Memory updated

Gaussian Mixture Models (GMM) – A Deep Dive

1. What is a Gaussian Mixture Model (GMM)?

Gaussian Mixture Models (GMM) is a **probabilistic clustering algorithm** that assumes the data is generated from **multiple Gaussian distributions** (clusters).

Unlike **K-Means**, which assigns each data point **to exactly one cluster**, GMM uses **soft clustering**, meaning:

- Each data point has a **probability** of belonging to each cluster.
- The clusters can have **different sizes, shapes, and orientations**.

How is GMM Different from K-Means?

Feature	K-Means	GMM
Cluster Shape	Spherical	Elliptical
Clustering Type	Hard (one point per cluster)	Soft (probabilistic)
Distance Metric	Euclidean	Probability Density
Good for?	Simple, well-separated clusters	Overlapping clusters

2. When to Use & Avoid GMM

✅ **Use GMM When:**

- Clusters are **not necessarily spherical**.
- Data has **overlapping clusters**.
- You need a **probabilistic membership** instead of hard assignments.

❌ **Avoid GMM When:**

- The dataset is **high-dimensional**, as GMM scales poorly.
- There are **too many clusters**, making convergence slow.
- The dataset has **many outliers**, as GMM is sensitive to noise.

3. How GMM Works – Expectation-Maximization (EM) Algorithm

GMM uses the **Expectation-Maximization (EM)** algorithm to iteratively estimate:

1. **Which Gaussian distribution each point likely belongs to (E-step)**
2. **Update Gaussian parameters (mean, variance, mixing coefficient) (M-step)**

Step-by-Step Overview

1. **Initialize** K Gaussian distributions (random means, variances, and mixing coefficients).
2. **Expectation Step (E-Step)**:
    - Compute the probability that each point belongs to each Gaussian.
3. **Maximization Step (M-Step)**:
    - Update Gaussian parameters using weighted means and variances.
4. **Repeat until convergence** (parameters stop changing significantly).

4. Step-by-Step Pseudo Code for GMM

plaintext Copy Edit

```
# Step 1: Initialize Parameters
1. Choose K (number of Gaussians).
2. Initialize means, variances, and mixing coefficients randomly.

# Step 2: Expectation-Maximization (EM) Loop
FOR each iteration:
    1. Expectation Step (E-Step):
        a. Compute responsibility: Probability that each data point belongs to each

    2. Maximization Step (M-Step):
        a. Update Gaussian parameters:
            - Compute new means (weighted sum).
            - Compute new variances.
            - Compute new mixing coefficients.

    3. Check for Convergence:
        - Stop if parameters change negligibly.

# Step 3: Output Final Cluster Probabilities
1. Assign each point to the Gaussian with the highest probability.
```

Mathematical Breakdown of Gaussian Mixture Models (GMM)

Below is a fully explicit mathematical breakdown of every step in **Expectation-Maximization (EM) for GMM**, including **initialization, E-step, M-step, and convergence**.

---

1. Initialization

We start by initializing three sets of parameters:

1. **Means (μ_k)**: The centers of each Gaussian cluster.
2. **Covariances (Σ_k)**: The spread (variance) of each Gaussian.
3. **Mixing Coefficients (π_k)**: The proportion of points in each cluster.

Mathematical Formulation

1. Randomly initialize K means:

$$\mu_k^{(0)} = x_{\text{random}} \quad \text{for } k = 1, \ldots, K$$

2. Initialize covariance matrices as identity matrices:

$$\Sigma_k^{(0)} = I_d \quad \text{for } k = 1, \ldots, K$$

3. Initialize mixing coefficients equally:

$$\pi_k^{(0)} = \frac{1}{K}, \quad \text{for } k = 1, \ldots, K$$

(Each cluster initially has equal probability.)

2. Expectation Step (E-Step)

For each data point x_i, compute the probability that it belongs to cluster k (**responsibility** r_{ik}):

$$r_{ik} = \frac{\pi_k p(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j p(x_i|\mu_j, \Sigma_j)}$$

where:

- $p(x_i|\mu_k, \Sigma_k)$ is the **multivariate Gaussian probability density function (PDF)**:

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

- π_k is the **mixing coefficient** (cluster weight).

Expanded E-Step Formula

$$r_{ik} = \frac{\pi_k \cdot \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x_i - \mu_k)^T \Sigma_k^{-1}(x_i - \mu_k)\right)}{\sum_{j=1}^{K} \pi_j \cdot \frac{1}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x_i - \mu_j)^T \Sigma_j^{-1}(x_i - \mu_j)\right)}$$

---

3. Maximization Step (M-Step)

Using the computed **responsibilities** r_{ik}, update the parameters μ_k, Σ_k, and π_k.

3.1 Updating the Means (μ_k)

The new mean for each Gaussian cluster is computed as a **weighted sum of all points** using the responsibilities:

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{N} r_{ik} x_i}{\sum_{i=1}^{N} r_{ik}}$$

where:

- N is the number of data points.
- The denominator represents the **total responsibility weight** assigned to cluster k.

