University of Milano–Bicocca
School of Economics and Statistics

Master's Degree Course in
Statistical and Economic Sciences

Bayesian nonparametric clustering with variational inference and Fusing of Localized Densities

Supervisor: Dr. Tommaso Rigon
Co-supervisor: Prof. Bernardo Nipoti

Degree thesis by:
Alessandro Colello
Student ID No. 851334

Academic Year 2024/2025

Contents

Introduction

Clustering is a fundamental task in unsupervised learning that aims to discover meaningful groupings within data. Unlike distance-based clustering (e.g., k-means, hierarchical clustering), which relies on distance metrics to group similar data points, model-based clustering assumes that data are generated from a mixture of underlying probabilistic models, allowing uncertainty quantification and interpretable inference. In recent years, Bayesian nonparametric models, especially the Dirichlet process mixture model (DPMM), have gained significant attention for their flexibility in inferring the number of clusters directly from the data (Antoniak, 1974).

However, Bayesian inference in DPMMs is computationally intensive. The widely used Markov chain Monte Carlo (MCMC) methods provide asymptotically exact inference, but suffer from high computational cost, slow convergence, and the label switching problem. These issues hinder the scalability of the model to large datasets and its usability in practice. Variational inference (VI) offers a faster, deterministic alternative by approximating the posterior with a tractable family of distributions (Blei et al., 2017). Although VI typically underestimates the posterior variance, it provides a valuable trade-off between accuracy and computational efficiency in many practical scenarios.

The Gaussian mixture model is the most popular choice when considering continuous data. However, the data frequently does not accommodate for the use of Gaussian kernels, leading to instances of kernel misspecification. This issue results in an overestimation of the number of clusters by the model. To address this problem, Dombowsky & Dunson (in press) introduced the Fusing of Localized Densities (FOLD) procedure, which constructs a point estimate and credible ball for clusterings by minimizing a kernel-based loss function. FOLD does not require a posterior similarity matrix and aligns closely with decision-theoretic principles.

This thesis contributes to the literature by exploring the combination of VI and FOLD in the context of Bayesian nonparametric clustering. Specifically, we implement a variational approach to estimate a Dirichlet process Gaussian

mixture model with a conjugate Gaussian-Wishart prior, and apply FOLD to the resulting posterior approximation to extract point estimates and quantify uncertainty. Our goal is to evaluate the effectiveness of this approach both in terms of clustering accuracy and computational cost, and compare it rigorously against MCMC-based inference followed by FOLD. The proposed approach is illustrated in three simulation studies and on benchmark datasets.

The structure of the thesis is as follows. We begin with a theoretical overview of the DPMM, providing its general specification and detailing its properties. Then, we describe inference strategies for DPMMS, contrasting MCMC and VI in terms of convergence behavior, computational complexity, and their impact on clustering estimates.

The FOLD methodology is then introduced in detail, including its loss function formulation, computational implementation, and interpretation of the resulting credible ball. We extend the FOLD procedure to the variational setting by sampling from the variational posterior and deriving clusterings accordingly.

We conduct simulation studies under both well-specified and misspecified models, analyzing the performance of VB and MCMC estimates before and after applying FOLD in terms of number of inferred clusters, accuracy, and runtime. Besides the Hellinger distance, which is utilized in the implementation of the FOLD method, the Wasserstein metric, which has recently gained popularity in the context of optimal transport, is taken into consideration. Finally, we apply the method to real datasets commonly used in clustering benchmarks.

Our findings from both simulated and real datasets show that variational inference, when combined with FOLD, yields competitive clustering results while reducing computational time compared to traditional MCMC methods. These promising outcomes open up new avenues for further methodological development and research.

Chapter 1

Bayesian nonparametric clustering

Bayesian nonparametric models offer a flexible approach to clustering by allowing the complexity of the model to be determined by the data rather than being fixed in advance. These models are particularly appealing for unsupervised learning tasks as they avoid the need to predefine the number of mixture components. Instead, they allow to incorporate prior information to center the expected number of clusters around an estimate. Moreover, Bayesian nonparametric models inherit the advantages of Bayesian modeling, including sequential updating of beliefs, model interpretability, and principled quantification of uncertainty.

This chapter begins by introducing a foundational tool in Bayesian nonparametrics: the Dirichlet process (DP). We start from the definition of this stochastic process used as a prior over discrete probability measures, explore its properties, and conclude with a constructive representation. Then, the Dirichlet process mixture model, which arises when the DP is employed as a prior for the weights in a mixture model, is introduced. We provide its basic definition and an augmented version that is particularly useful for inference. The two strategies to perform inference in DPMMs are discussed: Markov chain Monte Carlo (MCMC) algorithms, which provide asymptotically exact inference at the cost of high computational demands, and variational inference (VI), which offer faster and deterministic approximate solutions by optimizing a lower bound on the marginal likelihood. For MCMC, we discuss the theoretical foundations and provide a widely used algorithm as an example. For VI, we outline the general optimization framework and describe a practical algorithm to solve it. Finally, we illustrate how clustering is typically performed using the outputs of these inference strategies.

1.1 Dirichlet process

The Dirichlet process (DP) is a fundamental object in Bayesian nonparametrics, being a distribution over probability distributions. It was introduced by Ferguson (1973) to solve the problem of defining a nonparametric prior distribution whose support is large enough and which, given a sample, leads to an analytically tractable posterior distribution.

The DP was first defined as a generalization of the well-known Dirichlet distribution by considering a measurable space $(\mathcal{X}, \mathcal{A})$, where $\mathcal{A}$ is a σ-field of subsets of $\mathcal{X}$.

Definition 1.1. Let G_1 be a non-null finite measure (non-negative and finitely additive) on $\mathcal{X}$. We say that a random probability measure $\tilde{P}$ is a Dirichlet process $DP(G_1)$ on $\mathcal{X}$ if, for every $k = 1, 2, \ldots$ and measurable partition $B_1, \ldots, B_k$ of $\mathcal{X}$, the distribution of $\left[\tilde{P}(B_1), \ldots, \tilde{P}(B_k)\right]$ is $\mathtt{Dirichlet}(G_1(B_1), \ldots, G_1(B_k))$.

It is often convenient to decompose the base measure G_1 as $G_1 = \alpha G_0$, where α is a positive real number referred to as concentration parameter, and G_0 is a probability measure over $\mathcal{X}$, called base distribution. Under this reparameterization, we write that

$$\tilde{P} \sim DP(\alpha, G_0).$$

Considering Definition 1.1, it can be deducted that, for $A \in \mathcal{X}$, $\tilde{P}(A)$ follows a beta distribution with parameters $\alpha G_0(A)$ and $\alpha[1 - G_0(A)]$, that is

$$\tilde{P}(A) \sim \mathtt{Beta}\big(\alpha G_0(A), \alpha[1 - G_0(A)]\big).$$

Thus, by recalling the properties of the beta distribution, we can easily retrieve the following results:

$$\mathbb{E}\left[\tilde{P}(A)\right] = G_0(A) \quad \text{and} \quad \mathrm{Var}\left[\tilde{P}(A)\right] = \frac{G_0(A)\,[1 - G_0(A)]}{\alpha + 1}.$$

This makes explicit the interpretation of G_0 as the prior mean of the random measure $\tilde{P}$, and of α as controlling the concentration of $\tilde{P}$ around G_0. Therefore, a small α yields sample distributions that are more discrete and exhibit higher differences from G_0, while a large α yields draws that more closely resemble G_0.

A key property of the DP its is conjugacy to sampling from the unknown distribution. Let us consider $\tilde{P} \sim DP(\alpha, G_0)$ and $X_i \mid \tilde{P} \overset{iid}{\sim} \tilde{P}$ for $i = 1, \ldots, n$. It can

be shown that $\tilde{P}$ conditional on the observed values $X_1, \ldots, X_n$ is still a Dirichlet process with updated parameters, that is

$$\tilde{P} \mid X_1, \ldots, X_n \sim \text{DP}\left(\alpha + n, \frac{\alpha}{\alpha + n} G_0 + \frac{1}{\alpha + n} \sum_{i=1}^{n} \delta_{X_i}\right).$$

The latter property allows to determine the posterior predictive distribution, which will be useful in the following sections. Specifically, the posterior predictive distribution of a new observation X_{n+1} conditional on the observed values $X_1 \ldots, X_n$ is given by

$$X_{n+1} \mid X_1, \ldots, X_n \sim \frac{\alpha}{\alpha + n} G_0 + \sum_{i=1}^{n} \frac{1}{\alpha + n} \delta_{X_i}$$

assuming conditional independence, i.e. exchangeability. If we then assume that the observed data presents repeated values, we can write

$$X_{n+1} \mid \tilde{X}_1, \ldots, \tilde{X}_k \sim \frac{\alpha}{\alpha + n} G_0 + \sum_{j=1}^{k} \frac{n_j}{\alpha + n} \delta_{\tilde{X}_j}$$

where $\tilde{X}_1, \ldots, \tilde{X}_k$ denote the unique values, i.e. clusters, within the data with multiplicities $n_1, \ldots, n_k$. This says that X_{n+1} will assume either a new value drawn from the base distribution G_0 with probability $\alpha/(\alpha + n)$ or the value $\tilde{X}_j$ with probability $n_j/(\alpha + n)$ for $j = 1, \ldots, k$. This explains the so-called "rich get richer" property of the DP according to which clusters with more observations are more likely to attract new observations. The number of distinct values K_n among the first n draws from a Dirichlet process is itself a random variable. In particular, the following exact expression holds for its expectation (Ghosal & van der Vaart, 2017):

$$\mathbb{E}[K_n] = \sum_{i=1}^{n} \frac{\alpha}{\alpha + i - 1}.$$

This result shows that the expected number of clusters depends on the DP solely through the concentration parameter, and increases logarithmically with the sample size n. Moreover, assuming that the base distribution is non-atomic, the following bounds for the expected number of distinct values can be derived (Ghosal & van der Vaart, 2017):

$$\max\left\{1, \alpha \log\left(1 + \frac{n}{\alpha}\right)\right\} \leqslant \mathbb{E}[K_n] \leqslant 1 + \alpha \log\left(1 + \frac{n}{\alpha}\right).$$

Besides Ferguson's definition, there are multiple equivalent ways to define and construct the Dirichlet process that provide more intuitive or computationally convenient interpretations. One of the most widely used is the stick-breaking representation (Sethuraman, 1994).

Theorem 1.1. *If* $Z_1, Z_2, \ldots \overset{\text{iid}}{\sim} G_0$ *and* $W_1, W_2, \ldots \overset{\text{iid}}{\sim} \text{Beta}(1, \alpha)$ *are independent sequences of random variables and* $\pi_j = W_j \prod_{l=1}^{j-1}(1 - W_l)$*, then* $\sum_{j=1}^{\infty} \pi_j \delta_{Z_j} \sim \text{DP}(\alpha, G_0)$.

This construction offers an explicit form for a random draw from a DP. The name stick-breaking comes from interpreting the weights π_j as the proportions of a unit-length stick: W_1 is the first break, $W_2(1 - W_1)$ is the second, and so on. The atoms X_j are drawn independently from the base distribution G_0 and are associated with these weights.

The stick-breaking representation again emphasizes the discrete nature of samples from a DP. This property of generating discrete probability distributions almost surely makes the DP a valuable tool for clustering tasks, as it can be used as a prior on the mixing distribution of a mixture model. We explore this in the next section.

1.2 Dirichlet process mixture model

Since the realizations of a Dirichlet process are almost surely discrete measures, it is generally inappropriate to use them directly for modeling continuous data. Instead, a more suitable approach is to introduce a kernel function, that is any function $\mathcal{K}\colon \mathcal{X} \times \Theta \to \mathbb{R}^+$ such that for each $\theta \in \Theta$, the function $\mathcal{K}(\cdot; \theta)$ is a probability density function. In this framework, the DP is placed as a prior distribution over the latent parameters θ, leading to the Dirichlet process mixture model (DPMM).

Definition 1.2. A Dirichlet process mixture model is a random density function on $\mathcal{X}$ defined as

$$\tilde{f}(x) = \int_{\Theta} \mathcal{K}(x; \theta)\, d\tilde{P}(\theta)$$

where $\tilde{P} \sim \text{DP}(\alpha, G_0)$ and $\mathcal{K}(x; \theta)$ is a kernel function.

