Project - Big Data

Jiayi Sun

June 2025

1 Dirichlet Process Gaussian Mixture Models (DPGMM)

1.1 Preprocessing

I use the original dataset comprising 10,000 samples represented as a 600-dimensional sparse matrix.

Due to substantial scale differences among variables, the covariance matrix was initially ill-conditioned and failed under Cholesky decomposition. To address this, I applied standardization (zero mean and unit variance) across all features.

Furthermore, given the high dimensionality, I performed Principal Component Analysis (PCA) to reduce memory usage and computational cost. We retained the top principal components that captured 95% of the total variance, resulting in a reduced dimensionality of 325.

1.2 Method

We adopt a Dirichlet Process Gaussian Mixture Model (DPGMM) to perform non-parametric clustering over the preprocessed data. Each data point

$$x_i \in \mathbb{R}^D$$

is assumed to be generated from a Gaussian distribution with unknown mean and covariance. The generative process is specified as:

$$G \sim DP(\alpha, G_0), \theta_k = (\mu_k, \Sigma_k) \sim G_0, x_i \sim N(\mu_{z_i}, \Sigma_{z_i})$$

where G_0 is a Normal-Inverse-Wishart (NIW) prior distribution, chosen for its conjugacy to the Gaussian likelihood.

1.3 Clustering Strategy

1.3.1 Prior Parameters

We set the hyperparameters of the NIW base measure as follows:

Concentration parameter: $\alpha = 5.0$, controlling the expected number of active clusters.

Degrees of freedom: $\nu_0 = D + 2$, which ensures that the expectation of the covariance matrix exists.

Scale matrix: $\Lambda_0 = 0.1 \times \mathbf{I}_D$, a scaled identity matrix for weakly informative covariance prior.

Mean prior: $\mu_0 = \mathbf{0} \in \mathbb{R}^D$.

Mean precision: $\kappa_0 = 0.01$, indicating high uncertainty over the mean.

1.3.2 Sufficient Statistics Update

We implement collapsed Gibbs sampling, marginalizing out the mixture parameters. To avoid recomputing covariance matrices upon each sample reassignment, we maintain three sufficient statistics for each cluster:

the number of assigned points n,

the sum of vectors $\sum x_i$,

and the sum of outer products $\sum x_i x_i^T$.

This allows efficient updates of cluster parameters.

1.3.3 Inference Procedure

At each iteration, for each data point x_i.

1. **Remove** it from its current cluster and update the corresponding sufficient statistics.

2. **Compute posterior probabilities** of assigning x_i to existing clusters and a potential new cluster. The marginal likelihood for existing clusters is computed using the current estimates of μ_k and Σ_k, while the new cluster assignment probability is derived from the marginalized form of the NIW-Gaussian model. This yields a multivariate Student-t distribution:

$$x_i \sim StudentT(\nu = \nu_0 - D + 1, \mu_0, \frac{\kappa_0 + 1}{\kappa_0(\nu_0 - D + 1)}\Lambda_0).$$

3. **Normalize** the log-probabilities via a softmax function and resample the cluster assignment.

4. **Update parameters** only for clusters whose membership changed. Due to numerical instabilities, we enforce positive semi-definiteness of sampled covariance matrices by spectral projection if needed.

This procedure is repeated for a fixed number of iterations. In each iteration, new clusters can be dynamically created or removed based on the posterior draw, allowing the model to adaptively infer the appropriate number of clusters.

Due to computational constraints in the current implementation, a single iteration of the Gibbs sampler requires approximately two hours to complete. As a result, we report the clustering outcome based on only one iteration. While this limits convergence and posterior mixing, it still serves to illustrate the model's initialization behavior and early clustering tendencies.

1.4 Evaluation

To complement our Gibbs sampling-based DPGMM implementation, we also evaluated clustering results using the BayesianGaussianMixture model (BGMM) from scikit-learn. This model approximates the Dirichlet Process Gaussian Mixture via truncated variational inference rather than sampling-based inference.

Specifically, BGMM uses a finite but large number of components K_{max} to approximate the infinite mixture, and performs mean-field variational inference under a stick-breaking prior. Each component has its own variational parameters for the mean and covariance, and posterior responsibilities are optimized by maximizing the evidence lower bound (ELBO) using coordinate ascent. In contrast, our DPGMM implementation uses collapsed Gibbs sampling, where cluster assignments are sampled sequentially, and cluster parameters are integrated out analytically under a Normal-Inverse-Wishart prior.

