Gaussian Mixture Model and the EM Algorithm

Rohan Biswas
Roll Number: CSB21067

Introduction to the Clustering Techniques

Clustering is a fundamental task in data analysis, used to group similar data points together. This section discusses two common clustering methods relevant to us: hard clustering and smooth clustering.

Hard clustering, also known as crisp clustering, assigns each data point to exactly one cluster.
Smooth clustering, or fuzzy clustering, allows data points to belong to multiple clusters with varying degrees of membership.

Table 1: Comparison of Hard and Smooth Clustering

Feature	**Hard Clustering**	**Smooth Clustering**
Assignment	One cluster per point	Multiple clusters per point
Boundaries	Sharp and distinct	Overlapping
Algorithms	K-Means, Hierarchical	Fuzzy C-Means, GMM
Interpretability	Straightforward	More complex
Sensitivity to Noise	Sensitive to outliers	More robust
Applications	Image Segmentation, Document classification	Market segmentation, Bioinformatics, Speech Processing

K-means for example is a centroid-based method, so if the centers overlap? Then ??

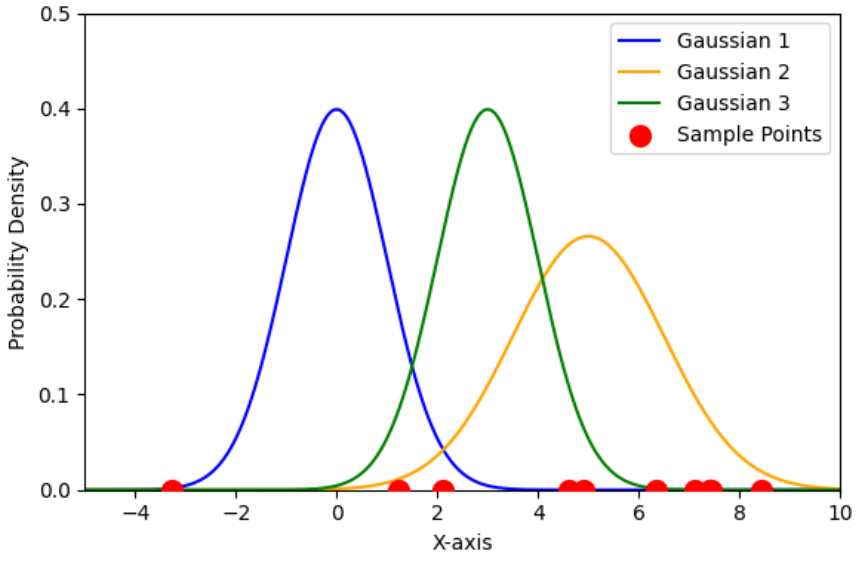

Figure 1: Overlapping clusters shown as overlapping gaussians

Brief Overview of the Gaussian Mixture Models

Gaussian Mixture Models are probabilistic techniques used for classification of data. It assumes data points to be generated like some sort of gaussian distribution which is quite common in nature.
The method/ technique goes like

- **Step 1:** Randomly initiate Gaussian(s) from the dataset.
- **Step 2:** Calculate the responsibility (probability of belongingness) for each point. This indicates the extent to which the data point relates to each of the Gaussian(s).
- **Step 3:** Consider the responsibility calculated in the previous step to determine the most appropriate Gaussian for that point.
- **Step 4:** After checking the probability and maintaining a particular threshold for the same, assign the Gaussian(s). Repeat this process until there is no change in the Gaussian(s), indicating that the solution has converged.

GMM is an iterative method like K-means with difference being use of parameters like **Mean** and **Variance**, instead of **Centroid**.
If we increase the dimensions, however, we get **Mean Vector** and **Covariance matrix** respectively.

For using Gaussian Mixture Models (GMM), we use an algorithm known as **Expectation-Maximization Algorithm**, popularly known as EM Algorithm.

GMM using EM Algorithm - A rigorous approach

Consider a gaussian distribution, $\mathcal{N}(x|\mu, \sigma^2)$ given as :

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$

Now this is Univariate. For Multivariate gaussian distribution we have :

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

where,
d is dimension
$\overrightarrow{x}$ is data vector
$\overrightarrow{\mu}$ is mean vector
ε is Covariance matrix

Now, we consider the Maximum Likelihood Estimation.
Suppose I have data points and I can represent them as Gaussian Distribution(s).
Now, we got to consider the log of the Gaussian distribution, take derivative and theb equatre to zero to get the values of the parameters $\overrightarrow{\mu}_{ML}$ and ε_{ML}.

$$\ln \mathcal{N}_m(\overrightarrow{x}|\overrightarrow{\mu}, \varepsilon) = -\frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\varepsilon| - \frac{1}{2}(\overrightarrow{x} - \overrightarrow{\mu})^T\varepsilon^{-1}(\overrightarrow{x} - \overrightarrow{\mu})$$

Now taking derivatives and equating to zero :

$$\frac{\partial \ln \mathcal{N}_m(\overrightarrow{x} \mid \overrightarrow{\mu}, \varepsilon)}{\partial \overrightarrow{\mu}} = 0 \qquad \frac{\partial \ln \mathcal{N}_m(\overrightarrow{x} \mid \overrightarrow{\mu}, \varepsilon)}{\partial \varepsilon} = 0$$

$$\implies \overrightarrow{\mu}_{ML} = \frac{1}{N}\sum_{n=1}^{N} \overrightarrow{x}_n \qquad \implies \varepsilon_{ML} = \frac{1}{N}\sum_{n=1}^{N}(\overrightarrow{x}_n - \overrightarrow{\mu}_{ML})^T(\overrightarrow{x}_n - \overrightarrow{\mu}_{ML})$$

where, $N \rightarrow$ Number of samples/ data points.