This model allows for a more convenient hierarchical representation:

$$\begin{aligned} X_i \mid \theta_i &\overset{\text{ind}}{\sim} \mathcal{K}(\theta_i) \\ \theta_i \mid \tilde{P} &\overset{\text{iid}}{\sim} \tilde{P} \\ \tilde{P} &\sim \text{DP}(\alpha, G_0) \end{aligned} \tag{1.1}$$

for $i = 1, \ldots, n$. Each observation X_i is thus assumed to be independently drawn from a distribution parametrized by a latent variable θ_i. Since $\tilde{P}$ is almost surely discrete, the θ_i can only take a finite number of values. As a result, there is a positive probability for observations i and j to have the same parameter value. Hence, we can interpret X_i and X_j as belonging to the same cluster if $\theta_i = \theta_j$. To make the cluster structure of the DPMM explicit, we introduce a sequence of latent variables $Z_i \in \mathbb{N}$, where $Z_i = j$ indicates that the i-th observation is associated with the j-th cluster. The assignment variables are drawn from a categorical distribution governed by a sequence of random mixture weights $\{\pi_j\}_{j=1}^{\infty}$, which results from the stick-breaking representation. The cluster-specific parameters $\tilde{\theta}_j$ are drawn i.i.d. from the base measure G_0, and observations X_i are then generated conditionally on the assigned cluster. This augmentation facilitates inference by revealing the partitioning structure induced by the Dirichlet process prior.

Let $\mathbf{Z} = [Z_1, \ldots, Z_n]$ be the vector of latent cluster assignments, $\tilde{\boldsymbol{\theta}} = (\tilde{\theta}_j)_{j=1}^{\infty}$ be the infinite sequence of component parameters, and $\boldsymbol{\pi} = (\pi_j)_{j=1}^{\infty}$ be the infinite sequence of mixture weights.

The augmented model can be thus expressed as follows:

$$\begin{aligned} X_i \mid Z_i, \tilde{\boldsymbol{\theta}} &\overset{ind}{\sim} \mathcal{K}\left(\tilde{\theta}_{Z_i}\right) \\ Z_i \mid \boldsymbol{\pi} &\overset{iid}{\sim} \mathrm{Categorical}(\boldsymbol{\pi}) \\ \tilde{\boldsymbol{\theta}} &\overset{iid}{\sim} G_0 \\ \boldsymbol{\pi} &\sim \mathrm{GEM}(\alpha) \end{aligned} \tag{1.2}$$

for $i = 1, \ldots, n$, where GEM denotes the Griffiths-Engen-McCloskey distribution, arising from the stick-breaking representation.

The likelihood of the data $\mathbf{X}$ given the assignments $\mathbf{Z}$ and the component parameters $\tilde{\boldsymbol{\theta}}$ is

$$p(\mathbf{X} \mid \mathbf{Z}, \tilde{\boldsymbol{\theta}}) = \prod_{i=1}^{n} \mathcal{K}(X_i \mid \tilde{\theta}_{Z_i}).$$

The prior distribution of the component parameters $\boldsymbol{\pi}$ is

$$p(\tilde{\boldsymbol{\theta}}) = \prod_{j=1}^{\infty} g_0(\tilde{\theta}_j)$$

where g_0 denotes the density function associated with the base distribution G_0. The distribution of the assignments $\mathbf{Z}$ given the mixing weights $\boldsymbol{\pi}$ is

$$p(\mathbf{Z} \mid \boldsymbol{\pi}) = \prod_{i=1}^{n} \pi_{Z_i}.$$

The distribution of the mixture weights is defined by the stick-breaking construction, and thus implicitly defined by the joint distribution of the independent V_j variables:

$$p(\boldsymbol{\pi}) = \prod_{j=1}^{\infty} \mathrm{Beta}(V_j \mid 1, \alpha) = \prod_{j=1}^{\infty} \alpha(1 - V_j)^{\alpha-1}.$$

The posterior distribution $p(\mathbf{Z}, \tilde{\theta}, \boldsymbol{\pi} \mid \mathbf{X})$ is proportional to the product of the likelihood of the data and the prior distributions of the parameters and latent variables:

$$p(\mathbf{Z}, \tilde{\theta}, \boldsymbol{\pi} \mid \mathbf{X}) \propto p(\mathbf{X} \mid \mathbf{Z}, \tilde{\theta}) p(\tilde{\theta}) p(\mathbf{Z} \mid \boldsymbol{\pi}) p(\boldsymbol{\pi}).$$

Inferring the posterior distribution directly from its unnormalized form is computationally challenging due to the infinite-dimensional nature of the parameter space. Therefore, various inference algorithms have been developed to explore this posterior distribution. The following section will expand on these computational methods, distinguishing between sampling-based methods and optimization-based methods.

1.3 Inference in Dirichlet process mixture models

This section discusses the two strategies for inference in Dirichlet process mixture models: Markov chain Monte Carlo (MCMC) methods and variational inference (VI). This introduction is based on Blei et al. (2017).

An MCMC method sets up a Markov chain, whose stationary distribution is the intractable posterior distribution of a Bayesian model. After running the chain for a sufficient number of iterations, the generated samples can be regarded as draws from the posterior distribution. These samples can thus be used to summarize the posterior, such as by estimating posterior expectations, or to study more complex posterior structures, such as clustering configurations. Thanks to its theoretical formulation, MCMC provides asymptotically exact results. This means that, given enough computing time, it can characterize the posterior with high precision. However, this computational intensity can also be a drawback, especially for large datasets or very complex models, as convergence to the stationary distribution can be slow and difficult to determine.

On the other hand, VI turns the inference problem into an optimization problem. First, a family of approximating distributions is chosen. This family must be flexible enough to capture the posterior features, but simple enough to allow for efficient optimization. Then, we determine the element of this family that best approximate the posterior distribution by minimizing the Kullback-Leibler divergence. Especially when dealing with exponential families, the iterative optimization algorithm becomes very simple and it tends to be faster than MCMC methods. Although, variational inference is known to underestimate posterior variance due to its objective function, and it lacks the theoretic guarantees of MCMC methods.

MCMC and VI are therefore two different approaches to solving the same problem. The former samples from a Markov chain, and the latter optimizes a function. In general, MCMC is suited to smaller datasets or situations where a heavier computational cost is accepted in exchange for more precise samples. Instead, VI is preferred for large datasets or when rapid exploration of many models is required.

In what follows, we provide basic definitions and theoretic results that justify MCMC methods. We then discuss the Gibbs sampling approach to draw samples from the stationary distribution in its generality, and illustrate Algorithm 8 of Neal (2000), a widely used Gibbs sampler to approximate the posterior distribution of DPMMs. Successively, after discussing alternatives to constraint the infinite-dimensional parameter space of DPMMs, we provide a general introduction to VI, discussing the optimization problem and how it is solved.

1.3.1 Markov chain Monte Carlo methods

Markov chain Monte Carlo (MCMC) methods are fundamental tools for performing inference in DPMMs. These methods enable sampling from complex posterior distributions by constructing Markov chains whose stationary distribution matches the target posterior. While these methods provide theoretically exact results, they can be computationally intensive. This section is based on Robert & Casella (2004), which offers a comprehensive treatment of MCMC techniques. We begin by discussing the fundamentals of Markov chains to justify their use in Bayesian inference. Then, we focus on Gibbs sampling, one of the most important MCMC algorithms. Finally, we present a widely used Gibbs sampler tailored for DPMMs, which will be essential in the subsequent chapters.

A Markov chain is a stochastic process, that is a sequence of random variables, in which the transition probability that determines the transition to a state depends only on the immediately preceding state.

Definition 1.3 (Markov chain)**.** A sequence $(X^{(n)})_{n=0}^{\infty}$ of random variables is a Markov chain if

$$P(X^{(n+1)} \in \cdot \mid X^{(0)}, \ldots, X^{(n)}) = P(X^{(n+1)} \in \cdot \mid X^{(n)}).$$

In the continuous case, the transition probability is determined by a transition kernel. This kernel is identified by a conditional density, $K(X^{(n+1)} \in \cdot \mid X^{(n)})$. We say that a Markov chain is homogeneous if the distribution of $(X_{n_1}, \ldots, X_{n_k})$ given X_{n_0} is the same as $(X_{n_1-n_0}, \ldots, X_{n_k-n_0})$ given X_0 for every k and every $n_0, \ldots, n_k$ such that $n_0 \leqslant n_1 \leqslant \cdots \leqslant n_k$. It follows that, given its initial state, the chain is completely determined by its transition kernel.

A Markov chain exhibits high stability when the marginal distribution of X_n is independent of n. This is a requirement for the existence of an invariant or stationary distribution π such that $X_n \sim \pi$ for each n. Therefore, it follows the definition.

Definition 1.4 (Invariant distribution)**.** A probability density function π is invariant for the transition kernel K (and for the associated chain) if

$$\pi(x^\star) = \int K(x^\star \mid x)\pi(x)\, dx.$$

The stability of a Markov chain is described by three properties: irreducibility, aperiodicity, and Harris recurrence. Intuitively, a Markov chain is said to be irreducible if it does not get stuck in a local region of the sample space, aperiodic if it does not have deterministic cycles, and Harris recurrent if it visits any region of the sample space frequently enough.

These properties alone do not guarantee the existence of an invariant distribution. To admit such a distribution, a Markov chain needs to satisfy the detailed balance condition.

Definition 1.5. A Markov chain with transition kernel K satisfies the detailed balance condition if there exists a function f satisfying

$$K(x^\star \mid x)f(x) = K(x \mid x^\star)f(x^\star).$$

The following theorem (Robert & Casella, 2004) formalizes what was previously stated.

Theorem 1.2. *Suppose that a Markov chain with transition function* K *satisfies the detailed balance condition with* π *a probability density function. Then:*

(i) The density π *is the invariant density of the chain.*

(ii) The chain is reversible.

A Markov chain is called Harris positive if it is Harris recurrent and it admits an invariant distribution.

The following theorem (Robert & Casella, 2004) states that, under certain conditions, a chain converges in total variation to its invariant distribution, regardless of the distribution of the initial state.

Theorem 1.3. *If a Markov chain* $(X^{(n)})_{n=0}^{\infty}$ *is Harris positive and aperiodic, then*

$$\lim_{n\to\infty} \|\pi_n(\cdot) - \pi(\cdot)\|_{TV} = 0$$

where π_n *denotes the marginal probability density function of* $X^{(n)}$ *and* $\|\cdot\|_{TV}$ *denotes the total variation norm.*

This result is crucial for the theoretical foundation of MCMC methods, as it gives strong convergence guarantees for chains that are Harris positive and aperiodic. Moreover, it justifies using MCMC methods to sample from the posterior, because it ensures that the distribution of the samples will get arbitrarily close to the true posterior as $n \to \infty$.

Finally, the Ergodic Theorem (Robert & Casella, 2004) is what makes MCMC useful in practice, as it ensures that sample averages from the Markov chain converge to the correct posterior expectation.

Theorem 1.4 (Ergodic Theorem)**.** *If a Markov chain* $(X^{(n)})_{n=0}^{\infty}$ *is Harris positive with stationary distribution* π*, then, for every integrable function* g*, it holds that*

$$\frac{1}{N}\sum_{n=1}^{N} g(X^{(n)}) \xrightarrow{a.s.} \int g(x)\pi(x)\,dx$$

as $N \to \infty$.

In practice, MCMC methods are used to generate R samples from a Markov chain whose stationary distribution is the posterior distribution. To allow the chain to approach this distribution, the first B iterations, known as the burn-in period, are typically discarded. The value of B is chosen empirically using

diagnostic tools. The remaining samples are then used to approximate posterior expectations or other functionals of interest.

Several algorithms exist to construct such Markov chains, with Gibbs sampling being one of the most widely used, particularly in mixture models.

Let Z denote the set of all the parameters and latent variables of a Bayesian model, and let $X = [X_1, \dots, X_n]$ be the observed data. Suppose that Z is decomposable into $D > 1$ elements such that we can simulate from the corresponding conditional densities (called full conditional distributions). That is, for each $d = 1, \dots, D$, we can sample

$$Z_d \mid Z_{-d}, X \sim p(Z_d \mid Z_{-d}, X).$$

where $Z_{-d} = [Z_1, \dots, Z_{d-1}, Z_{d+1}, \dots, Z_D]$ denotes all components of Z except the d-th.

Gibbs sampling proceeds by repeatedly sampling each Z_d from its full conditional. This algorithm defines a Markov chain that has the posterior distribution $p(Z \mid X)$ as its stationary distribution, thereby yielding an MCMC method that targets the posterior.

In practice, Gibbs samplers often generate highly autocorrelated trajectories. To mitigate this, one may apply thinning, that is keeping only every k-th draw from the chain, to reduce dependence between consecutive samples.

This basic Gibbs sampling framework can be extended to more complex models, such as DPPMs, where the number of parameters theoretically infinite. One of the most famous examples is Algorithm 8 of Neal (2000), also known as marginal sampler, which exploits the cluster assignment variables and a finite number of proposed new clusters to improve mixing. Specifically, the algorithm alternate between:

- Sampling parameters of the clusters.
- Proposing new clusters via latent variables.
- Sampling cluster assignments for each data point given the current parameters.

