}$$

🎉 Final MLE Estimators for Gaussian

Parameter	MLE Estimate
Mean μ	$\boxed{\hat{\mu} = \frac{1}{n} \sum x_i}$
Variance σ^2	$\boxed{\hat{\sigma}^2 = \frac{1}{n} \sum (x_i - \hat{\mu})^2}$

✅ Applications of These MLE Estimates

🔹 1. Modeling Class-Conditional Densities $p(x|\omega_i)$

In **Bayesian classification**, you often assume that data from each class ω_i follows a **normal distribution**.

But you don't know:

- The actual mean μ_i of that class
- The actual variance σ_i^2

✅ So you estimate:

$$\hat{\mu}_i = \frac{1}{n_i} \sum_{x_j \in \omega_i} x_j, \quad \hat{\sigma}_i^2 = \frac{1}{n_i} \sum_{x_j \in \omega_i} (x_j - \hat{\mu}_i)^2$$

Then you plug them into:

$$p(x|\omega_i) = \frac{1}{\sqrt{2\pi\hat{\sigma}_i^2}} \exp\left(-\frac{(x - \hat{\mu}_i)^2}{2\hat{\sigma}_i^2}\right)$$

🔹 2. Building Discriminant Functions

Once you have $p(x|\omega_i)$, and class priors $P(\omega_i)$, you compute:

$$g_i(x) = \ln p(x|\omega_i) + \ln P(\omega_i)$$

This helps in **Bayesian decision-making**: assign x to the class with the highest $g_i(x)$.

✅ To calculate this, you **must know** μ_i and σ_i^2 → which you estimate using MLE.

🔹 3. Clustering & GMMs

In **Gaussian Mixture Models (GMMs)**:

- Each component is a Gaussian with its own μ_k, σ_k^2
- EM algorithm uses MLE in its **M-step** to update those parameters

✅ So MLE is a building block in unsupervised learning methods too.

🔹 4. Simulation & Sampling

Once you've estimated μ and σ, you can:

- **Generate synthetic data** from the fitted Gaussian
- **Visualize** the probability density over a histogram
- **Compare** fit with other models

🔢 Example: Estimate Parameters and Compute Class Likelihood

🎯 Problem:

You are given **5 samples** from a class ω_1:

$$x = \{5, 7, 6, 9, 8\}$$

Assume these values are i.i.d. samples from a normal distribution:

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

✅ Task:

1. Estimate μ and σ^2 using MLE
2. Compute $p(x = 6.5|\omega_1)$ using the estimated parameters
3. Compute the **discriminant function** (optional)

🧮 Step 1: Compute MLE for Mean μ

$$\hat{\mu} = \frac{1}{n}\sum x_i = \frac{1}{5}(5 + 7 + 6 + 9 + 8) = \frac{35}{5} = \boxed{7}$$

🧮 Step 2: Compute MLE for Variance σ^2

$$\hat{\sigma}^2 = \frac{1}{n}\sum(x_i - \hat{\mu})^2 = \frac{1}{5}[(5-7)^2 + (7-7)^2 + (6-7)^2 + (9-7)^2 + (8-7)^2]$$

$$= \frac{1}{5}[4 + 0 + 1 + 4 + 1] = \frac{10}{5} = \boxed{2}$$

So, $\hat{\sigma} = \sqrt{2} \approx 1.414$

📊 Step 3: Compute $p(x = 6.5|\omega_1)$

Using the Gaussian PDF:

$$p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Plug in:

- $\mu = 7$
- $\sigma^2 = 2$, so $\sigma = \sqrt{2}$

$$p(6.5|\omega_1) = \frac{1}{\sqrt{2\pi \cdot 2}} \exp\left(-\frac{(6.5-7)^2}{2 \cdot 2}\right) = \frac{1}{\sqrt{4\pi}} \exp\left(-\frac{0.25}{4}\right)$$

$$= \frac{1}{2\sqrt{\pi}} \exp(-0.0625) \approx \frac{1}{2 \cdot 1.772} \cdot 0.939 \approx \frac{0.939}{3.544} \approx \boxed{0.265}$$

