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 μ) and its shape/spread (covariance σ).
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** α: 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 α to learn its value from the data.

- - **Cluster Weights `w`**: Generated via the Stick-Breaking process from α. These are the probabilities of drawing a data point from any given cluster.

  - **Cluster Parameters** (μ, σ): Each potential cluster has a mean μ and a covariance matrix σ. If the data have multiple dimensions, we use a multivariate normal distribution (see chapter, 14). Howver, 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, jnp
from BI.Models.DPMM import mix_weights
from sklearn.datasets import make_blobs
import numpyro

m = bi()

# 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 = m.dist.gamma(1.0, 10.0,name='alpha')
    with m.dist.plate("beta_plate", T - 1):
        beta = m.dist.beta(1, alpha)

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

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

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

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

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

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
```

```
jax.local_device_count 16
```

```
/home/sosa/work/BI/BI/Main/main.py:236: FutureWarning:

Some algorithms will automatically enumerate the discrete latent site z of your model. In th

  0%|          | 0/1000 [00:00<?, ?it/s]warmup:   0%|          | 1/1000 [00:03<57:21,  3.45s
```

R

```
```

Mathematical Details

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 K as a finite approximation for the infinite number of possible clusters in a true Dirichlet Process.

$$\begin{pmatrix} Y_{i,1} \\ \vdots \\ Y_{i,D} \end{pmatrix} \sim \text{MVN}\left(\begin{pmatrix} \mu_{z_i,1} \\ \vdots \\ \mu_{z_i,D} \end{pmatrix}, \Sigma_{z_i} \right)$$

$$\begin{pmatrix} \mu_{k,1} \\ \vdots \\ \mu_{k,D} \end{pmatrix} \sim \text{MVN}\left(\begin{pmatrix} Ak, 1 \\ \vdots \\ Ak, D \end{pmatrix}, B \right)$$

$$\Sigma_k = \sigma_k \Omega_k \sigma_k$$

$$\sigma_k \sim \text{HalfCauchy}(1)$$

$$\Omega_k \sim \text{LKJ}(2)$$

$$z_i \sim \text{Categorical}(\pi)$$

$$\pi = \text{StickBreaking}(\beta_1, ..., \beta_K)$$

$$\beta_k \sim \text{Beta}(1, \alpha)$$

$$\alpha \sim \text{Gamma}(1, 10)$$

Where :

- $\begin{pmatrix} Y_{[i,1]} \\ \vdots \\ Y_{[i,D]} \end{pmatrix}$ is the *i*-th observation of a D-dimensional data array.
- $\begin{pmatrix} \mu_{[k,1]} \\ \vdots \\ \mu_{[k,D]} \end{pmatrix}$ is the *k*-th parameter vector of dimension D.
- $\begin{pmatrix} A_{[k,1]} \\ \vdots \\ A_{[k,D]} \end{pmatrix}$ is a prior for the *k*-th mean vector as derived by a *KMEANS* clustering algorithm.
- B is the prior covariance of the cluster means, and is setup as a diagonal matrix with 0.1 along the diagonal.

- Σ_k is the DxD covariance matrix of the k-th cluster (it is composed from σ_k and Ω_k).
- σ_k is a diagonal matrix of standard deviations for the k-th cluster.
- Ω_k is a correlation matrix for the k-th cluster.
- z_i is a latent variable that maps observation i to cluster k.
- π is a vector of K cluster weights.
- β_k: The set of K Beta-distributed random variables used in the stick-breaking process to construct the mixture weights.
- α: The concentration parameter, controlling the effective number of clusters.

Notes

> ℹ 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.