The generic iteration of Neal's Algorithm 8 for the generic augmented DPMM in Equation (1.2) is presented in Algorithm 1.

The key idea of this algorithm is to avoid directly sampling from the infinite set of possible mixture components by instead introducing a finite number m of auxiliary clusters at each iteration. These proposed clusters provide

Algorithm 1: Generic iteration of the Algorithm 8 of Neal (2000)

Let the state of the Markov chain consist of $\boldsymbol{Z} = [Z_1, \ldots, Z_n]$ and $\tilde{\boldsymbol{\theta}} = \{\tilde{\theta}_z : z \in \{Z_1, \ldots, Z_n\}\}$. Let m denote the fixed number of auxiliary clusters temporarily introduced during the update of cluster assignments.

for $i = 1, \ldots, n$ **do**

  Let k^- be the number of distinct Z_j for $j \neq i$, and let $h = k^- + m$.

  Label these Z_j with values in $\{1, \ldots, k^-\}$.

  if $Z_i = Z_j$ *for some* $j \neq i$ **then**

    Draw values independently from G_0 for $\tilde{\theta}_j$ with $j = k^- + 1, \ldots, h$.

  else

    Let Z_i have label $k^- + 1$, and draw values independently from G_0 for $\tilde{\theta}_j$ with $j = k^- + 2, \ldots, h$.

  Draw a new value for Z_i from $\{1, \ldots, h\}$ using the following probabilities:

$$P(Z_i = z \mid Z_{-i}, X_i, \tilde{\theta}_1, \ldots, \tilde{\theta}_h) \propto \begin{cases} \frac{n_{-i,z}}{n-1+\alpha} \mathcal{K}(X_i \mid \tilde{\theta}_z) & \text{for } 1 \leqslant z \leqslant k^- \\ \frac{\alpha/m}{n-1+\alpha} \mathcal{K}(X_i \mid \tilde{\theta}_z) & \text{for } k^- + 1 \leqslant z \leqslant h \end{cases}$$

  where $n_{-i,z}$ is the number of Z_j for $j \neq i$ that are equal to z.

  Change the state to contain only those $\tilde{\theta}_z$ that are now associated with one or more observations.

  for $z \in \{Z_1, \ldots, Z_n\}$ **do**

    Draw a new value from $\tilde{\theta}_z \mid \{X_i : Z_i = z\}$, or perform some other update to $\tilde{\theta}_z$, that leaves this distribution invariant.

additional flexibility when updating the cluster assignments, improving posterior exploration.

The choice of m plays a critical role in the performance of the algorithm. A larger m offers more candidate clusters and thus can improve mixing, but also increases computational cost per iteration.

Overall, Neal's Algorithm 8 remains a foundational Gibbs sampling scheme for DPMMs, as it balances tractability with flexibility, making it well-suited for Bayesian nonparametric inference.

1.3.2 Variational inference

In a Bayesian setting where the posterior distribution is intractable, variational inference (VI) or variational Bayes (VB) provides an approximation with a distribution that belongs to a pre-specified family of tractable distributions.

Variational inference for DPMMs requires careful considerations due to the infinite-dimensional nature of the Dirichlet process, particularly its stick-breaking representation. One strategy, reviewed by Blei & Jordan (2006), circumvents this by truncating only the variational family rather than the generative model itself. In their formulation, the variational distribution is defined over a finite number T of components, setting all mixture weights beyond the T-th component to zero within the approximation.

Another approach consists in simplifying the model itself by introducing a finite-dimensional approximation to the Dirichlet process directly in the generative prior. This practical strategy, discussed in Ishwaran & Zarepour (2002), replaces the infinite stick-breaking process with a symmetric Dirichlet distribution over a fixed number H of mixture components:

$$p(\boldsymbol{\pi}) = \text{Dirichlet}\left(\boldsymbol{\pi} \,\middle|\, \frac{\alpha}{H}\mathbf{1}_H\right). \tag{1.3}$$

In this way, we obtain the finite-dimensional Dirichlet prior:

$$\tilde{P}_H = \sum_{h=1}^{H} \pi_h \delta_{\tilde{\theta}_h}$$

where $\tilde{\theta}_j$ are drawn i.i.d. from G_0, independently of $\boldsymbol{\pi}$.

Conditional on $\tilde{\boldsymbol{\theta}} = [\tilde{\theta}_1, \ldots, \tilde{\theta}_H]$, $\tilde{P}_H$ is a Dirichlet process:

$$\tilde{P}_H \mid \tilde{\boldsymbol{\theta}} \sim \text{DP}(\alpha, \xi_H)$$

where $\xi_H = \frac{1}{H}\sum_{h=1}^{H}\delta_{\tilde{\theta}_h}$ is the empirical measure of $\tilde{\theta}_1, \ldots, \tilde{\theta}_H$. Since $\xi_H \approx G_0$, we expect $\tilde{P}_H \approx DP(\alpha, G_0)$. Therefore, intuitively, $\tilde{P}_H$ is a good approximation of the $DP(\alpha, G_0)$.

It is demonstrated that functionals of the finite-dimensional Dirichlet prior can be used to approximate functionals of the Dirichlet process. Ishwaran & Zarepour (2002) give the following theorem.

Theorem 1.5. *For each real-valued measurable function* g, *integrable with respect to* G_0, $\tilde{P}_H(g) \xrightarrow{d} \tilde{P}(g)$, *where* $\tilde{P}(g) \sim DP(\alpha, G_0)$.

Furthermore, Ishwaran & Zarepour (2002) compare the finite-dimensional Dirichlet prior and the Dirichlet process by studying their clustering behavior under sampling. In particular, they give the following theorem.

Theorem 1.6. *Let* K_H *and* K_∞ *be the number of distinct values in* $Y = [Y_1, \ldots, Y_n]$ *when sampled under* $\tilde{P}_H$ *and* $\tilde{P} \sim DP(\alpha, G_0)$, *respectively. If* G_0 *is non-atomic, then*

$$\frac{H!}{H^k(H-k)!} \leqslant \frac{P(K_H = k)}{P(K_\infty = k)} \leqslant n^{\alpha k/H}, \quad \textit{for } k = 1, \ldots, \min\{n, H\}.$$

Note that the two distributions agree in limit as $H \to \infty$ because both the left and right sides tend to 1 for each k.

These properties provide strong evidence for the finite-dimensional approximation being very accurate in practical scenarios. The choice of H is critical as it must be sufficiently large to retain the flexibility of the Dirichlet process while keeping the optimization problem manageable. In practice, H is selected based on the sample size n and concentration parameter α, aiming to capture the data's clustering structure within the truncated model.

We will employ a finite-dimensional Dirichlet prior within the DPMM, resulting in a finite mixture model with a Dirichlet prior on the weights. This formulation facilitates a general discussion of variational Bayes.

Let us denote with $p(Z \mid X)$ the posterior distribution arising from a Bayesian model, where Z is the set of all parameters and latent variables, and $X = [X_1, \ldots, X_n]$ is the set of observations. Let $q(Z)$ be a density function, belonging to a family of tractable densities, denoted by $\mathbb{Q}$. The function $q(\cdot)$ is referred to as variational distribution. Once chosen a divergence or metric $\mathcal{D}\{\cdot, \cdot\}$ over the space of probability distributions, we can define an optimal approximation $\hat{q}(Z) \in \mathbb{Q}$ of the posterior distribution $p(Z \mid X)$ as

$$\hat{q}(Z) = \arg\min_{q \in \mathbb{Q}} \mathcal{D}\{q(\cdot), p(\cdot \mid X)\}.$$

If we choose the Kullback-Leibler (KL) divergence to measure the goodness of the approximation, that is $\mathcal{D}\{\cdot,\cdot\} = \mathrm{KL}\{\cdot\|\cdot\}$, then we obtain what is called as variational Bayes (VB) method. Exploiting the definition of the KL divergence, that is

$$\mathrm{KL}\{q(\cdot)\|p(\cdot \mid X)\} = -\int q(Z) \log \frac{p(Z \mid X)}{q(Z)}\, dZ,$$

it can be shown that

$$\mathrm{KL}\{q(\cdot)\|p(\cdot \mid X)\} = \log p(X) - \mathcal{L}(q(\cdot)) \tag{1.4}$$

where $\mathcal{L}(q(\cdot))$ denotes the evidence lower bound (ELBO), defined as

$$\begin{aligned} \mathcal{L}(q(\cdot)) &= \int q(Z) \log \frac{p(X,Z)}{q(Z)}\, dZ \\ &= \int q(Z) \log p(X,Z)\, dZ - \int q(Z) \log q(Z)\, dZ, \end{aligned} \tag{1.5}$$

and $\log p(X)$ is the log marginal likelihood. The positivity of the KL divergence clarifies the name of $\mathcal{L}(q(\cdot))$, since we can see that $\mathcal{L}(q(\cdot)) \leqslant \log p(X)$. Therefore, the ELBO is a lower bound of the log marginal likelihood, which is assumed to be unknown, otherwise the posterior distribution would be tractable.

From Equation (1.4) we can see that minimizing the KL divergence with respect to the variational distribution is equivalent to maximizing the ELBO. For this purpose, the choice of the family of variational distributions is crucial, as it can affect the feasibility of the method. It is often convenient to restrict $\mathbb{Q}$ to the family of density functions such that

$$q(Z) = \prod_{i=1}^{M} q_i(Z_i)$$

where Z_i denotes the i-th subgroup of the latent variables. This restriction, known as mean-field approximation, assumes the independence between groups, while keeping dependence within them.

Under the latter assumption, the ELBO can be written as

$$\mathcal{L}(q(\cdot)) = \int \log p(X,Z) \prod_{i=1}^{M} q_i(Z_i)\, dZ - \sum_{i=1}^{M} \int q_i(Z_i) \log q_i(Z_i)\, dZ_i.$$

The maximization of the ELBO is achieved numerically, by sequentially optimizing with respect to the i-th group Z_i, keeping the others fixed, and repeating these M steps until a convergence criterion is reached. This optimization algorithm

is know as coordinate ascent variational inference (CAVI). In order to derive the updating rule, we express the ELBO in a convenient form, that is

$$\mathcal{L}(q(\cdot)) = \int q_i(Z_i)\mathbb{E}_{q_{-i}}[\log p(X, Z)]\, dZ - \int q_i(Z_i) \log q_i(Z_i)\, dZ_i + c \tag{1.6}$$

where

$$\mathbb{E}_{q_{-i}}[\log p(X, Z)] = \int \prod_{j \neq i} q_j(Z_j) \log p(X, Z)\, dZ_{-i}$$

is the expectation of the log marginal likelihood with respect to the distributions q_j over all Z_j's for $j \neq i$ and c is a constant that absorbs any integral that does not depend on Z_i.

The ELBO as expressed in Equation (1.6) is the negative KL divergence between $q_i(Z_i)$ and $\exp\{\mathbb{E}_{q_{-i}}[\log p(X, Z)]\}$. Therefore, that quantity is maximized when

$$q_i(Z_i) \propto \exp\left\{\mathbb{E}_{q_{-i}}[\log p(X, Z)]\right\}. \tag{1.7}$$

In other terms, to obtain the optimal solution for the factor q_i, we compute the expectation of the logarithm of the joint distribution of the observed and latent variables with respect to all the factors q_j for $j \neq i$, and then consider its exponential. To make computation easier, it is convenient to take the logarithm of the members of Equation (1.7):

$$\log q_i(Z_i) \propto \mathbb{E}_{q_{-i}}[\log p(X, Z)]. \tag{1.8}$$

Finally, at each step of the CAVI algorithm, each factor q_i for $i = 1, \ldots, M$ is updated according to Equation (1.7). This process is repeated until a maximum number of iterations is reached, or until the increment of the ELBO with respect to the previous iteration becomes negligible. It can be shown that each step of the CAVI algorithm increases the ELBO, yielding a monotonic sequence that converges to a local optimum of the variational objective. See Chapter 5 for examples of this monotonic increase.

Chapter 2

Fusing of Localized Densities

Fusing of Localized Densities (FOLD) is a novel Bayesian clustering method introduced by Dombowsky & Dunson (in press) to address the common issue of kernel misspecification in mixture models, which often leads to the erroneous splitting of true clusters into multiple smaller components. This new approach offers a robust solution by merging these over-split components based on the posterior of the kernels themselves.