🎯 Given:

- Class ω_1
- Data samples: $x = \{5, 7, 6, 9, 8\}$
- Estimated using MLE:

$$\hat{\mu}_1 = 7, \quad \hat{\sigma}_1^2 = 2$$

- We want to **classify** the input point $x = 6.5$

Also assume:

- Prior probability: $P(\omega_1) = 0.5$ (equal class priors)

🧮 Step-by-Step Discriminant Function

For Gaussian case, the **discriminant function** is:

$$g_1(x) = \ln p(x|\omega_1) + \ln P(\omega_1)$$

You already computed:

$$p(6.5|\omega_1) \approx 0.265 \Rightarrow \ln p(6.5|\omega_1) = \ln(0.265) \approx -1.328$$

Also:

$$\ln P(\omega_1) = \ln(0.5) \approx -0.693$$

So:

$$g_1(6.5) = -1.328 - 0.693 = \boxed{-2.021}$$

🔢 Problem Statement:

You are given two classes:

Class ω_1:

- Samples: $x = \{4, 5, 6\}$

Class ω_2:

- Samples: $x = \{8, 9, 10\}$

Assume:

- Both classes follow a **Gaussian distribution**
- **Equal prior probabilities**:

$$P(\omega_1) = P(\omega_2) = 0.5$$

- You want to classify a test point: $x = 7$

✅ Step 1: Estimate Parameters via MLE

🔹 For ω_1:

- Mean:

$$\hat{\mu}_1 = \frac{4 + 5 + 6}{3} = 5$$

- Variance:

$$\hat{\sigma}_1^2 = \frac{1}{3}[(4-5)^2 + (5-5)^2 + (6-5)^2] = \frac{2}{3} \approx 0.667$$

🔹 For ω_2:

- Mean:

$$\hat{\mu}_2 = \frac{8 + 9 + 10}{3} = 9$$

- Variance:

$$\hat{\sigma}_2^2 = \frac{1}{3}[(8-9)^2 + (9-9)^2 + (10-9)^2] = \frac{2}{3} \approx 0.667$$

✅ Step 2: Compute Discriminant Functions at $x = 7$

🔹 General Formula (for Gaussian):

$$g_i(x) = \ln p(x|\omega_i) + \ln P(\omega_i)$$

Where:

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

So:

$$g_i(x) = -\frac{1}{2}\ln(2\pi\sigma_i^2) - \frac{(x-\mu_i)^2}{2\sigma_i^2} + \ln P(\omega_i)$$

Given:

- $\sigma_1^2 = \sigma_2^2 = 0.667$
- $\ln P(\omega_1) = \ln P(\omega_2) = \ln(0.5) \approx -0.693$

- $g_1(7)$:

$$g_1(7) = -\frac{1}{2}\ln(2\pi \cdot 0.667) - \frac{(7-5)^2}{2 \cdot 0.667} + \ln(0.5)$$

$$g_1(7) = -\frac{1}{2}\ln(4.1888) - \frac{4}{1.334} - 0.693 \approx -0.720 - 2.998 - 0.693 = \boxed{-4.411}$$

- $g_2(7)$:

$$g_2(7) = -\frac{1}{2}\ln(4.1888) - \frac{(7-9)^2}{1.334} - 0.693 = -0.720 - 2.998 - 0.693 = \boxed{-4.411}$$

😮 Interesting Observation:

Both $g_1(7)$ and $g_2(7)$ are **exactly equal**. Why?

➡️ Because:

- $x = 7$ lies **exactly halfway** between $\mu_1 = 5$ and $\mu_2 = 9$
- Both distributions have **equal variance**
- Priors are equal

So, the classifier is **indecisive** — you can assign randomly or use a tie-breaking rule.

✅ Advantages of MLE:

1. Intuitive and General Purpose

- MLE provides a simple and intuitive framework: choose the parameter values that make the observed data most probable.
- Works for a wide variety of parametric models.

