

# Dirichlet Process Mixture Models

## General Principles

To discover group structures or clusters in data without pre-specifying the number of groups, we can use a **Dirichlet Process Mixture Model (DPMM)**. This is a non-parametric clustering method. Essentially, the model assumes the data is generated from a collection of different Gaussian distributions, and it simultaneously tries to figure out:

1. **How many clusters ( $K$ ) exist:** Unlike algorithms like K-Means, the DPMM infers the most probable number of clusters directly from the data.
2. **The properties of each cluster:** For each inferred cluster, it estimates its center (mean  $\mu$ ) and its shape/spread (covariance  $\sigma$ ).
3. **The assignment of each data point:** It determines the probability of each data point belonging to each cluster.

## Considerations

### 🔥 Caution

- A DPMM is a Bayesian model that considers uncertainty in all its parameters. The core idea is to use the Dirichlet Process prior that allows for a potentially infinite number of clusters. In practice, we use a finite approximation called the Stick-Breaking Process .
- The key parameters and their priors are:
  - **Concentration  $\alpha$ :** This single parameter controls the tendency to create new clusters. A low favors fewer, larger clusters, while a high allows for many smaller clusters. We typically place a **Gamma** prior on  $\alpha$  to learn its value from the data.
  - **Cluster Weights  $w$ :** Generated via the Stick-Breaking process from  $\alpha$ . These are the probabilities of drawing a data point from any given cluster.

- **Cluster Parameters** ( $\mu, \sigma$ ): Each potential cluster has a mean  $\mu$  and a covariance matrix  $\sigma$ . If the data have multiple dimensions, we use a multivariate normal distribution (see chapter, 14). However, if the data is one-dimensional, we use a univariate normal distribution.
- 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.

## Example

Below is an example of a DPMM implemented in BI. The goal is to cluster a synthetic dataset into its underlying groups. The code first generates data with 4 distinct centers and then applies the DPMM to recover these clusters.

## 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 dpmm(data, T=10):
    N, D = data.shape # Number of features
    data_mean = jnp.mean(data, axis=0)
    data_std = jnp.std(data, axis=0)*2

    # 1) stick-breaking weights
    alpha = dist.gamma(1.0, 10.0, name='alpha')

    with m.plate("beta_plate", T - 1):
        beta = m.dist.Beta(1, alpha)

    w = numpyro.deterministic("w", mix_weights(beta))
```

```

# 2) component parameters
with m.plate("components", T):
    mu = m.dist.MultivariateNormal(loc=data_mean, covariance_matrix=data_std*jnp.eye(D), name='mu')# shape (T, D)
    sigma = m.dist.lognormal(0.0, 1.0, shape=(D,), event=1, name='sigma')# shape (T, D)
    Lcorr = m.dist.lkjcholesky(dimension=D, concentration=1.0, name='Lcorr')# shape (T, D)

    scale_tril = sigma[..., None] * Lcorr # shape (T, D, D)

# 3) Latent cluster assignments for each data point
with m.plate("data", N):
    # Sample the assignment for each data point
    z = m.dist.Categorical(w) # shape (N,)

    # Sample the data point from the assigned component
    m.dist.MultivariateNormal(loc=mu[z], scale_tril=scale_tril[z],
                               obs=data
                           )

m.data_on_model = dict(data=data)
m.fit(dpmm) # Optimize model parameters through MCMC sampling
m.plot(X=data, sampler=m.sampler) # Prebuild plot function for GMM

```

## R

### Mathematical Details

This level describes how any single data point,  $x_i$ , is generated. The process involves two steps: first, assigning the data point to a cluster, and second, drawing it from that cluster's specific distribution. We use a truncation level  $T$  as a finite approximation for the infinite number of possible clusters in a true Dirichlet Process.

$$x_i \mid z_i = k \sim \text{MultivariateNormal}(\mu_k, \Sigma_{\text{obs}}) z_i \sim \text{Categorical}(w) w = \text{StickBreaking}(\beta_1, \dots, \beta_{T-1}) \beta_k \sim \text{Beta}(1, \alpha) \alpha$$

**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 vector of mixture weights, where  $w_k$  is the probability of belonging to cluster  $k$ .
  - $\beta_k$ : The set of Beta-distributed random variables for the stick-breaking process.
  - $\alpha$ : The concentration parameter, controlling the effective number of clusters.
  - $\mu_k$ : The D-dimensional mean vector of the  $k$ -th cluster.
- **Hyperparameters (Fixed):**
  - $\mu_0$ : The prior mean for the cluster centers (e.g., `mean(data)`).
  - $\Sigma_0$ : The prior covariance for the cluster centers (e.g., `10 * I_D`).

## Notes

 Note

The primary advantage of the DPMM over methods like K-Means or a GMM is the **automatic inference of the number of clusters**. Instead of running the model multiple times with different values of K and comparing them, the DPMM explores different numbers of clusters as part of its fitting process. The posterior distribution of the weights  $w$  reveals which components are “active” (have significant weight) and thus gives a probabilistic estimate of the number of clusters supported by the data.

## Reference(s)

Gershman and Blei (2012)

Gershman, Samuel J, and David M Blei. 2012. “A Tutorial on Bayesian Nonparametric Models.” *Journal of Mathematical Psychology* 56 (1): 1–12.