Traditional Bayesian clustering relies on mixture models, where each component is interpreted as a distinct cluster. Data is typically clustered by minimizing a loss function that favors similarity to these component labels. However, if the chosen kernels (e.g., Gaussian) don't perfectly match the true underlying data distribution within a cluster, even slight deviations can cause a single, non-Gaussian cluster to be represented by several Gaussian components. This over-clustering phenomenon motivates the development of FOLD, which aims to provide more meaningful and accurate groupings by fusing these erroneously separated components.

FOLD is built upon a Bayesian decision theoretic framework. Instead of directly focusing on the component labels assigned by the mixture model, FOLD utilizes "localized densities" for each data point. These localized densities are defined by the kernel associated with the component to which a data point is assigned. The core idea is to group observations if their respective localized densities are statistically close, thereby encouraging the fusion of overlapping component kernels.

At the heart of the FOLD methodology is a novel loss function designed to counteract cluster splitting. This loss function is a continuous relaxation of Binder's loss function (Binder, 1978), and penalizes assigning two observations to the same cluster if the statistical distance between their localized densities is large, and conversely, penalizes assigning them to different clusters if this distance is

small. Let us denote a clustering with $\hat{\mathbf{c}} = \{\hat{c}_1, \ldots, \hat{c}_n\}$ where each $\hat{c}_i$ corresponds to the label associated with the i-th observation. The loss for a given clustering $\hat{\mathbf{c}}$ is defined as

$$\mathcal{L}(\hat{\mathbf{c}}, \theta) = \sum_{i<j} \left[\mathbb{1}_{\hat{c}_i = \hat{c}_j} \mathcal{D}_{ij} + \omega \mathbb{1}_{\hat{c}_i \neq \hat{c}_j} (1 - \mathcal{D}_{ij}) \right] \tag{2.1}$$

where $\mathcal{D}_{ij} = d\{\mathcal{K}(\cdot; \theta_i), \mathcal{K}(\cdot; \theta_j)\}$ with a unit-bounded statistical distance d, and ω is a positive parameter that calibrates the separation of clusters. In particular, large values of ω promote the merging of clusters. In fact, as $\omega \to \infty$, the partition that minimizes the loss in (2.1) assigns all observations to a single cluster. Conversely, as $\omega \to 0$, the optimal partition places each observation in its own cluster.

The FOLD method determines the optimal clustering using a decision theoretic approach, that leads to an interpretable way to quantify uncertainty, using credible balls. The authors provide an implementation of the procedure using from the output of any MCMC algorithm for mixture models.

We extend the implementation of FOLD using the output obtained when the model is estimated via variational inference.

2.1 Point estimation

FOLD employs a Bayesian decision theoretic approach to determine the optimal clustering. The risk of a particular clustering $\hat{\mathbf{c}}$ is its posterior expected loss:

$$\mathcal{R}(\hat{\mathbf{c}}) = \mathbb{E}[\mathcal{L}(\hat{\mathbf{c}}, \theta) \,|\, \mathbf{X}] = \sum_{i<j} \left[\mathbb{1}_{\hat{c}_i = \hat{c}_j} \Delta_{ij} + \omega \mathbb{1}_{\hat{c}_i \neq \hat{c}_j} (1 - \Delta_{ij}) \right]$$

where $\Delta_{ij} = \mathbb{E}[\mathcal{D}_{ij} \,|\, \mathbf{X}]$ is the posterior expected distance between the localized densities of observations i and j. For a fixed ω, the estimated FOLD clustering, denoted $\mathbf{c}_{\text{FOLD}}$, is the one that minimizes this risk function $\mathcal{R}(\hat{\mathbf{c}})$, that is

$$\mathbf{c}_{\text{FOLD}} = \arg\min_{\hat{\mathbf{c}}} \mathcal{R}(\hat{\mathbf{c}}).$$

2.1.1 Strategies for optimizing the risk

Minimizing $\mathcal{R}(\hat{\mathbf{c}})$ exactly is computationally challenging because the space of all possible partitions of n observations is extraordinarily large, even for modest n. Therefore, the FOLD methodology employs approximation strategies to find $\mathbf{c}_{\text{FOLD}}$. Two primary approaches are discussed.

The first approach involves significantly reducing the search space. Instead of considering all possible partitions, the optimization is restricted to a tree of candidate clusterings generated by performing hierarchical clustering on the $n \times n$ matrix Δ of posterior expected distances Δ_{ij}. Average linkage is suggested for generating these candidates, as the average linkage dissimilarity metric aligns with the method's intrinsic cluster merging criterion. This use of hierarchical clustering to generate a manageable set of candidate partitions is a known heuristic in Bayesian clustering.

A second approach leverages recent advancements in greedy search algorithms designed to minimize risk functions for clustering. These algorithms typically start with an initial candidate partition and iteratively make locally-optimal adjustments, such as reallocating individual observations to different clusters or splitting and merging existing clusters. An example of such an algorithm is SALSO (Search Algorithm via Local Shift Operations), proposed by Dahl et al. (2021), which initializes multiple clusterings and refines them through reallocations and split/merge operations over several parallel runs, ultimately selecting the partition with the lowest observed risk.

According to Dombowsky & Dunson (in press), implementing FOLD using both the hierarchical clustering heuristic and SALSO generally yields consistent optimal clustering results in simulations and applications.

2.1.2 Choice of the loss parameter

The loss minimization requires the choice of a value for ω. This can be achieved in two ways, either manually or automatically.

Let us denote the FOLD point estimator associated with a specific ω with $\mathbf{c}^*_\omega$, and the partition associated with $\mathbf{c}^*_\omega$ with $C^*_\omega = \left\{C^*_{\omega 1}, \ldots, C^*_{\omega K^*_\omega}\right\}$. Now, let us consider the quantity

$$r_\omega = \frac{\sum_{h=1}^{K^*_\omega} r_{\omega h}}{\sum_{i<j} \Delta_{ij}} \quad \text{with} \quad r_{\omega h} = \sum_{i,j \in C^*_{\omega h}} \Delta_{ij}.$$

Note that, as $\omega \to \infty$, the optimal partition places all the observations in the same cluster and then $r_\omega \to 1$. As we decrease the value of ω, more clusters are created, leading to smaller values of r_ω. When $\omega \to 0$, each observation belongs to its own cluster and $r_\omega = 0$. Thus, an elbow plot can be built by evaluating r_ω at a grid of possible ω values. The optimal ω is the value after which the

improvement in r_ω becomes negligible. See Chapter 5 for examples of elbow plots and their interpretations.

On the other hand, one could use the default value

$$\omega_{\text{AVG}} = \frac{\gamma_{\text{AVG}}}{1-\gamma_{\text{AVG}}} \quad \text{with} \quad \gamma_{\text{AVG}} = \binom{n}{2}^{-1} \sum_{i<j} \Delta_{ij}$$

where γ_{AVG} is an estimate of the average pairwise dissimilarity given by $\bar{\mathcal{D}} = \binom{n}{2}^{-1} \sum_{i<j} \mathcal{D}_{ij} = \sum_{k<k'} |S_k||S_{k'}| \binom{n}{2}^{-1} d\{\mathcal{K}(\cdot;\tilde{\theta}_k), \mathcal{K}(\cdot;\tilde{\theta}_{k'})\}$, assuming $S_k = \{i : \theta_i = \tilde{\theta}_k\}$. It has been proven that if $\bar{\mathcal{D}}/(1-\bar{\mathcal{D}})$ is used as the value of ω in the loss function in Equation (2.1), FOLD will promote merging components when $d\{\mathcal{K}(\cdot;\tilde{\theta}_k), \mathcal{K}(\cdot;\tilde{\theta}_{k'})\} < \bar{\mathcal{D}}$. Moreover, the decision to merge two components depends on how separated they are from the others and their sizes. For this reason, ω_{AVG} excels in settings where the true kernels are well-separated but approximated by multiple components.

2.2 Uncertainty quantification

The FOLD framework allows for interpretable uncertainty quantification through measures like credible balls around the point estimate clustering.

The notion of credible balls for Bayesian clustering estimators provided by Wade & Ghahramani (2018) has been extended to the FOLD procedure. The 95% credible ball around c_{FOLD} is defined as

$$B(\mathbf{c}_{\text{FOLD}}) = \{\mathbf{c} : \mathbb{D}(\mathbf{c}_{\text{FOLD}}, \mathbf{c}) \leqslant \epsilon_{\text{FOLD}}\}$$

where $\mathbb{D}$ denotes either the variation of information (Meilă, 2007) or the Binder's loss (Binder, 1978), and $\epsilon_{\text{FOLD}} \geqslant 0$ is the smallest radius such that

$$P(\mathbb{D}(\mathbf{c}, \mathbf{c}_{\text{FOLD}}) \leqslant \epsilon_{\text{FOLD}} \mid \mathbf{X}) \geqslant 0.95.$$

The credible ball characterizes the uncertainty in the clustering estimate, and is interpreted as a neighborhood of clusterings centered at $\mathbf{c}_{\text{FOLD}}$ with posterior probability mass of at least 0.95. A larger value of ϵ_{FOLD} suggests that the FOLD posterior mass is more dispersed around $\mathbf{c}_{\text{FOLD}}$, indicating greater uncertainty in the specific cluster assignments of the point estimate.

To make credible balls more interpretable, Wade & Ghahramani (2018) define specific representative partitions known as bounds, similar to how intervals are described on the real line. The *vertical upper bounds* and the *vertical lower bounds*

respectively consist of the partitions in the credible ball with the smallest and largest number of clusters that are most distant from the clustering point estimate. The *horizontal bounds* consist of the partitions in the credible ball that are most distant from the clustering point estimate.

Formal definitions are given below, where the clustering point estimate is denoted by $\mathbf{c}^*$, and the number of clusters in a clustering $\mathbf{c}$ is denoted by $k(\mathbf{c})$.

Definition 2.1 (vertical upper bounds)**.** The vertical upper bounds of the credible ball $B(\mathbf{c}^*)$, denoted $v^u(\mathbf{c}^*)$, are defined as

$$v^u(\mathbf{c}^*) = \{\mathbf{c} \in B(\mathbf{c}^*) \colon k(\mathbf{c}) \leqslant k(\mathbf{c}') \; \forall \, \mathbf{c}' \in B(\mathbf{c}^*) \text{ and } \mathbb{D}(\mathbf{c}, \mathbf{c}^*) \geqslant \mathbb{D}(\mathbf{c}'', \mathbf{c}^*) \; \forall \, \mathbf{c}'' \in B(\mathbf{c}^*) \text{ with } k(\mathbf{c}) = k(\mathbf{c}'')\}.$$

Definition 2.2 (vertical lower bounds)**.** The vertical lower bounds of the credible ball $B(\mathbf{c}^*)$, denoted $v^l(\mathbf{c}^*)$, are defined as

$$v^l(\mathbf{c}^*) = \{\mathbf{c} \in B(\mathbf{c}^*) \colon k(\mathbf{c}) \geqslant k(\mathbf{c}') \; \forall \, \mathbf{c}' \in B(\mathbf{c}^*) \text{ and } \mathbb{D}(\mathbf{c}, \mathbf{c}^*) \geqslant \mathbb{D}(\mathbf{c}'', \mathbf{c}^*) \; \forall \, \mathbf{c}'' \in B(\mathbf{c}^*) \text{ with } k(\mathbf{c}) = k(\mathbf{c}'')\}.$$

Definition 2.3 (horizontal upper bounds)**.** The horizontal bounds of the credible ball $B(\mathbf{c}^*)$, denoted $h(\mathbf{c}^*)$, are defined as

$$h(\mathbf{c}^*) = \{\mathbf{c} \in B(\mathbf{c}^*) \colon \mathbb{D}(\mathbf{c}, \mathbf{c}^*) \geqslant \mathbb{D}(\mathbf{c}', \mathbf{c}^*) \; \forall \, \mathbf{c}' \in B(\mathbf{c}^*)\}.$$

2.3 Implementation with MCMC output

The FOLD implementation is natively designed to work with Markov chain Monte Carlo outputs. Suppose we have run an MCMC algorithm to estimate a mixture model, producing T samples from the posterior distribution. Each sample consists of the allocations vector $\mathbf{s}^{(t)} = [s_1^{(t)}, \ldots, s_n^{(t)}]$, as well as the set of unique kernel parameters $\tilde{\theta}^{(t)} = [\tilde{\theta}_1^{(t)}, \ldots, \tilde{\theta}_{K^{(t)}}^{(t)}]$. These MCMC samples can be used to estimate the pairwise distance matrix Δ, whose entries Δ_{ij} quantify the dissimilarity between the localized densities associated with observations i and j. Specifically, at each iteration t, we compute the statistical distance

$$\mathcal{D}_{ij}^{(t)} = d\{\mathcal{K}(\cdot; \theta_i^{(t)}), \mathcal{K}(\cdot; \theta_j^{(t)})\}$$

where $\theta_i^{(t)} = \tilde{\theta}_k^{(t)}$ if $s_i^{(t)} = k$. Since this definition depends only on the assigned parameters and not on the label indexing, FOLD is inherently robust to the label-switching problem. The pairwise distances Δ_{ij} are approximated by averaging

across the T iterations, that is

$$\Delta_{ij} \approx \frac{1}{T} \sum_{t=1}^{T} \mathcal{D}_{ij}^{(t)}.$$

Once Δ is estimated, the FOLD clustering $\mathbf{c}_{\text{FOLD}}$ can be obtained by minimizing the loss in Equation (2.1). This optimization can be carried out either manually or automatically, as outlined in Section 2.1.