2. Asymptotic Efficiency

- As sample size increases, MLE becomes **statistically efficient**:
  - It achieves **minimum variance** (Cramér-Rao lower bound)
  - It becomes **unbiased** and **normally distributed**

3. Consistency

- MLE produces estimators that **converge to the true parameter values** as the number of observations increases.

4. Foundation for Advanced Methods

- MLE is the base for:
  - Expectation-Maximization (EM) , Likelihood Ratio Tests
  - Logistic regression and neural networks

Transition : From MLE to GMM(Gaussian Mixture Model)

🔁 Recap: What MLE Does in the Gaussian Case

With **Maximum Likelihood Estimation (MLE)** for Gaussian distributions, you assumed:

- All your data points $x_1, x_2, \ldots, x_n$ came from a **single Gaussian distribution**:

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

We:

- Use the full dataset to compute:

$$\hat{\mu} = \frac{1}{n}\sum x_i, \quad \hat{\sigma}^2 = \frac{1}{n}\sum (x_i - \hat{\mu})^2$$

- Fit a **single bell-shaped curve** to all data

❌ But What If That Assumption Is Wrong?

Let's now take the case where your data **actually looks like this**:

Example:

You observe these 1D points:

$$\{3.1, 3.3, 3.4, 3.2, \quad 9.8, 10.2, 10.1, 9.9\}$$

Clearly:

- Half of the points are **clustered around 3.3**
- The other half are **clustered around 10**
- **No single Gaussian** can accurately fit both these regions at once

⚠️ Problem with MLE in This Case

Problem	Explanation
Wrong mean	The single MLE mean $\hat{\mu}$ will lie **somewhere between 3 and 10**, which is **not representative** of either true cluster
High variance	The MLE will report a **large** $\hat{\sigma}^2$ to cover both clusters — leading to a **flat, wide bell curve**
Poor classification	If used in pattern recognition, the model will be **confused** about the true cluster of any new point
Bad density estimation	The probability density function will not match the actual distribution — **multimodality is lost**

🧠 **Why This Happens?**

Because: MLE treats all data as coming from the same source

MLE does **not** know that there are **multiple hidden groups (clusters)** generating the data:

- There is no **mechanism** in MLE to model **latent variables**
- It tries to summarize all data using **a single parameter set**

📌 Real-World Analogy

Imagine you're fitting a **bell curve to the height distribution** of people at a college.

- But your data includes **men and women**
- They form **two overlapping but distinct groups**

If you use **MLE with a single Gaussian**, you get:

- A mean height that **doesn't represent either group**
- A wide variance that **blurs the real structure**

➡️ You lose valuable structure because **MLE cannot separate the groups**.

✅ What You Need Instead: A Mixture Model

You need:

- **Two separate Gaussians**: one for men, one for women
- A way to **assign soft responsibility** of each data point to each Gaussian
- Estimate parameters **individually** for each group

That's exactly what **GMM + EM** allows you to do.

Gaussian Mixture Model (GMM)

A **Gaussian Mixture Model** models a probability distribution as a **weighted sum of multiple Gaussian (normal) distributions**:

$$p(x) = \sum_{k=1}^{K} w_k \cdot \mathcal{N}(x \mid \mu_k, \Sigma_k)$$

Where:

- $x \in \mathbb{R}^d$: a **d-dimensional** data point (for 2D or higher)
- K: number of Gaussian components (clusters)
- w_k: weight or **mixing coefficient** for the k-th component
- μ_k: **mean vector** of the k-th Gaussian
- Σ_k: **covariance matrix** of the k-th Gaussian

Subject to the constraint:

$$\sum_{k=1}^{K} w_k = 1, \quad w_k \geq 0$$

◆ K: Number of Components

This defines **how many Gaussians** you are mixing.

- Each component captures one **cluster**
- Choosing K properly is critical (via heuristics or algorithms like BIC)

◆ w_k: Mixing Coefficients

These are **prior probabilities** (like class priors in Bayesian classification).