Now suppose the density(GMM) is basically given by a probability density function $p(x)$:

$$p(x) = \pi_1 f_1(x) + \pi_2 f_2(x) + ... + \pi_k f_k(x)$$

where k is the number of Gaussian(s), and $f_k(x)$ are the probability density function of the individual Gaussian(s) and π_i are mixing-coefficients or weights.
Thus, we can say that GMM is the weighted sum of the mixtures of the Gaussian(s) where the weights are determined by the mixing coefficients , or its the Linear superposition of Gaussian(s).
Now,

$$\sum_{i=1}^{K} \pi_i = 1$$

So,

$$\begin{aligned} p(x) &= \sum_{i=1}^{K} \pi_i f_i(x) \\ &= \pi_1 \mathcal{N}_m(x \mid \mu_1, \varepsilon_1) + \pi_2 \mathcal{N}_m(x \mid \mu_2, \varepsilon_2) + ... + \pi_k \mathcal{N}_m(x \mid \mu_k, \varepsilon_k) \\ &= \sum_{i=1}^{K} \pi_i \mathcal{N}_m(x \mid \mu_i, \varepsilon_i) \end{aligned}$$

Now we got to use EM Algorithm as we cannot use the Maximum Likelihood Estimation to get μ and ε in case of GMM, as there will be no convergence to a particular solution.

So, we use normalization ($0 \leq \pi_k \leq 1$) and Positivity $\left(\sum_{i=1}^{K} \pi_i = 1\right)$

We can think of the mixing co-efficients as prior probabilities of the components.

For a given x, we calculate the responsibilities or the posterior probabilities given as $\gamma_k(x)$.
Using Bayes' rule:

$$\gamma_K(x) = P(K|x) = \frac{P(K)P(x|K)}{P(x)}$$

Here, $P(K)$ is class prior.
$P(x|K)$ is class conditional probability.
$P(x)$ is probability prior or the unconditional prior.

$$P(k|x) = \frac{\pi_k \mathbf{N}(x|\mu_k, \varepsilon_k)}{\sum_{j=1}^{K} \pi_j \mathbf{N}(x|\mu_j, \varepsilon_j)}$$

where $\pi_k = \frac{\mathbf{N}_k}{\mathbf{N}}$,
and $\mathbf{N}_k$ is the total number of points assigned to cluster k.

EM Algorithm is thus an iterative optimization technique.

Estimation Step : For given parameter values, we compute the expected values of the latent variable (responsibility).
Maximization Step : Update parameters of the model based on the latent variable calculated using Maximum Likelihood method.

EM Algorithm for Gaussian Mixture Models

1. Initialize the means μ_j, covariance matrix ε_j and the mixing coefficients π_j.
   You may wanna evaluate the initial values by likelihood or randomly.

2. E-Step :
   Evaluate responsibility using current parameter values.

$$\gamma_j(x) = \frac{\pi_k \mathcal{N}(x|\mu_k, \varepsilon_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x|\mu_j, \varepsilon_j)}$$

3. M=Step :
   Re-estimate the parameters using the current responsibilities.

$$\mu_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n) x_n}{\sum_{n=1}^{N} \gamma_j(x_n)}$$

$$\varepsilon_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n)(x_n - \mu_j)(x_n - \mu_j)^T}{\sum_{n=1}^{N} \gamma_j(x_n)}$$

$$\pi_j = \frac{1}{N} \sum_{n=1}^{N} \gamma_j(x_n)$$

4. Evaluate Log Likelihood.

$$\ln P(x \mid \mu, \varepsilon, \pi) = \sum_{n=1}^{N} \ln \left(\sum_{i=1}^{K} \pi_i \mathcal{N}(x_n \mid \mu_i, \varepsilon_i) \right)$$

5. If there is no convergence, bgo back to step 2.

Python Code: GMM Clustering with Misclassifications

We now see how the code for GMM looks like. We here use the inbuilt python library Scikit-Learn to do the things which uses the EM Algorithm.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture

# Generate overlapping data (two Gaussian distributions)
np.random.seed(42)
n_samples = 300

# Cluster 1: Mean at (0, 0), spread
mean1 = [0, 0]
cov1 = [[1.0, 0.6], [0.6, 1.0]]
data1 = np.random.multivariate_normal(mean1, cov1, n_samples)