To assess the uncertainty associated with the FOLD point estimate, we need to generate a sample from the distribution of FOLD clusterings by applying the FOLD method to each replicate distance matrix $\Delta^{(t)}$, where the generic entry is $\mathcal{D}_{ij}^{(t)}$. This yields a sample $\mathbf{c}_{\text{FOLD}}^{(t)}$ for $t = 1, \ldots, T$. These samples can then be used to construct a credible ball around $\mathbf{c}_{\text{FOLD}}$, following the procedure depicted in Section 2.2.

2.4 Implementation with VI output

The FOLD method can also be applied to clustering outputs obtained via variational inference. Suppose a CAVI algorithm has been used to approximate the posterior distribution of a mixture model with H components. This yields an $n \times H$ matrix of responsibilities r, where each row contains the variational posterior probabilities of assignment to the mixture components, as well as variational distributions over the component parameters $\tilde{\theta}_h$ for $h = 1, \ldots, H$.

To construct the distance matrix Δ, we approximate the expected pairwise distances between the localized densities of observations i and j under the variational distribution. Specifically, we compute

$$\Delta_{ij} \approx \mathbb{E}_q[\mathcal{D}_{ij}] = \sum_{h,h'=1}^{H} r_{ih} r_{jh'} \mathbb{E}_q \left[d\{\mathcal{K}(\cdot; \tilde{\theta}_h), \mathcal{K}(\cdot; \tilde{\theta}_{h'})\} \right] \tag{2.2}$$

where $\mathbb{E}_q$ denotes expectation with respect to the variational distribution.

Since the responsibilities are directly available from the CAVI output, the only quantities requiring approximation in Equation (2.2) are the expectations over the component parameters. We propose two strategies for computing these expectations.

The first is a plug-in approximation, where the variational parameters are replaced with their expected values. Specifically, we compute

$$\mathbb{E}_q \left[d\{\mathcal{K}(\cdot; \tilde{\theta}_h), \mathcal{K}(\cdot; \tilde{\theta}_{h'})\} \right] \approx d \left\{ \mathcal{K}(\cdot; \mathbb{E}_q[\tilde{\theta}_h]), \mathcal{K}(\cdot; \mathbb{E}_q[\tilde{\theta}_{h'}]) \right\}.$$

Alternatively, the expectation can be approximated using a Monte Carlo procedure. From the variational distribution of each $\tilde{\theta}_h$, we generate a sample $\tilde{\theta}_h^{(t)}$ for $t = 1, \ldots, T$, and compute the distance between the corresponding densities for each replicate. These distance values are then averaged across replicates to obtain a final estimate.

Because variational approximations often result in sparse responsibility matrices, where only a few components carry substantial weight for each observation, it may be computationally beneficial to ignore negligible contributions. That is, we may approximate Δ_{ij} by summing only over components with non-negligible responsibilities for observations i and j.

Once the matrix Δ has been computed, the FOLD estimate $\mathbf{c}_{\text{FOLD}}$ can be obtained by minimizing the loss in Equation (2.1), using the same strategies outlined in Section 2.1.

Uncertainty quantification in the variational setting follows the same general principles as in the MCMC-based approach, but requires generating both the cluster allocations and the associated group-specific parameters from the variational posterior. For each Monte Carlo replicate, we first sample a set of parameters $\tilde{\theta}_h^{(t)}$ for each component h, and then sample allocations for all observations according to the variational responsibilities. These samples are used to compute a replicate distance matrix $\Delta^{(t)}$, to which the FOLD method is applied, yielding a clustering $\mathbf{c}_{\text{FOLD}}^{(t)}$. Repeating this process over multiple replicates provides a collection of clusterings that can be used to construct a credible ball around the point estimate $\mathbf{c}_{\text{FOLD}}$, following the procedure described in Section 2.2.

Chapter 3

Dirichlet process Gaussian mixture model

A widely used DPMM when it comes to analyzing continuous data is the Dirichlet process mixture model with Gaussian components and a normal-Wishart base distribution. This model is specified as follows:

$$\begin{aligned} \mathbf{X}_i \mid \boldsymbol{\mu}_i, \Lambda_i &\overset{ind}{\sim} \mathcal{N}_p(\boldsymbol{\mu}_i, \Lambda_i^{-1}) \\ (\boldsymbol{\mu}_i, \Lambda_i) \mid \tilde{P} &\overset{iid}{\sim} \tilde{P} \\ \tilde{P} &\sim DP(\alpha, G_0) \end{aligned} \tag{3.1}$$

where $i = 1, \ldots, n$, and

$$\begin{aligned} G_0(\boldsymbol{\mu}, \Lambda) &= \mathcal{NW}_p(\boldsymbol{\mu}, \Lambda \mid \mathbf{m}_0, \Lambda^{-1}/\beta_0, W_0, \nu_0) \\ &= \mathcal{N}_p(\boldsymbol{\mu} \mid \mathbf{m}_0, \Lambda^{-1}/\beta_0)\mathcal{W}_p(\Lambda \mid W_0, \nu_0). \end{aligned}$$

To simplify the notation, we define the following objects:

$$\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_n], \quad \mathbf{Z} = [Z_1, \ldots, Z_n],$$

$$\tilde{\boldsymbol{\mu}} = (\tilde{\boldsymbol{\mu}}_j)_{j=1}^{\infty}, \quad \tilde{\boldsymbol{\Lambda}} = (\tilde{\Lambda}_j)_{j=1}^{\infty}, \quad \boldsymbol{\pi} = (\pi_j)_{j=1}^{\infty}.$$

By introducing the cluster assignment latent variables Z_i, the model becomes

$$\begin{aligned} \mathbf{X}_i \mid Z_i, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}} &\overset{ind}{\sim} \mathcal{N}_p(\tilde{\boldsymbol{\mu}}_{Z_i}, \tilde{\Lambda}_{Z_i}^{-1}) \\ Z_i \mid \boldsymbol{\pi} &\overset{iid}{\sim} Categorical(\boldsymbol{\pi}) \\ \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}} &\overset{iid}{\sim} \mathcal{NW}_p(\tilde{\boldsymbol{\mu}}_j, \tilde{\Lambda}_j \mid \mathbf{m}_0, \tilde{\Lambda}_j^{-1}/\beta_0, W_0, \nu_0) \\ \boldsymbol{\pi} &\sim GEM(\alpha) \end{aligned} \tag{3.2}$$

for $i = 1, \ldots, n$ and $j = 1, 2, \ldots$.

We can now study the posterior distribution of the model, that is the distribution of both model parameters $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\Lambda}}$, and the latent variables $\mathbf{Z}$ and $\boldsymbol{\pi}$, conditional on the data $\mathbf{X}$. As usual when analyzing posterior distributions, we can ignore the normalizing constant and focus on the joint distribution of $\mathbf{X}$, $\mathbf{Z}$, $\tilde{\boldsymbol{\mu}}$, $\tilde{\boldsymbol{\Lambda}}$, and $\boldsymbol{\pi}$. This distribution factorizes in the complete-data likelihood $p(\mathbf{X} \mid \mathbf{Z}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})$, the allocation distribution $p(\mathbf{Z} \mid \boldsymbol{\pi})$, the prior over mixing weights $p(\boldsymbol{\pi})$, and the prior over the component-specific parameters $p(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})$. That is

$$p(\mathbf{Z}, \boldsymbol{\pi}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}} \mid \mathbf{X}) \propto p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\pi}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}) = p(\mathbf{X} \mid \mathbf{Z}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}) p(\mathbf{Z} \mid \boldsymbol{\pi}) p(\boldsymbol{\pi}) p(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}) \tag{3.3}$$

where

$$p(\mathbf{X} \mid \mathbf{Z}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}) = \prod_{i=1}^{n} \prod_{j=1}^{\infty} \mathcal{N}_p(\mathbf{X}_i \mid \tilde{\boldsymbol{\mu}}_j, \tilde{\Lambda}_j^{-1})^{\mathbb{1}_{\{Z_i = j\}}}$$

$$p(\mathbf{Z} \mid \boldsymbol{\pi}) = \prod_{i=1}^{n} \text{Categorical}(Z_i \mid \boldsymbol{\pi})$$

$$\begin{aligned} p(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}) &= p(\tilde{\boldsymbol{\mu}} \mid \tilde{\boldsymbol{\Lambda}}) p(\tilde{\boldsymbol{\Lambda}}) \\ &= \prod_{j=1}^{\infty} \mathcal{N}_p(\tilde{\boldsymbol{\mu}}_j \mid \mathbf{m}_0, \tilde{\Lambda}_j^{-1}/\beta_0) \mathcal{W}_p\left(\tilde{\Lambda}_j \mid W_0, \nu_0\right). \end{aligned}$$

The prior over the infinite sequence of mixing weights $\boldsymbol{\pi}$ is defined by the stick-breaking construction. The distribution $p(\boldsymbol{\pi})$ is then implicitly defined by the joint distribution of the independent V_j variables:

$$p(\boldsymbol{\pi}) = \prod_{j=1}^{\infty} \text{Beta}(V_j \mid 1, \alpha).$$

The infinite-dimensional parameter space makes the posterior distribution analytically intractable, requiring specific methods for inference. The following sections provide an MCMC method and the CAVI algorithm, both tailored for the Dirichlet process Gaussian mixture model DPGMM with a normal-Wishart prior.

3.1 Inference via sampling

Inference in Dirichlet process Gaussian mixture models can be performed using various sampling-based methods. In this section, we focus on the Algorithm 8 of Neal (2000), presented in Subsection 1.3.1, and implement it for a DPGMM with a normal-Wishart prior.

In a DPGMM, each cluster parameter $\tilde{\theta}_z$ consists of a location vector $\tilde{\mu}_z$ and a precision matrix $\tilde{\Lambda}_z$. When drawing new values for the auxiliary parameters $(\tilde{\mu}_z, \tilde{\Lambda}_z)$ from the base distribution G_0, we need to simulate from the normal-Wishart distribution:

$$(\tilde{\mu}_z, \tilde{\Lambda}_z) \sim \mathcal{NW}_p(m_0, \tilde{\Lambda}_z^{-1}/\beta_0, W_0, \nu_0).$$

This can be done by sampling $\tilde{\Lambda}_z$ from $\mathcal{W}_p(W_0, \nu_0)$, and then $\tilde{\mu}_z \mid \tilde{\Lambda}_z$ from $\mathcal{N}_p\left(m_0, \tilde{\Lambda}_z^{-1}/\beta_0\right)$.

When updating the cluster assignment for the i-th observation, we compute the assignment probabilities using the kernel

$$\mathcal{K}(X_i \mid \tilde{\mu}_z, \tilde{\Lambda}_z) = \mathcal{N}_p(X_i \mid \tilde{\mu}_z, \tilde{\Lambda}_z^{-1})$$

which is the multivariate Gaussian density evaluated at X_i.

Finally, to update the cluster parameters $(\tilde{\mu}_z, \tilde{\Lambda}_z)$ for clusters with assigned observations, we exploit the conjugacy of the normal-Wishart prior to the Gaussian model. Given the data assigned to cluster z, denoted by $X_z = \{X_i : Z_i = z\}$, the posterior distribution is:

$$\begin{aligned}\tilde{\mu}_z, \tilde{\Lambda}_z \mid X_z &\sim \mathcal{NW}_p(\tilde{\mu}_z, \tilde{\Lambda}_z \mid m_0, \tilde{\Lambda}_z^{-1}/\beta_0, W_0, \nu_0) \prod_{i:Z_i=z} \mathcal{N}_p(X_i \mid \tilde{\mu}_z, \tilde{\Lambda}_z^{-1}) \\ &\sim \mathcal{NW}_p(\tilde{\mu}_z, \tilde{\Lambda}_z \mid m_z, \tilde{\Lambda}_z^{-1}/\beta_z, W_k, \nu_k)\end{aligned}$$

where

$$\begin{aligned}\beta_z &= \beta_0 + n_z \\ m_z &= \frac{\beta_0 m_0 + n_z \bar{X}_z}{\beta_z} \\ W_z^{-1} &= W_0^{-1} + n_z S_z + \frac{\beta_0 n_z}{\beta_z}(\bar{X}_z - m_0)(\bar{X}_z - m_0)^\intercal \\ \nu_z &= \nu_0 + n_z\end{aligned}$$

with $n_z = \text{Card}(X_i)$, $\bar{X}_z = \sum_{i:\, Z_i=z} X_i/n_z$ and $S_z = \sum_{i:\, Z_i=z}(X_i - \bar{X}_z)(X_i - \bar{X}_z)^\intercal/n_z$. Thus, similarly to the update for auxiliary cluster parameters, we sample $\tilde{\Lambda}_z \mid X_z$ from $\mathcal{W}_p(W_z, \nu_z)$, and then $\tilde{\mu}_z \mid \tilde{\Lambda}_z, X_z$ from $\mathcal{N}_p(m_z, \tilde{\Lambda}_z^{-1}/\beta_z)$.