- They represent the **probability that a data point belongs to the k-th component**:

$$w_k = P(z = k)$$

- Must satisfy:

$$w_k \geq 0, \quad \sum_{k=1}^{K} w_k = 1$$

- If $w_1 = 0.6$, $w_2 = 0.4$, it means:
  - 60% of the data is expected from component 1
  - 40% from component 2

◆ $\mathcal{N}(x \mid \mu_k, \Sigma_k)$: Gaussian PDF

This is the **probability density function** of the k-th Gaussian:

For multivariate case:

$$\mathcal{N}(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k)\right)$$

- μ_k: the center (mean) of the k-th cluster
- Σ_k: the shape, orientation, and spread of that cluster
  - Diagonal Σ = uncorrelated dimensions
  - Full Σ = arbitrary ellipses

🔹 $p(x)$: Overall Probability Density Function

This is the **total density function** — the output of the GMM:

$$p(x) = \sum_{k=1}^{K} w_k \cdot \mathcal{N}(x|\mu_k, \Sigma_k)$$

It computes the probability of observing a point x, **regardless of which component generated it**.

Think of this as a **weighted average of multiple bell curves**.

🧠 Interpretation of GMM

- Each data point xxx is **assumed to be generated** by one of the KKK Gaussians.
- But we don't know **which one** — instead, each component gives a **soft assignment** (probability of belonging).
- GMM provides **flexible modeling** of real-world, multimodal distributions.

📊 Geometric View

Each Gaussian in the mixture:

- Forms an **elliptical cluster** in space (2D/3D)
- GMM overlays all those clusters
- The resulting surface is a **smooth, multimodal probability density**

⚔️ Comparison: Single Gaussian vs Gaussian Mixture Model

Feature	Single Gaussian (MLE)	Gaussian Mixture Model (GMM)
Assumption	All data is generated from **one Gaussian**	Data is generated from **a mixture of K Gaussians**
Can model multimodal data?	❌ No (only 1 peak)	✅ Yes (multiple peaks)
Clustering capability	❌ No (unsuitable for clustering)	✅ Yes (soft clustering possible)
Latent variable (cluster label)	❌ Not modeled	✅ Modeled implicitly via responsibilities γik\gamma_{ik}
Used for	Density estimation, simple classification	Clustering, density estimation for complex distributions
MLE solution	Closed-form solution	MLE is **intractable**, solved using **EM algorithm**
Visual	One bell curve	Weighted sum of multiple bell curves

📌 When to Use What?

Scenario	Use...
Data is **unimodal and symmetric**	Single Gaussian
Data shows **multiple peaks/clusters**	GMM
Need to **classify or model unknown groups**	GMM + EM
Only interested in a **simple distribution fit**	Single Gaussian (MLE)

MLE with a single Gaussian is **simple and fast** but assumes **homogeneous data**.
GMM is a **powerful generalization** that allows modeling **real-world, multimodal, and structured datasets** — at the cost of more complexity and computation.

🧠 Summary of Formula-Level Difference

- **Single Gaussian:**

$$p(x) = \mathcal{N}(x|\mu, \Sigma) \quad \text{(one peak)}$$

- **GMM:**

$$p(x) = \sum_{k=1}^{K} w_k \cdot \mathcal{N}(x|\mu_k, \Sigma_k) \quad \text{(many peaks, each weighted)}$$

The Problem with GMM

You are given:

- A dataset $\mathcal{D} = \{x_1, x_2, ..., x_n\} \in \mathbb{R}^d$
- You assume it was generated from a **mixture of K Gaussian components**

So, the probability model is:

$$p(x) = \sum_{k=1}^{K} w_k \cdot \mathcal{N}(x|\mu_k, \Sigma_k)$$

Where:

- w_k: mixing weights (with $\sum w_k = 1$)
- μ_k: mean of the k-th Gaussian
- Σ_k: covariance of the k-th Gaussian

🎯 Goal:

Estimate the full parameter set:

$$\theta = \{w_k, \mu_k, \Sigma_k\}_{k=1}^{K}$$

by **maximizing the likelihood**:

$$L(\theta) = \prod_{i=1}^{n} \sum_{k=1}^{K} w_k \cdot \mathcal{N}(x_i | \mu_k, \Sigma_k)$$

