

# 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  $\mu$ ) and its shape/spread (covariance  $\Sigma$ ).
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** ( $\mu, \Sigma$ ): Each of the  $K$  clusters has a mean  $\mu$  and a covariance matrix  $\Sigma$ . 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.multivariate_normal(loc=initial_means, covariance_matrix=0.1*jnp.eye(D), name='mu')
        sigma = dist.halfcauchy(1, shape=(D,), event=1, name='sigma')
        Lcorr = dist.lkjcholesky(dimension=D, concentration=1.0, name='Lcorr')
```