With this setup, we are equipped to run MCMC-based inference in the DPGMM and analyze posterior clustering behavior and component parameters.

After estimating the model, we obtain a sequence of partitions generated at each iteration of the MCMC algorithm. As suggested by Wade & Ghahramani (2018), to select the optimal partition, we minimize a clustering loss function

such as the variation of information (Meilă, 2007) or the Binder's loss (Binder, 1978) a posteriori. Since the space of all possible partitions is too large to explore exhaustively, we restrict our search to a subset of candidate partitions. This subset is obtained by applying hierarchical clustering to a distance matrix derived from the posterior similarity matrix, where the entry in position i, j is estimated as

$$P(Z_i = Z_j \mid \mathbf{X}) \approx \frac{1}{R - B} \sum_{r=B+1}^{R} \mathbb{1}_{(Z_i^{(r)} = Z_j^{(r)})}$$

where $Z_i^{(r)}$ denotes the cluster assignment of observation i at iteration r, B is the number of burn-in iterations, and R is the total number of MCMC samples. The corresponding distance matrix has entries

$$d_{ij} = 1 - P(Z_i = Z_j \mid \mathbf{X})$$

and hierarchical clustering with average or complete linkage is applied to its estimate to generate a set of candidate partitions. The final partition is chosen as the one within the candidate set that minimizes the expected loss under the posterior, where the expectation is approximated using the posterior similarity matrix.

3.2 Inference via optimization

We now derive the CAVI algorithm for the model in Equation (3.2), using the finite Dirichlet approximation from Equation (1.3). This approximation limits the maximum number of clusters to H. Consequently, the cluster assignment variables will follow a multinomial distribution with one trial and probabilities $\boldsymbol{\pi}$, that is

$$Z_i \sim \mathrm{Multinomial}(1, \boldsymbol{\pi})$$

for $i = 1, \ldots, n$, and

$$\tilde{\boldsymbol{\mu}} = [\tilde{\mu}_1, \ldots, \tilde{\mu}_H] \quad \text{and} \quad \tilde{\boldsymbol{\Lambda}} = [\tilde{\Lambda}_1, \ldots, \tilde{\Lambda}_H].$$

With this setup, we can derive the CAVI algorithm similarly to what has been done in Bishop (2006).

We consider the class of variational distributions that factorize as follows:

$$q(\mathbf{Z}, \boldsymbol{\pi}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}) = q(\mathbf{Z})\, q(\boldsymbol{\pi}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}).$$

This assumption implies independence between the latent variables and the parameters.

The update equations are determined by using the generic result in Equation (1.8).

We begin from the factor $q(\mathbf{Z})$. The log optimal factor is

$$\log q^\star(\mathbf{Z}) \propto \mathbb{E}_{\boldsymbol{\pi},\tilde{\boldsymbol{\mu}},\tilde{\boldsymbol{\Lambda}}} \left[\log p(\mathbf{X}, \mathbf{Z}, \boldsymbol{\pi}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})\right]. \tag{3.4}$$

Using the decomposition of the posterior distribution in Equation (3.3) and considering the terms depending on $\mathbf{Z}$, we get

$$\log q^\star(\mathbf{Z}) \propto \mathbb{E}_{\tilde{\boldsymbol{\mu}},\tilde{\boldsymbol{\Lambda}}} \left[\log p(\mathbf{X} \mid \mathbf{Z}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})\right] + \mathbb{E}_{\boldsymbol{\pi}} \left[\log p(\mathbf{Z} \mid \boldsymbol{\pi})\right].$$

It can be shown that

$$\begin{aligned}\mathbb{E}_{\tilde{\boldsymbol{\mu}},\tilde{\boldsymbol{\Lambda}}} \left[\log p(\mathbf{X} \mid \mathbf{Z}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})\right] = \sum_{i=1}^{n}\sum_{h=1}^{H} \frac{z_{ih}}{2}\{&\mathbb{E}_{\tilde{\Lambda}_h}\left[\log|\tilde{\Lambda}_h|\right] - p\log 2\pi \\ &- \mathbb{E}_{\tilde{\mu}_h,\tilde{\Lambda}_h}\left[(\mathbf{x}_i - \tilde{\boldsymbol{\mu}}_h)^\intercal \tilde{\Lambda}_h (\mathbf{x}_i - \tilde{\boldsymbol{\mu}}_h)\right]\}\end{aligned}$$

and that

$$\mathbb{E}_{\boldsymbol{\pi}} \left[\log p(\mathbf{Z} \mid \boldsymbol{\pi})\right] = \sum_{i=1}^{n}\sum_{h=1}^{H} z_{ih}\mathbb{E}_{\pi_h}[\log \pi_h].$$

Substituting the latter two results into Equation (3.4), we obtain that

$$\log q^\star(\mathbf{Z}) \propto \sum_{i=1}^{n}\sum_{h=1}^{H} z_{ih} \log \rho_{ih} \tag{3.5}$$

where

$$\begin{aligned}\log \rho_{ih} = \mathbb{E}_{\pi_h}\left[\log \pi_h\right] + \frac{1}{2}\mathbb{E}_{\tilde{\Lambda}_h}\left[\log|\tilde{\Lambda}_h|\right] - \frac{p}{2}\log(2\pi) \\ - \frac{1}{2}\mathbb{E}_{\tilde{\mu}_h,\tilde{\Lambda}_h}\left[(\mathbf{x}_i - \tilde{\boldsymbol{\mu}}_h)^\intercal \tilde{\Lambda}_h (\mathbf{x}_i - \tilde{\boldsymbol{\mu}}_h)\right].\end{aligned} \tag{3.6}$$

Taking the exponential of both sides of Equation (3.5) and normalizing with respect to $h = 1, \ldots, H$, we can see that each $\mathbf{Z}_i$ has a multinomial variational distribution. In particular,

$$q^\star(\mathbf{Z}) = \prod_{i=1}^{n} \mathtt{Multinomial}(\mathbf{Z}_i \mid 1, \mathbf{r}_i) \tag{3.7}$$

where $\mathbf{r}_i = [r_{i1}, \ldots, r_{iH}]$ with

$$r_{ih} = \frac{\rho_{ih}}{\sum_{k=1}^{H} \rho_{ik}} \tag{3.8}$$

for $i = 1, \ldots, n$ and $h = 1, \ldots, H$.

It is now useful to define some statistics of the observed data, as they will arise from the next calculations:

$$\begin{aligned} N_h &= \sum_{i=1}^{n} r_{ih} \\ \bar{\mathbf{x}}_h &= \frac{1}{N_h} \sum_{i=1}^{n} r_{ih} \mathbf{x}_i \\ S_h &= \frac{1}{N_h} \sum_{i=1}^{n} r_{ih} (\mathbf{x}_i - \bar{\mathbf{x}}_h)(\mathbf{x}_i - \bar{\mathbf{x}}_h)^\intercal \end{aligned} \tag{3.9}$$

for $h = 1, \ldots, H$. Let us determine the optimal factor $q(\boldsymbol{\pi}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})$. Again, using Equation (1.8), we have that

$$\log q^\star(\boldsymbol{\pi}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}) \propto \mathbb{E}_{\mathbf{Z}} \left[\log p(\mathbf{X} \mid \mathbf{Z}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}) \right] + \log p(\mathbf{Z} \mid \boldsymbol{\pi}) + \log p(\boldsymbol{\pi}) + \log p(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}). \tag{3.10}$$

Since $\boldsymbol{\pi}$ never appears in terms also involving $\tilde{\boldsymbol{\mu}}$ or $\tilde{\boldsymbol{\Lambda}}$, $\boldsymbol{\pi}$ and $(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})$ are independent under the variational distribution. Focusing on the terms of Equation (3.10) that depend on $\boldsymbol{\pi}$, we can determine $q^\star(\boldsymbol{\pi})$:

$$\log q^\star(\boldsymbol{\pi}) \propto \mathbb{E}_{\mathbf{Z}} \left[\log p(\mathbf{Z} \mid \boldsymbol{\pi}) \right] + \log p(\boldsymbol{\pi})$$

where, using the result in Equation (3.7), it can be shown that

$$\mathbb{E}_{\mathbf{Z}} \left[\log p(\mathbf{Z} \mid \boldsymbol{\pi}) \right] = \sum_{i=1}^{n} \sum_{h=1}^{H} r_{ih} \log \pi_h = \sum_{h=1}^{H} N_h \log \pi_h.$$

Thus,

$$\begin{aligned} \log q^\star(\boldsymbol{\pi}) &\propto \left(\frac{\alpha}{H} - 1 \right) \sum_{h=1}^{H} \log \pi_h + \sum_{h=1}^{H} N_h \log \pi_h \\ &\propto \left(\frac{\alpha}{H} + N_h - 1 \right) \sum_{h=1}^{H} \log \pi_h. \end{aligned} \tag{3.11}$$

Again, by taking the exponential of both sides of Equation (3.11), we realize that $\boldsymbol{\pi}$ has a Dirichlet variational distribution, that is

$$q(\boldsymbol{\pi}) = \text{Dirichlet}(\boldsymbol{\pi} \,|\, \boldsymbol{\alpha})$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_H]$ with

$$\alpha_h = \frac{\alpha}{H} + N_h \tag{3.12}$$

for $h = 1, \ldots, H$.

Finally, we can derive the optimal factor $q(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})$. By considering the terms of Equation (3.10) that depend on $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\Lambda}}$, we obtain that

$$\log q^\star(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}) \propto \mathbb{E}_Z \left[\log p(\mathbf{X} \,|\, \mathbf{Z}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})\right] + \log p(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})$$

where

$$\mathbb{E}_Z \left[\log p(\mathbf{X} \,|\, \mathbf{Z}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})\right] = \sum_{i=1}^{n} \sum_{h=1}^{H} r_{ih} \log \mathcal{N}_p(\mathbf{x}_i \,|\, \tilde{\boldsymbol{\mu}}_h, \tilde{\Lambda}_h^{-1}).$$

Consequently, we can write that

$$q^\star(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}) \propto \prod_{h=1}^{H} \mathcal{N}_p(\mathbf{x}_i \,|\, \tilde{\boldsymbol{\mu}}_h, \tilde{\Lambda}_h^{-1})^{r_{ih}} \mathcal{N}_p(\tilde{\boldsymbol{\mu}}_h \,|\, \mathbf{m}_0, \tilde{\Lambda}_h^{-1}/\beta_0) \mathcal{W}_p(\tilde{\Lambda}_h \,|\, W_0, \nu_0).$$

Now, if we focus on a fixed h and use the conjugacy property of the Gaussian-Wishart prior to the Gaussian model, we can determine the joint variational distribution of $\tilde{\boldsymbol{\mu}}_h$ and $\tilde{\Lambda}_h$:

$$\begin{aligned} q^\star(\tilde{\boldsymbol{\mu}}_h, \tilde{\Lambda}_h) &= q^\star(\tilde{\boldsymbol{\mu}} \,|\, \tilde{\Lambda}_h) q^\star(\tilde{\Lambda}_h) \\ &= \mathcal{N}_p(\tilde{\boldsymbol{\mu}}_h \,|\, \mathbf{m}_h, \tilde{\Lambda}_h^{-1}/\beta_h) \mathcal{W}_p(\tilde{\Lambda}_h \,|\, W_h, \nu_h) \end{aligned}$$

where

$$\begin{aligned} \beta_h &= \beta_0 + N_h \\ \mathbf{m}_h &= \frac{\beta_0 \mathbf{m}_0 + N_h \bar{\mathbf{x}}_h}{\beta_h} \\ W_h^{-1} &= W_0^{-1} + N_h S_h + \frac{\beta_0 N_h}{\beta_h} (\bar{\mathbf{x}}_h - \mathbf{m}_0)(\bar{\mathbf{x}}_h - \mathbf{m}_0)^\intercal \\ \nu_h &= \nu_0 + N_h. \end{aligned} \tag{3.13}$$

Now that we determined the functional form of variational distribution, we can complete the expression of the responsibilities by computing the three

expectations that appear in Equation (3.6). By using the properties of the Dirichlet, Wishart, and Gaussian distributions, it is easy to see that

$$\begin{aligned}
\mathbb{E}_{\pi_h}[\log \pi_h] &= \psi(\alpha_h) - \psi(\alpha^+) \\
\mathbb{E}_{\tilde{\Lambda}_h}[\log|\tilde{\Lambda}_h|] &= \psi_p\left(\frac{\nu_h}{2}\right) + p\log 2 + \log|W_h| \\
\mathbb{E}_{\tilde{\mu}_h,\tilde{\Lambda}_h}\left[(\mathbf{x}_i - \tilde{\boldsymbol{\mu}}_h)^\intercal \tilde{\Lambda}_h(\mathbf{x}_i - \tilde{\boldsymbol{\mu}}_h)\right] &= \frac{p}{\beta_h} + \nu_h(\mathbf{x}_i - \mathbf{m}_h)^\intercal W_h(\mathbf{x}_i - \mathbf{m}_h)
\end{aligned}$$

where $\psi(\cdot)$ and $\psi_p(\cdot)$ denote respectively the univariate and the p-variate digamma functions, and $\alpha^+ = \sum_{h=1}^{H} \alpha_h$.