# Cluster 2: Mean at (2, 2), similar spread with overlap
mean2 = [2, 2]
cov2 = [[1.0, 0.6], [0.6, 1.0]]
data2 = np.random.multivariate_normal(mean2, cov2, n_samples)

```

```
# Combine the two clusters to create the dataset
X = np.vstack([data1, data2])

# K-means clustering (Hard clustering)
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans_labels = kmeans.fit_predict(X)

# GMM clustering (Soft clustering)
gmm = GaussianMixture(n_components=2, covariance_type='full', random_state=42)
gmm.fit(X)
gmm_labels = gmm.predict(X)
gmm_probs = gmm.predict_proba(X)

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 500),
                     np.linspace(y_min, y_max, 500))

Z_kmeans = kmeans.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
Z_gmm = gmm.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

# Identify misclassified points by K-means
misclassified = (kmeans_labels != gmm_labels)

# Plots
fig, axes = plt.subplots(1, 2, figsize=(15, 6))
axes[0].contourf(xx, yy, Z_kmeans, cmap='viridis', alpha=0.3)
axes[0].scatter(X[:, 0], X[:, 1], c=kmeans_labels, cmap='viridis',
                edgecolor='k', s=40)
axes[0].scatter(X[misclassified, 0], X[misclassified, 1],
                color='red', marker='x', s=80, label='Misclassified')
axes[0].scatter(kmeans.cluster_centers_[:, 0],
                kmeans.cluster_centers_[:, 1], c='red',
                marker='o', s=100, label='Centroids')
axes[0].set_title('K-means Clustering with Misclassifications')
axes[0].legend(loc='best')

axes[1].contourf(xx, yy, Z_gmm, cmap='viridis', alpha=0.3)
scatter = axes[1].scatter(X[:, 0], X[:, 1], c=gmm_probs[:, 1],
                          cmap='coolwarm', s=40, edgecolor='k')
axes[1].set_title('GMM Clustering with Soft Probabilities')
fig.colorbar(scatter, ax=axes[1], label='Probability of Cluster 2')

plt.tight_layout()
plt.show()
```

The output obtained is :

Figure 2: KMeans Vs Gaussian Mixture Model Code Example

Applications in Speech Processing

Speech Recognition

- **Feature Extraction:** Use GMMs to model the distribution of feature vectors extracted from speech signals (e.g., Mel-frequency cepstral coefficients, MFCCs).
- **Hidden Markov Models (HMMs):** GMMs are often used as the observation probability density functions in HMMs for speech recognition. Each state of the HMM can be modeled by a GMM, capturing the variability in speech across different speakers or phonemes.

Speaker Recognition

Model the voice characteristics of different speakers using GMMs. Each speaker can be represented by a GMM trained on their speech data, allowing for differentiation between speakers based on their unique vocal characteristics.

Voice Activity Detection (VAD)

Use GMMs to model the distribution of features in speech and non-speech segments. The EM algorithm can help optimize the GMM parameters for distinguishing between voice and background noise.

Speech Synthesis

In statistical parametric speech synthesis, GMMs can model the relationship between linguistic features and acoustic features, facilitating natural-sounding speech generation.

Gaussian Mixture Models for Speech Recognition

Gaussian Mixture Models (GMMs) are widely used in speech recognition tasks, particularly in modeling the distribution of feature vectors extracted from speech signals. The key steps involved in using GMMs for speech recognition are outlined below.

GMM for Speech Recognition

1. **Feature Extraction:** Extract features from the speech signal, such as Mel-frequency cepstral coefficients (MFCCs):

$$\text{MFCC}_t = \sum_{n=1}^{N} C_n \log \left(\sum_{m=1}^{M} W_m |X_m(t)|^2 \right) \tag{1}$$

   where $X_m(t)$ is the m-th frequency bin of the Fourier transform at time t, C_n are the coefficients, and W_m are the filter bank weights.

2. **Model Training:** Train the GMM using the Expectation-Maximization (EM) algorithm. The algorithm iteratively updates the parameters of the GMM:

   - **E-step:** Estimate the responsibilities $\gamma_j(x)$.
   - **M-step:** Update the parameters μ_j, ε_j, and π_j

3. **Decoding:** Use the trained GMM to compute the likelihood of a sequence of feature vectors:

$$P(\mathbf{X}|\text{Model}) = \prod_{t=1}^{T} p(\mathbf{x}_t) \tag{2}$$

   where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ is the sequence of feature vectors.

4. **Decision Making:** Apply a decision rule (e.g., Maximum Likelihood, Viterbi decoding) to determine the most likely word or phoneme sequence:

$$W^* = \arg\max_{W} P(W|\mathbf{X}) \tag{3}$$

   where W represents the hypothesis word sequence.

References

1. Bishop, C. M. (2006). **Pattern Recognition and Machine Learning**. Springer.
2. Murphy, K. P. (2012). **Machine Learning: A Probabilistic Perspective**. MIT Press.
3. Zhang, Y., Alder, M., & L'ogneri, R. (2003). **Using Gaussian Mixture Modeling in Speech Recognition**. Center for Intelligent Information Processing System, The University of Western Australia, Nedlands, Australia