The key differences can be summarized as follows:

Aspect	BGMM	Gibbs-DPGMM
Inference	Variational (mean-field, coordinate ascent)	MCMC (collapsed Gibbs sampling)
Cluster number	Truncated (fixed K_{max})	Adaptive (drawn from DP posterior)
Posterior distribution of θ	Point estimate (variational mode)	Full sampling-based posterior
Computation	Fast and scalable	Slower, exact posterior exploration

Table 1: Differences between Bayesian Gaussian Mixture Model and Gibbs-Dirichlet Process Gaussian Mixture Model

To compare the two models under a controlled setting, we fixed the number of clusters in BGMM to 50 by specifying the n_components parameter and using a Dirichlet Process prior with moderate concentration (i.e., weight_concentration_prior_type='dirichlet_process'). This ensured that the model converged to a compact representation with 50 active clusters and minimal fragmentation.

```
BIRCH labels written to outputs/DPMM_labels(sklearn).csv.

Cluster 0: 18 points
Cluster 1: 298 points
Cluster 2: 193 points
Cluster 3: 140 points
Cluster 4: 199 points
Cluster 5: 234 points
Cluster 6: 246 points
Cluster 7: 339 points
Cluster 8: 256 points
Cluster 9: 143 points
Cluster 10: 213 points
Cluster 11: 258 points
Cluster 12: 273 points
Cluster 13: 184 points
Cluster 14: 116 points
Cluster 15: 281 points
Cluster 16: 178 points
Cluster 17: 125 points
Cluster 18: 139 points
Cluster 19: 252 points
Cluster 20: 105 points
Cluster 21: 82 points
Cluster 22: 371 points
```

Figure 1: Cluster Summary of BGMM

In contrast, our Gibbs sampling-based DPGMM is fully nonparametric and does not explicitly constrain the number of clusters. Moreover, since we report results after only one iteration due to runtime limitations, the sampler has not yet converged and exhibits substantial over-fragmentation. Specifically, it produced 4,141 clusters, many of which contain only 1–2 points. The largest cluster (cluster 0) contains 5,859 points, while the vast majority of the remaining clusters are extremely small and likely reflect early-stage noise in the assignment process.

```
1-th iteration has beginned.

1-th iteration has completed.

DPGMM labels written to outputs/dpgmm_labels.csv.

Cluster 0: 5859 points
Cluster 1: 1 points
Cluster 2: 1 points
Cluster 3: 1 points
Cluster 4: 1 points
Cluster 5: 1 points
Cluster 6: 1 points
Cluster 7: 1 points
Cluster 8: 1 points
Cluster 9: 1 points
Cluster 10: 1 points
Cluster 11: 1 points
Cluster 12: 1 points
Cluster 13: 1 points
Cluster 14: 1 points
Cluster 15: 1 points
Cluster 16: 1 points
Cluster 17: 1 points
Cluster 18: 1 points
```

Figure 2: Cluster Summary of DPGMM

This contrast highlights the importance of sufficient iterations in sampling-based DPGMMs. While variational inference in BGMM quickly converges to a compact solution with fixed capacity, the sampling-based DPGMM requires more time to gradually merge small clusters and approach a stable posterior over the number of components.

2 Balanced Iterative Reducing and Clustering using Hierarchies

2.1 Preprocessing

I use the original dataset comprising 10,000 samples represented as a 600-dimensional sparse matrix.

I applied standardization (zero mean and unit variance) across all features.

2.2 Method

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) is a scalable hierarchical clustering algorithm designed for large datasets. It incrementally constructs a compact summary of the data—called the CF-tree (Clustering Feature tree)—that captures the distributional structure using a hierarchy of clustering features. Each non-leaf node in the CF-tree contains a set of clustering feature entries, which summarize subclusters through three sufficient statistics: the number of points N, the linear sum $LS = \sum x_i$, and the squared sum $SS = \sum x_i^2$.

The algorithm proceeds in two main phases:

Online CF-tree construction: data points are inserted sequentially into the tree. Each point is either absorbed into an existing subcluster or triggers the creation of a new leaf entry, depending on a predefined threshold that limits the radius of each cluster. Nodes are split as needed to maintain balance.

Global clustering: after the CF-tree is built, the centroids of leaf entries are clustered using a standard algorithm such as k-means or agglomerative clustering to obtain the final cluster assignments.

BIRCH is particularly efficient for large-scale, memory-constrained settings, as it avoids retaining all raw data in memory and achieves a single-pass clustering structure. However, it is sensitive to the threshold hyperparameter and may over-fragment the data if not tuned properly. In this study, I employ BIRCH to produce a compressed hierarchical summary, followed by global clustering to recover a specified number of clusters.

2.3 Clustering Strategy

I implemented a custom version of the BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm to accommodate our high-dimensional dataset and to gain full control over thresholding and clustering behavior. The implementation follows the original two-phase structure: (1) construction of a compact CF-tree (Clustering Feature tree), and (2) global clustering using k-means on the leaf centroids.