❌ Why You Cannot Use MLE Directly

🔸 1. Log-Likelihood Has a Log-of-Sum Structure

MLE maximizes the **log-likelihood**:

$$\log L(\theta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} w_k \cdot \mathcal{N}(x_i | \mu_k, \Sigma_k) \right)$$

This expression has **no closed-form solution** for maximizing with respect to μ_k, Σ_k, or w_k because:

- The **logarithm of a sum** cannot be simplified
- You can't decouple the contributions of different components

🔸 2. Unknown Component Labels (Latent Variables)

Each data point x_i was generated by **one of the K components**, but:

- You do **not know which one** → this is a **hidden (latent) variable**
- Standard MLE expects **fully observed data**, but here the labels z_i are missing

🔸 3. Responsibility Is Unclear

You don't know:

- Which Gaussian generated x_i
- What **portion of** x_i is attributable to each component

This causes **MLE gradients to become entangled**, and you **cannot isolate terms** for individual components

You cannot:

- Separate the log-likelihood into **independent terms**
- Use the usual MLE formulas for each μk , Σk
- Solve the optimization analytically

✅ The Solution: Use the EM Algorithm

To handle these challenges, we:

- Introduce **latent variables** z_i representing the unknown component for each x_i
- Use **Expectation-Maximization (EM)** to:
  - **E-step**: Estimate the **responsibilities** $\gamma_{ik} = P(z_i = k|x_i)$
  - **M-step**: Maximize the **expected complete-data log-likelihood**

EM **alternates** between:

- Computing soft cluster assignments (like probabilities)
- Updating parameters based on these assignments

🧠 What Are Latent Variables in GMM?

🔸 Definition:

A **latent variable** is a variable that **we do not observe directly**, but it affects the data generation process.

In GMMs:

- Each data point x_i is **generated by one of the K Gaussians**
- But we do **not know which one**
- So we introduce a **latent (hidden) variable** z_i, where:

$$z_i \in \{1, 2, ..., K\}$$

It tells us:

> "Which component Gaussian generated x_i?"

🔹 Representation of Latent Variable

Often z_i is represented as a **one-hot vector**:

For $K = 3$, if:

- x_i came from component 2, then:

$$z_i = [0,\ 1,\ 0]$$

So:

- $z_{ik} = 1$ if x_i belongs to component k
- $z_{ik} = 0$ otherwise

But since we **don't know** z_i, we can't directly use it in MLE.

💡 What Are Responsibilities?

🔸 Motivation: The Hidden Membership Problem

In GMM, your data $x_1, x_2, \ldots, x_n$ is assumed to be generated from **multiple Gaussian distributions** — but for each x_i, **you don't know** which Gaussian generated it.

That missing information is represented by a **latent variable** z_i, where:

- $z_i = k$ means point x_i was generated by the k-th Gaussian

Since z_i is unobserved, we instead ask:

> What is the **probability** that component k generated point x_i?

That probability is called the **responsibility** of component k for data point x_i.

🔹 Formal Definition:

The **responsibility** of component k for point x_i is:

$$\gamma_{ik} = P(z_i = k \mid x_i, \theta)$$

This is a **posterior probability** — "given that we observed x_i, how likely is it that it came from component k ?"

- Each component k takes **partial responsibility** for generating point x_i
- Unlike hard clustering (e.g., k-means), **every cluster contributes** to every point — **to different degrees**

The values γ_{ik} are:

- Close to 1 if component k closely explains x_i
- Close to 0 if component k is very unlikely to explain x_i

🔸 Responsibility Formula (Derived from Bayes' Rule):

$$\gamma_{ik} = \frac{w_k \cdot \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} w_j \cdot \mathcal{N}(x_i|\mu_j, \Sigma_j)}$$