We are left to determine the expression of the ELBO, which is both useful for convergence checks and for validating the implementation of the algorithm, as its value must be monotonically non-decreasing across iterations. We recall the decomposition of the joint distribution of the observed data, latent variables, and parameters in Equation (3.3) to plug it into the expression of the ELBO in Equation (1.5). Thereby, we obtain that

$$\begin{aligned}
\mathcal{L}(q^\star(\cdot)) &= \mathbb{E}\left[\log p(\mathbf{X} \mid \mathbf{Z}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}) p(\mathbf{Z} \mid \boldsymbol{\pi}) p(\boldsymbol{\pi}) p(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})\right] - \mathbb{E}\left[\log q^\star(\mathbf{Z}, \boldsymbol{\pi}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})\right] \\
&= \mathbb{E}\left[\log p(\mathbf{X} \mid \mathbf{Z}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})\right] + \mathbb{E}\left[\log p(\mathbf{Z} \mid \boldsymbol{\pi})\right] + \mathbb{E}\left[\log p(\boldsymbol{\pi})\right] + \mathbb{E}\left[\log p(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})\right] \\
&\quad - \mathbb{E}\left[\log q^\star(\mathbf{Z})\right] - \mathbb{E}\left[\log q^\star(\boldsymbol{\pi})\right] - \mathbb{E}\left[\log q^\star(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})\right]
\end{aligned}$$

where the expectations are taken with respect to the whole variational distribution. At this stage, it should be easy to show that they are equal to the following results:

$$\begin{aligned}
\mathbb{E}\left[\log p(\mathbf{X} \mid \mathbf{Z}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})\right] = \frac{1}{2}\Bigg\{ &\sum_{h=1}^{H} N_h \mathbb{E}\left[\log|\tilde{\Lambda}_h|\right] - p\log 2\pi \\
&- \sum_{i=1}^{n}\sum_{h=1}^{H} r_{ih}\mathbb{E}\left[(\mathbf{x}_i - \tilde{\boldsymbol{\mu}}_h)^\intercal \tilde{\Lambda}_h(\mathbf{x}_i - \tilde{\boldsymbol{\mu}}_h)\right] \Bigg\}
\end{aligned}$$

$$\mathbb{E}\left[\log p(\mathbf{Z} \mid \boldsymbol{\pi})\right] = \sum_{i=1}^{n}\sum_{h=1}^{H} r_{ih}\mathbb{E}\left[\log \pi_h\right]$$

$$\mathbb{E}\left[\log p(\boldsymbol{\pi})\right] = \left(\frac{\alpha}{H} - 1\right)\sum_{h=1}^{H}\mathbb{E}\left[\log \pi_h\right] - \log \mathcal{B}\left(\frac{\alpha}{H}\mathbf{1}_H\right)$$

$$\begin{aligned}
\mathbb{E}\left[\log p(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})\right] = &-\frac{pH}{2}\log 2\pi - \frac{\nu_0 pH}{2}\log 2 + \frac{pH}{2}\log\beta_0 - \frac{p}{2}\sum_{h=1}^{H}\frac{\beta_0}{\beta_h} \\
&- \frac{\beta_0}{2}\sum_{h=1}^{H}\nu_h(\tilde{\boldsymbol{\mu}}_h - \boldsymbol{m}_0)^\intercal W_h(\tilde{\boldsymbol{\mu}}_h - \boldsymbol{m}_0) - \frac{\nu_h}{2}\log|W_0| \\
&+ \frac{1}{2}(\nu_0 - p)\sum_{h=1}^{H}\mathbb{E}\left[\log|\tilde{\Lambda}_h|\right] - \frac{1}{2}\sum_{h=1}^{H}\nu_h \operatorname{Tr}(W_0^{-1}W_h) \\
&- H\log\Gamma_p\left(\frac{\nu_0}{2}\right)
\end{aligned}$$

$$\mathbb{E}\left[\log q^\star(\boldsymbol{Z})\right] = \sum_{i=1}^{n}\sum_{h=1}^{H} r_{ih}\log r_{ih}$$

$$\mathbb{E}\left[\log q^\star(\boldsymbol{\pi})\right] = \sum_{h=1}^{H}(\alpha_h - 1)\mathbb{E}[\log\pi_h] - \log\mathcal{B}(\boldsymbol{\alpha})$$

$$\begin{aligned}
\mathbb{E}\left[\log q^\star(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}})\right] = &-\frac{pH}{2}\log 2\pi + \frac{p}{2}\sum_{h=1}^{H}\log\beta_h - \frac{pH}{2} \\
&+ \frac{1}{2}\sum_{h=1}^{H}(\nu_h - p)\mathbb{E}\left[\log|\tilde{\Lambda}_h|\right] - \frac{p}{2}\sum_{h=1}^{H}\nu_h \\
&- \frac{p}{2}\log 2\sum_{h=1}^{H}\nu_h - \frac{1}{2}\sum_{h=1}^{H}\nu_h\log|W_h| - \sum_{h=1}^{H}\log\Gamma_p\left(\frac{\nu_h}{2}\right)
\end{aligned}$$

where $\mathcal{B}(\cdot)$ denotes the normalizing constant of the Dirichlet distribution, and $\Gamma_p(\cdot)$ denotes the p-variate gamma function, arising from the normalizing constant of the Wishart distribution.

After performing some simplifications, we are left with the following expression of the lower bound:

$$\begin{aligned}\mathcal{L}(q^\star(\cdot)) = &\sum_{h=1}^{H} \log\Gamma(\alpha_h) - \log\Gamma\left(\alpha^+\right) - H\log\Gamma\left(\frac{\alpha_0}{H}\right) + \log\Gamma(\alpha_0) \\ &- \sum_{i=1}^{n}\sum_{h=1}^{H} r_{ih}\log r_{ih} + \frac{pH}{2}\log\beta_0 - \frac{p}{2}\sum_{h=1}^{H}\log\beta_h \\ &- \frac{\beta_0}{2}\sum_{h=1}^{H}\nu_h(m_h - m_0)^\intercal W_h(m_h - m_0) \\ &- \frac{1}{2}\sum_{i=1}^{n}\sum_{h=1}^{H}\nu_h r_{ih}(x_i - m_h)^\intercal W_h(x_i - m_h) \\ &- \frac{1}{2}\sum_{h=1}^{H}\nu_h \operatorname{Tr}(W_0^{-1}W_h) + \frac{1}{2}\sum_{h=1}^{H}\nu_h \log|W_h| - \frac{\nu_0 H}{2}\log|W_0| \\ &+ \sum_{h=1}^{H}\log\Gamma_p\left(\frac{\nu_h}{2}\right) - H\log\Gamma_p\left(\frac{\nu_0}{2}\right) + \frac{1}{2}\sum_{h=1}^{H}\nu_h - \frac{np}{2}\log\pi\end{aligned} \tag{3.14}$$

where $\Gamma(\cdot)$ denotes the univariate gamma function, arising from the normalizing constant of the Dirichlet distribution.

In summary, the CAVI algorithm consists of the following steps:

1. Initialize the responsibilities.
2. Compute the statistics in Equation (3.9).
3. Update the hyperparameters according to Equations (3.12), (3.13), and (3.8).
4. Compute the ELBO in Equation (3.14).
5. Check for convergence. If convergence has not been reached, return to step 2; otherwise, terminate the algorithm.

We conclude this section with a few of remarks regarding the algorithm and the interpretation of its output clustering purposes.

Initialization. For the algorithm to proceed with the computations of the statistics on the observed data, initial values for the responsibilities are essential. Therefore, we initialize each $\mathbf{r}_i$ with a sample drawn from the prior distribution over the mixing weights, that is generating from a Dirichlet distribution with H parameters all equal to α/H. The algorithm is sensitive to the initialization. For this reason, it is common practice to run it multiple times from different starting points and select the estimate associated with the highest lower bound value.

Termination. There are typically two primary ways to stop the iterations. The first is to specify a maximum number of iterations to prevent indefinite running. The second method involves assessing convergence, which is usually achieved when the relative (or absolute) increment of a monitored objective function, such as the ELBO, between successive iterations falls below a predefined small tolerance threshold ϵ. For instance, in our implementation, the algorithm might stop at iteration t if

$$\frac{\mathcal{L}^t - \mathcal{L}^{t-1}}{|\mathcal{L}^{t-1}|} < \epsilon,$$

for some small ϵ.

Cluster assignment. Upon convergence of the CAVI algorithm, we obtain for each observation i a probability vector r_{ih} for $h = 1, \ldots, H$, indicating the posterior responsibility of cluster h. To produce the optimal partition, each observation is assigned to the cluster with the highest posterior responsibility:

$$z_i = \arg \max_{h=1,\ldots,H} r_{ih}.$$

See Appendix A for the R implementation of the CAVI algorithm applied to DPGMMS with normal-Wishart priors.

3.3 FOLD implementations

In this section, we discuss the possible implementations of the FOLD method for the Dirichlet process Gaussian mixture model. Although, this method is very general and can be applied to a wide range of models, the Gaussian kernel allows for some simplifications. In particular, the kernel plays a crucial role when computing the distances between localized densities.

The FOLD loss function is well-specified when a unit-bounded distance between probability distributions is chosen. In their paper, Dombowsky & Dunson (in press) propose the Hellinger distance, as it is bounded in the interval $[0, 1]$ by definition, and its interpretation makes it particularly suitable for the FOLD method. In general, the square of the Hellinger distance between two probability density functions f and g is defined as

$$H^2\{f, g\} = \frac{1}{2} \int \left(\sqrt{f(x)} - \sqrt{g(x)} \right)^2 dx.$$

The Hellinger distance measures the lack of overlap between f and g; that is, it quantifies how much the two distributions disagree in the density that they

assign to different regions of the support. A value of zero indicates that the two distributions are identical (i.e., complete overlap), while a value of 1 indicates that one distribution assigns probability only to regions where the other assigns none (i.e., no overlap).

The Hellinger distance simplifies and admits a closed-form expression when applied to Gaussian distributions. Specifically, the square of the Hellinger distance between $f(\cdot) = \mathcal{N}_p(\cdot \mid \mu_1, \Sigma_1)$ and $g(\cdot) = \mathcal{N}_p(\cdot \mid \mu_2, \Sigma_2)$ is defined as

$$H^2\{f, g\} = 1 - \frac{|\Sigma_1|^{1/4}|\Sigma_2|^{1/4}}{\left|\frac{\Sigma_1+\Sigma_2}{2}\right|^{1/2}} \exp\left\{-\frac{1}{8}(\mu_1 - \mu_2)^\intercal \left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{-1} (\mu_1 - \mu_2)\right\}.$$

This closed-form expression makes the Hellinger distance particularly convenient and efficient to compute when comparing Gaussian distributions, which can significantly speed up computations.

Furthermore, one could want to use a not unit-bounded statistical distance d. In this case, we define the loss of assigning i and j to the same cluster as follows:

$$\mathcal{D}_{ij} = 1 - \exp\{-d(\mathcal{K}(\cdot; \theta_i), \mathcal{K}(\cdot; \theta_j)\}$$

This transformation enables us to explore how the FOLD method performs when different statistical distances are used.