2.3.1 Parameters

The main parameters of the model are:

Threshold (threshold): the maximum radius allowed for merging a new data point into an existing subcluster. It controls the granularity of the CF-tree and must be chosen carefully to avoid over-fragmentation or over-merging.

Branching factor (B): the maximum number of entries in non-leaf nodes.

Leaf capacity (L): the maximum number of entries per leaf node before triggering a split.

Number of final clusters (nclusters): the desired number of output clusters, enforced in the second phase via global k-means clustering.

Seed (seed): used for deterministic behavior in both tree traversal and clustering.

2.3.2 Implementation Strategy

Clustering Feature (CF) Entry: Each leaf entry maintains three sufficient statistics:

N (number of points),
$LS = \sum x_i$ (linear sum),
$SS = \sum x_i^2$ (squared sum).

These are used to compute the centroid and radius of the cluster in closed form.

2.3.3 Insertion

For each new data point, the algorithm searches down the CF-tree recursively, always selecting the nearest child based on Euclidean distance to centroids. Upon reaching a leaf node, it attempts to merge the data point into the closest CF-entry if the resulting radius remains below the threshold. Otherwise, a new CF-entry is created. If the number of entries exceeds L, the node is split and the split propagates upward recursively.

2.3.4 Node Splitting

When a node exceeds its capacity, it is split by selecting the two most distant entries as seeds (max-pairwise-distance heuristic), and redistributing all other entries according to proximity. Parent entries are updated accordingly, and new root nodes are created dynamically if necessary.

2.4 Leaf Linking

Leaf nodes maintain a next_leaf pointer to enable linear traversal across the entire leaf layer. This facilitates efficient collection of all CF-entries for global clustering.

2.5 Global Clustering

After the tree is built, we extract all CF-entry centroids and cluster them using K-means with weights proportional to the number of points in each entry. This step enables consolidation of overly fine-grained subclusters into a fixed number of interpretable groups.

2.6 Evaluation

For comparative purposes, I additionally employed the Birch implementation provided by scikit-learn. Compared to our custom BIRCH algorithm, the scikit-learn version abstracts away several low-level operations, such as node splitting logic, explicit leaf traversal, and the use of sufficient statistics (LS, SS, N) for CFEntry updates. Furthermore, while our implementation allows for manual tuning and inspection of tree construction at each insertion step, the scikit-learn

implementation offers a more black-box interface optimized for ease-of-use and computational speed. Notably, scikit-learn's version does not expose internal tree structure or allow flexible post-processing strategies such as weighted global K-means over CFEntry centroids.

In both BIRCH models—the custom implementation and scikit-learn's version—we successfully applied K-means clustering in the global refinement phase to consolidate subclusters into approximately 50 final clusters. This post-processing step significantly reduced over-fragmentation caused by small CF entries and provided a more interpretable partitioning of the dataset. Below, we present selected clustering statistics and visualizations to compare the effectiveness of both implementations under the same post-clustering target.

```
The data has been inserted.

BIRCH labels written to outputs/birch_labels.csv.

Cluster 0.0: 197 points
Cluster 1.0: 47 points
Cluster 2.0: 40 points
Cluster 3.0: 455 points
Cluster 4.0: 1 points
Cluster 5.0: 399 points
Cluster 6.0: 129 points
Cluster 7.0: 244 points
Cluster 8.0: 86 points
Cluster 9.0: 2279 points
Cluster 10.0: 105 points
Cluster 11.0: 26 points
Cluster 12.0: 8 points
Cluster 13.0: 218 points
Cluster 14.0: 37 points
Cluster 15.0: 71 points
Cluster 16.0: 45 points
Cluster 17.0: 33 points
Cluster 18.0: 4 points
Cluster 19.0: 50 points
```

Figure 3: Cluster Summary for BIRCH

```
BIRCH labels written to outputs/birch_labels(sklearn).csv.

Cluster 0: 250 points
Cluster 1: 88 points
Cluster 2: 115 points
Cluster 3: 63 points
Cluster 4: 135 points
Cluster 5: 555 points
Cluster 6: 3158 points
Cluster 7: 251 points
Cluster 8: 80 points
Cluster 9: 87 points
Cluster 10: 160 points
Cluster 11: 226 points
Cluster 12: 123 points
Cluster 13: 244 points
Cluster 14: 201 points
Cluster 15: 56 points
Cluster 16: 58 points
Cluster 17: 120 points
```

Figure 4: Cluster Summary for BIRCH(SKLEARN)

3 Bibliography

Literature References:

Ferguson, Thomas S. 1973. A Bayesian Analysis of Some Nonparametric Problems. *The Annals of Statistics, 1*(2), 209-230.

Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: an efficient data clustering method for very large databases. *SIGMOD Rec. 25*(2), 103–114.