🔍 **What does this mean?**

- **Numerator** = the *joint probability* that component k generated x_i:

$$P(x_i, z_i = k) = w_k \cdot \mathcal{N}(x_i|\mu_k, \Sigma_k)$$

- **Denominator** = total probability of x_i under **all** components:

$$P(x_i) = \sum_{j=1}^{K} w_j \cdot \mathcal{N}(x_i|\mu_j, \Sigma_j)$$

- So this is just:

$$\gamma_{ik} = \frac{P(x_i, z_i = k)}{P(x_i)} = P(z_i = k \mid x_i)$$

🔄 Expectation-Maximization (EM) Algorithm for GMM

📌 Objective

We are given:

- A dataset $\{x_1, x_2, ..., x_n\} \in \mathbb{R}^d$
- We assume data is generated from a **mixture of K Gaussian components**

Each point is:

$$x_i \sim \sum_{k=1}^{K} w_k \cdot \mathcal{N}(x_i|\mu_k, \Sigma_k)$$

We want to **estimate parameters**:

$$\theta = \{w_k, \mu_k, \Sigma_k\}_{k=1}^{K}$$

But we cannot apply MLE directly due to:

- **Unknown cluster labels z_i**
- Log-likelihood involving **log of a sum**

✅ Step-by-Step Algorithm

The EM algorithm alternates between two steps:

Step	Purpose
E-step	Estimate the hidden variables (soft cluster assignments)
M-step	Maximize the expected complete-data log-likelihood

It **iteratively improves** the parameter estimates θ until convergence.

🔹 Step 0: Initialization

Start with **random initial guesses** for parameters:

- Means μ_k
- Covariances Σ_k
- Weights w_k

You can:

- Choose random points as initial means
- Set all weights $w_k = 1/K$
- Use identity matrices for Σ_k

🔹 Step 1: E-Step (Expectation)

Compute **responsibilities** γ_{ik}, the probability that component k generated point x_i:

$$\gamma_{ik} = \frac{w_k \cdot \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} w_j \cdot \mathcal{N}(x_i|\mu_j, \Sigma_j)}$$

Now you have an $n \times K$ matrix of responsibilities.

🔹 Step 2: M-Step (Maximization)

Use γ_{ik} to update the parameters:

- **Effective number of points assigned to each cluster:**

$$N_k = \sum_{i=1}^{n} \gamma_{ik}$$

- **Update Means:**

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{n} \gamma_{ik} x_i$$

- **Update Covariance:**

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^{n} \gamma_{ik} (x_i - \mu_k)(x_i - \mu_k)^T$$

- **Update Weights:**

$$w_k = \frac{N_k}{n}$$

These are your **new estimates** for θ.

🔁 Step 3: Repeat

- Compute the **log-likelihood**:

$$\log L(\theta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} w_k \cdot \mathcal{N}(x_i|\mu_k, \Sigma_k) \right)$$

- Check if the **change in log-likelihood** is smaller than some threshold (e.g., 10^{-4})
- If not, **repeat E-step and M-step**

📘 Expectation-Maximization in K-Means Clustering (pyq makaut)

📘 Expectation-Maximization (EM) for K-Means Clustering

K-means clustering is a fundamental unsupervised learning algorithm used to group similar data points into KKK clusters. Although it is **not a probabilistic model**, it closely resembles the structure of the **Expectation-Maximization (EM) algorithm**, which is used to estimate parameters when data involves **hidden or latent variables**.

🔍 Intuition Behind the Connection

The EM algorithm is designed to deal with **incomplete data** by:

1. **E-step**: Estimating hidden variables given the current parameters
2. **M-step**: Updating parameters based on those estimates

K-means can be seen as a **special case of EM**:

- The **latent variable** is the cluster assignment for each point.
- The **parameters** are the centroids of clusters.
- The **likelihood maximization** becomes a **distance minimization** task.

🔄 K-Means as an EM Algorithm

K-means alternates between two steps:

🔹 1. E-Step: Assign Data Points to Nearest Cluster

In this step, K-means assigns each data point x_i to the cluster whose centroid μ_k is closest in Euclidean distance:

$$z_i = \arg\min_k \|x_i - \mu_k\|^2$$

- This is equivalent to estimating the **latent variable** z_i (cluster label).
- Unlike in EM for GMMs, where the assignment is **soft (probabilistic)**, here it is **hard (binary)**.

