Gaussian Mixture Models

General Principles

To discover group structures or clusters in data, we can use a **Gaussian Mixture Model (GMM)**. This is a parametric clustering method. A GMM assumes that the data is generated from a mixture of a **pre-specified number (`K`)** of different Gaussian distributions. The model's goal is to figure out:

1. **The properties of each of the `K` clusters**: For each of the `K` clusters, it estimates its center (mean μ) and its shape/spread (covariance Σ).
2. **The mixture weights**: It estimates the proportion of the data that belongs to each cluster.
3. **The assignment of each data point**: It determines the probability of each data point belonging to each of the `K` clusters.

Considerations

Caution

- A GMM is a Bayesian model that considers uncertainty in all its parameters, *except for the number of clusters, `K`*, which must be fixed in advance.
- The key parameters and their priors are:
  - **Number of Clusters `K`**: This is a **fixed hyperparameter** that you must choose before running the model. Choosing the right `K` often involves running the model multiple times and using model comparison criteria (like cross-validation, AIC, or BIC).
  - **Cluster Weights `w`**: These are the probabilities of drawing a data point from any given cluster. Since there are a fixed number `K` of them and they must sum to 1, they are typically given a `Dirichlet` prior. A symmetric `Dirichlet` prior (e.g., `Dirichlet(1, 1, ..., 1)`) represents an initial belief that all clusters are equally likely.

- **Cluster Parameters** (μ, Σ): Each of the `K` clusters has a mean μ and a covariance matrix Σ. We place priors on these to define our beliefs about their plausible values.

- Like the DPMM, the model is often implemented in its marginalized form . Instead of explicitly assigning each data point to a cluster, we integrate out this choice. This creates a smoother probability surface for the inference algorithm to explore, leading to much more efficient computation.

- To increase accuracy we run a k-means algorithm to initialize the cluster mean priors.

Example

Below is an example of a GMM implemented in BI. The goal is to cluster a synthetic dataset into **a pre-specified `K=4` groups**.

Python

```
from BI import bi
from sklearn.datasets import make_blobs

# Generate synthetic data
data, true_labels = make_blobs(
    n_samples=500, centers=8, cluster_std=0.8,
    center_box=(-10,10), random_state=101
)


#  The model
def gmm(data, K, initial_means): # Here K is the *exact* number of clusters
    D = data.shape[1]  # Number of features
    alpha_prior = 0.5 * jnp.ones(K)
    w = dist.dirichlet(concentration=alpha_prior, name='weights')

    with dist.plate("components", K): # Use fixed K
        mu = dist.multivariatenormal(loc=initial_means, covariance_matrix=0.1*jnp.eye(D), nam
        sigma = dist.halfcauchy(1, shape=(D,), event=1, name='sigma')
        Lcorr = dist.lkjcholesky(dimension=D, concentration=1.0, name='Lcorr')
```

```
        scale_tril = sigma[..., None] * Lcorr

    dist.mixturesamefamily(
        mixing_distribution=dist.categorical(probs=w, create_obj=True),
        component_distribution=dist.multivariatenormal(loc=mu, scale_tril=scale_tril, create
        name="obs",
        obs=data
    )

m.data_on_model = {"data": data,"K": 4 }
m.fit(gmm) # Optimize model parameters through MCMC sampling
m.plot(X=data,sampler=m.sampler) # Prebuild plot function for GMM
```

R

Mathematical Details

This section describes the generative process for a GMM. For each data point x_i, the model first selects one of the `K` clusters according to the weights w, and then draws the point from that cluster's Gaussian distribution.

$$z_i \sim \text{Categorical}(w) \text{for } i = 1, \dots, N x_i \mid z_i = k \sim \text{MultivariateNormal}(\mu_k, \Sigma_k) \text{for } i = 1, \dots, N$$

$$w \sim \text{Dirichlet}(\alpha_0) \quad \text{(Mixture weights vector for K clusters)}$$

$$\mu_k \sim \text{MultivariateNormal}(\mu_0, \Sigma_0) \text{for } k = 1, \dots, K \sigma_k \sim \text{HalfCauchy}(1) \text{for } k = 1, \dots, K L_{\text{corr},k} \sim \text{LKJCholesky}(D, 1$$

Parameter Definitions: * **Observed Data:** * x_i: The i-th observed D-dimensional data point.

- **Latent Variables (Inferred):**
  - z_i: The integer cluster assignment for the i-th data point.
  - w: The K-dimensional vector of mixture weights.
  - μ_k: The D-dimensional mean vector of the k-th cluster.
  - Σ_k: The DxD covariance matrix of the k-th cluster (composed of σ_k and $L_{\text{corr},k}$).

- **Hyperparameters (Fixed):**
  - K: The total number of clusters.
  - α_0: The concentration parameter vector for the Dirichlet prior on weights (e.g., `[1, 1, ..., 1]`).
  - μ_0: The prior mean for the cluster centers.
  - Σ_0: The prior covariance for the cluster centers.

Notes

> **Note**
>
> The primary challenge of the GMM compared to the DPMM is the need to **manually specify the number of clusters** `K`. If the chosen `K` is too small, the model may merge distinct clusters. If `K` is too large, it may split natural clusters into meaningless sub-groups. Therefore, applying a GMM often involves an outer loop of model selection where one fits the model for a range of `K` values and uses a scoring metric to select the best one.

Reference(s)

C. M. Bishop (2006). *Pattern Recognition and Machine Learning*. Springer. (Chapter 9)