3.2 Updating the Covariance Matrices (Σ_k)

The covariance for each cluster is computed as:

$$\Sigma_k^{(t+1)} = \frac{\sum_{i=1}^{N} r_{ik}(x_i - \mu_k^{(t+1)})(x_i - \mu_k^{(t+1)})^T}{\sum_{i=1}^{N} r_{ik}}$$

- This represents the **weighted covariance matrix** for the cluster.
- If the data is 1D, this simplifies to a **variance update**.

3.3 Updating the Mixing Coefficients (π_k)

The mixing coefficient for each cluster is:

$$\pi_k^{(t+1)} = \frac{1}{N} \sum_{i=1}^{N} r_{ik}$$

- This ensures the sum of all mixing coefficients equals 1:

$$\sum_{k=1}^{K} \pi_k = 1$$

---

4. Log-Likelihood & Convergence Criteria

After each iteration, compute the **log-likelihood** of the data given the current parameters:

$$\log L = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k p(x_i | \mu_k, \Sigma_k) \right)$$

- If **log-likelihood stops increasing significantly**, stop iterations (convergence achieved).
- The threshold for stopping is set using **tolerance (ϵ)**:

$$|\log L^{(t+1)} - \log L^{(t)}| < \epsilon$$

```python

class GMM:
    """
    Implementation of Gaussian Mixture Model using the Expectation-Maximization algorithm.
    """
    def __init__(self, k=3, max_iters=100, tol=1e-4):
        """
        Initialize GMM parameters.
        :param k: Number of Gaussian components.
        :param max_iters: Maximum number of iterations.
        :param tol: Convergence threshold.
        """
        self.k = k
        self.max_iters = max_iters
        self.tol = tol

    def initialize_parameters(self, X):
        """Initialize means, covariances, and mixing coefficients."""
        np.random.seed(42)
        self.n_samples, self.n_features = X.shape
        self.means = X[np.random.choice(self.n_samples, self.k, replace=False)]
        self.covariances = np.array([np.eye(self.n_features)] * self.k)
        self.mixing_coeffs = np.ones(self.k) / self.k

    def gaussian_pdf(self, X, mean, covariance):
        """Compute Gaussian probability density function."""
        d = X.shape[1]
        det = np.linalg.det(covariance)
        inv = np.linalg.inv(covariance)
        norm_const = 1 / ((2 * np.pi) ** (d / 2) * np.sqrt(det))
        diff = X - mean
        exponent = np.einsum('ij,jk,ik->i', diff, inv, diff)
        return norm_const * np.exp(-0.5 * exponent)

    def expectation_step(self, X):
        """E-Step: Compute responsibilities."""
        responsibilities = np.zeros((self.n_samples, self.k))
        for k in range(self.k):
            responsibilities[:, k] = self.mixing_coeffs[k] * self.gaussian_pdf(X, self.means[k], self.covariances[k])
        responsibilities /= responsibilities.sum(axis=1, keepdims=True)
        return responsibilities
```

```
43
44      def maximization_step(self, X, responsibilities):
45          """M-Step: Update parameters."""
46          Nk = responsibilities.sum(axis=0)
47          self.means = (responsibilities.T @ X) / Nk[:, None]
48          self.covariances = np.array([
49              ((responsibilities[:, k, None] * (X - self.means[k])).T @ (X - self.means[k])) / Nk[k]
50              for k in range(self.k)
51          ])
52          self.mixing_coeffs = Nk / self.n_samples
53
54      def fit(self, X):
55          """Fit GMM to data using EM algorithm."""
56          self.initialize_parameters(X)
57          for _ in range(self.max_iters):
58              responsibilities = self.expectation_step(X)
59              new_means = self.maximization_step(X, responsibilities)
60              if np.linalg.norm(self.means - new_means) < self.tol:
61                  break
62          self.means = new_means
```

```
    def predict(self, X):
        """Assign each data point to the Gaussian with the highest responsibility."""
        responsibilities, _ = self.expectation_step(X)
        return np.argmax(responsibilities, axis=1)

# Generate synthetic dataset for GMM
X, _ = make_blobs(n_samples=300, centers=3, cluster_std=1.0, random_state=42)

# Apply GMM manually
gmm = GMM(k=3)
gmm.fit(X)
labels = gmm.predict(X)

# Visualize GMM clustering
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', edgecolor='k', alpha=0.7)
plt.colorbar(label='Cluster Label')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('GMM Clustering')
plt.show()
```

7. Scikit-Learn Implementation

python Copy Edit

```python
from sklearn.mixture import GaussianMixture

# Train GMM
gmm = GaussianMixture(n_components=3, random_state=42)
gmm.fit(X)

# Predict cluster labels
labels = gmm.predict(X)
```