🔹 2. M-Step: Update Cluster Centroids

In the M-step, we update the centroid μ_k of each cluster by computing the **mean of all points assigned** to that cluster:

$$\mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$$

- This step is analogous to **maximizing the expected complete-data log-likelihood**.
- We are minimizing the **within-cluster sum of squares (WCSS)** — the total squared distance between each point and its assigned cluster center.

🧠 K-Means Objective Function

K-means aims to minimize:

$$J = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$

- This is equivalent to **minimizing the variance within each cluster**.
- It does **not model** the underlying data distribution — unlike GMMs, which assume Gaussian distributions.

K-means (like EM) repeats the E-step and M-step until:

- Assignments no longer change
- Or the centroids converge (change is below a small threshold)

🔔 **Summary**

- **K-means clustering follows the EM pattern**: alternates between estimating hidden labels (E-step) and updating model parameters (M-step).
- The difference lies in the fact that:
  - K-means uses **hard assignments** and **distance minimization**,
  - While EM in GMM uses **soft probabilities** and **likelihood maximization**.
- Hence, **K-means is a non-probabilistic, simplified instance of the EM algorithm.**

✅ THEORETICAL QUESTION LIST

🔶 Maximum Likelihood Estimation (MLE)

1. **Define Maximum Likelihood Estimation. Explain with an example.**
2. **Derive MLE estimators for mean and variance of a Gaussian distribution.**
3. **What are the advantages and disadvantages of MLE?**
4. **Why does MLE fail to estimate parameters in Gaussian Mixture Models directly?**
5. **Compare MLE and Bayesian estimation.**
6. **Explain how MLE is used in pattern classification.**
7. **Differentiate between likelihood and probability.**
8. **Write the likelihood and log-likelihood function for Gaussian data.**
9. **Discuss the assumptions required in MLE.**
10. **Give one numerical example illustrating the use of MLE in Gaussian parameter estimation.**

🔶 Gaussian Mixture Models (GMM)

11. **Define Gaussian Mixture Model. Explain each component of its formula.**
12. **Discuss the need for Gaussian Mixture Models in classification.**
13. **Explain the difference between a single Gaussian model and a Gaussian Mixture Model.**
14. **What are the limitations of using a single Gaussian? How does GMM solve them?**
15. **Why is GMM called a generative model?**
16. **What are the main applications of GMM in real-world pattern recognition tasks?**
17. **Write down the expression for GMM log-likelihood and explain its terms.**
18. **Discuss how GMM performs soft clustering.**
19. **List and explain the parameters of a GMM that need to be estimated.**
20. **Write a short note on modeling multimodal data with GMM.**

🔶 Latent Variables and Responsibilities

21. **What is a latent variable in the context of GMM?**
22. **Explain the concept of responsibility in GMM. Why is it needed?**
23. **Derive the formula for responsibility γik\gamma_{ik}γik using Bayes' Rule.**
24. **Discuss how responsibilities help in soft classification.**

25. **What does the E-step in the EM algorithm compute, and why?**

Expectation-Maximization (EM) Algorithm

26. **Discuss the EM algorithm used for GMM parameter estimation.**
27. **Explain the steps involved in the EM algorithm for GMM.**
28. **Derive the update equations for mean, covariance, and weight in the M-step.**
29. **Why is the EM algorithm needed in GMMs instead of direct MLE?**
30. **Illustrate one iteration of the EM algorithm on a small 1D dataset.**
31. **What is convergence criteria in the EM algorithm?**
32. **Explain the complete-data log-likelihood and how it differs from the marginal likelihood.**
33. **What are the limitations of the EM algorithm?**
34. **Compare EM algorithm and k-means clustering.**

EM Algorithm in K-Means (Simplified EM)

35. **Discuss expectation-maximization for the K-means clustering.**
36. **Explain how K-means follows the EM framework.**
37. **Compare EM for GMM vs EM-like steps in K-means.**
38. **What is the objective function of K-means? Is it a likelihood?**
39. **Why is K-means considered a special case of EM?**
40. **Write and explain the E-step and M-step of K-means clustering.**