In recent years, the Wasserstein metric (or earth mover's distance) has gained significant attention in machine learning, statistics, and computational optimal transport. It provides a geometrically meaningful way to compare probability distributions by quantifying the "cost" of optimally transporting mass from one distribution to another (Peyré & Cuturi, 2019). Unlike unit-bounded divergences such as the Hellinger distance, the Wasserstein distance is not bounded, but offers a more intuitive notion of discrepancy.

Formally, for $p \geqslant 1$, the p-Wasserstein distance between two probability distributions μ and ν on an Euclidean space $\mathcal{X}$ is defined as

$$W_p\{\mu, \nu\} = \left[\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{X}} \|\mathbf{x} - \mathbf{y}\|_2^p \, d\gamma(\mu, \nu)\right]^{1/p}$$

where $\Gamma(\mu, \nu)$ is the set of all couplings (i.e., joint distributions) with marginal distributions μ and ν. Note that here p denotes the order of the Wasserstein distance, not the dimension of the sample space. Intuitively, the Wasserstein distance measures the minimum expected cost of transforming one distribution into the other by moving mass through the space.

The Wasserstein metric has an appealing geometric interpretation: in Euclidean spaces, it corresponds to the minimal "effort" required to rearrange the mass of one distribution to match another.

A particularly useful result is the closed-form expression of the 2-Wasserstein distance between two Gaussian distributions. Specifically, the square of the 2-Wasserstein distance between $f(\cdot) = \mathcal{N}_p(\cdot \mid \mu_1, \Sigma_1)$ and $g(\cdot) = \mathcal{N}_p(\cdot \mid \mu_2, \Sigma_2)$ is defined as

$$W_2^2\{f, g\} = \|\mu_1 - \mu_2\|_2^2 + \mathrm{Tr}\left[\Sigma_1 + \Sigma_2 - 2\left(\Sigma_1^{1/2}\Sigma_2\Sigma_1^{1/2}\right)^{1/2}\right]$$

This formula shows the decomposition of the Wasserstein distance into a mean displacement term and a covariance mismatch term. The first term measures how far apart the two distributions are. The second term, on the other hand, quantifies the difference in the shape, scale, and orientation of the two distributions.

Thanks to this closed form, the 2-Wasserstein distance can be computed efficiently and used to explore the behavior of the FOLD method under geometrically-aware statistical distances.

The FOLD method is implemented by setting $d\{\cdot, \cdot\}$ to be either the Hellinger distance $H\{\cdot; \cdot\}$ or the transformed 2-Wasserstein distance $1 - \exp\{-W_2\{\cdot; \cdot\}\}$.

Finally, when computing the plug-in approximation of the expectation $\mathbb{E}_q[d\{\mathcal{N}_p(\cdot \mid \tilde{\mu}_h, \tilde{\Lambda}_h^{-1}), \mathcal{N}_p(\cdot \mid \tilde{\mu}_{h'}, \tilde{\Lambda}_{h'}^{-1})\}]$, we compute the following expected values with respect to the variational distribution:

$$\mathbb{E}_q[\tilde{\mu}_h] = m_h \quad \text{and} \quad \mathbb{E}_q\left[\tilde{\Lambda}_h^{-1}\right] = \frac{W_h}{\nu_h - p - 1}.$$

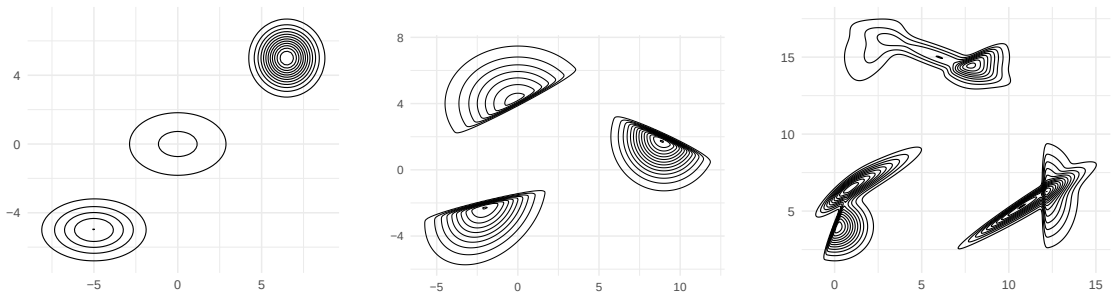
Chapter 4

Simulation studies

This chapter presents a comparative study of the Dirichlet process Gaussian mixture model estimates obtained via variational Bayes and Markov chain Monte Carlo methods. We then apply the FOLD procedure to both outputs to investigate its effectiveness in reducing the number of clusters while preserving their interpretability and meaningfulness.

We design three simulation settings, each introducing an increasing degree of kernel misspecification, to systematically assess model performance under various deviations from model assumptions. The contour plots of the three simulation settings are shown in Figure 4.1.

We compare the number of estimated clusters before and after using the FOLD method. We use the adjusted Rand index (ARI) to measure the meaningfulness of the clusters. As in Hubert & Arabie (1985), the ARI quantifies the degree of agreement between two partitions beyond what would be expected by chance. An ARI of 1 indicates perfect agreement between the partitions. An ARI around 0 indicates that the agreement is at the level expected by chance. Negative values are possible, indicating less agreement than would expected by chance.



(a) Contour plot of the Gaussian mixture.

(b) Contour plot of the skewed Gaussian mixture.

(c) Contour plot of the mixture of mixtures.

Figure 4.1: Contour plots of the three mixtures considered in the simulation studies.

We also take into account the runtime of the FOLD procedures on the VB output and the MCMC output.

To facilitate comparison across the different methods and variants, we use the following labels throughout the plots and tables of the following sections. The models are estimated either via variational Bayes or Markov chain Monte Carlo, without any post-processing. These are referred to as VB (Pre) and MCMC (Pre), respectively. The FOLD method is then applied to these estimates using either the Hellinger distance or the Wasserstein-2 distance, and using either the plug-in approximation or the theoretic formulation. The resulting combinations are denoted accordingly: VB + FOLD-H (Plug-in) and VB + FOLD-W (Plug-in) refer to FOLD applied to the VB output using the plug-in method with the Hellinger and Wasserstein metrics, respectively; VB + FOLD-H and VB + FOLD-W refer to the same but using the theoretic version of FOLD; analogous labels are used for the MCMC-based results: MCMC + FOLD-H, and MCMC + FOLD-W.

In each simulation study, the data are standardized to have zero mean and unit variance along each dimension. The hyperparameter values are chosen to reflect weak prior information. Specifically, the hyperparameters are set as follows:

$$\alpha = 1, \quad \beta_0 = 0.1, \quad \mathbf{m}_0 = \mathbf{0}, \quad \nu_0 = 4, \quad W_0 = I_2.$$

The prior mean $\mathbf{m}_0$ centers the distribution of the cluster means around the origin. The scaling parameter β_0 controls the prior variance of the cluster means, yielding to $\text{Var}[\boldsymbol{\mu}] = 10 I_2$. This allows substantial deviation from $\mathbf{m}_0$. The degrees of freedom ν_0 are set as the smallest value for which the expectation of the covariance matrix Λ^{-1} exists in two dimensions, ensuring a proper yet weakly informative inverse-Wishart prior. The scale matrix W_0 implies $\mathbb{E}[\Lambda^{-1}] = I_2$, which reflects unit variance of the scaled data. Finally, the choice of concentration parameter of the Dirichlet process strikes a balance between promoting parsimony in the number of clusters and allowing the data to drive the discovery of additional components. In this setting, the expected number of clusters is $\mathbb{E}[K_n] \approx 6.21$ for a sample size of $n = 500$, according to the result in Ghosal & van der Vaart (2017).

The model is estimated via variational inference and a Markov chain Monte Carlo method. The variational inference algorithm, implemented in the `R` function in Appendix A, terminates when it reaches 100 iterations, or when the relative increase in the ELBO between successive iterations falls below 10^{-4}. The maximum number of iterations is set to prevent excessive computation when convergence is slow. In practice, this limit is never reached. The MCMC estimation is performed using the marginal sampler implemented in the `R` package `BNPmix` (Corradin

et al., 2021), with the default number of auxiliary clusters ($m = 100$). We generate 10 000, discard the first 1 000 as burn-in, and keep one every three draws. The optimal partition belongs to a set of candidates obtained using hierarchical clustering with average linkage, and minimizes the posterior expectation of the variation of information. Finally, to apply the FOLD method to the VB output, we draw 1 000 samples from the optimal variational distribution.

4.1 Data generated from a Gaussian mixture

In the first simulation study, data are generated from a bivariate Gaussian mixture with three well-separated components. In particular, the data generating process is

$$f_0 = a_1 \mathcal{N}_2(\mu_1, \Sigma_1) + a_2 \mathcal{N}_2(\mu_2, \Sigma_2) + a_3 \mathcal{N}_2(\mu_3, \Sigma_3)$$

where the mixing weights are $(a_1, a_2, a_3) = (0.45, 0.25, 0.3)$, the locations are $\mu_1 = (6.5, 5)$, $\mu_2 = (0, 0)$, and $\mu_3 = (-5, -5)$, and the covariance matrices are $\Sigma_1 = I_2$, $\Sigma_2 = \text{diag}(5, 2)$, and $\Sigma_3 = \text{diag}(3, 2)$.

Figure 4.2, Figure 4.3, and Table 4.1 summarize the average number of clusters, ARI, and FOLD runtime for the different estimation and post-processing methods applied to the bivariate Gaussian mixture.

In this setting, the model is well-specified. Both the VB and MCMC estimates without post-processing yield interpretable and meaningful clusterings. On average, they estimate 4 clusters with high values of the ARI: 0.95 and 0.98, respectively.

Applying FOLD improves the clustering produced by the VB estimate of the model, regardless of the distance or approach. The estimated number of clusters is nearly equal to the true number of clusters, that is 3. The values of the ARI increase to 0.97, indicating a slight refinement of the already good clusterings.

Some variability remains in the VB results due to sensitivity to initialization of the CAVI algorithm, which can occasionally yield suboptimal solutions and lead to poorer clusterings even after FOLD is applied.

FOLD also improves the clusterings produced by the MCMC estimates. The estimated number of clusters is reduced to near the true value, and the ARI maintains a value of 0.98, but with lower variance.

In summary, when the model is correctly specified, applying the FOLD method to the VB estimates yields results similar to those produced by applying FOLD to the MCMC estimates. Additionally, FOLD is significantly faster on VB estimates because it operates on an i.i.d. sample drawn from the variational posterior, which

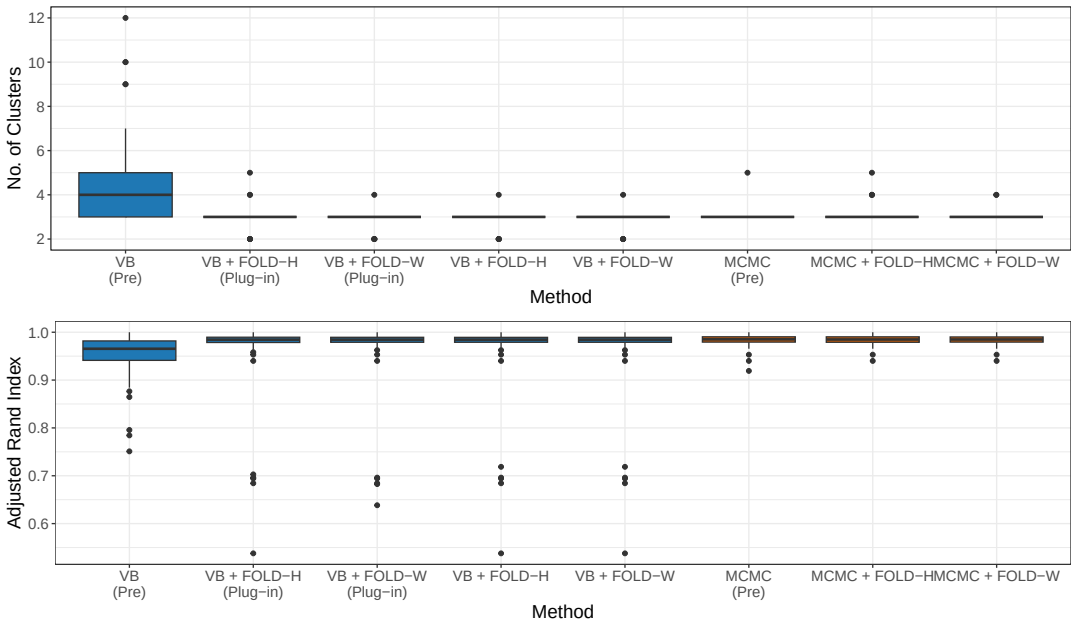


Figure 4.2: Comparison of the number of clusters and adjusted Rand index on 500 observations from a bivariate Gaussian mixture.

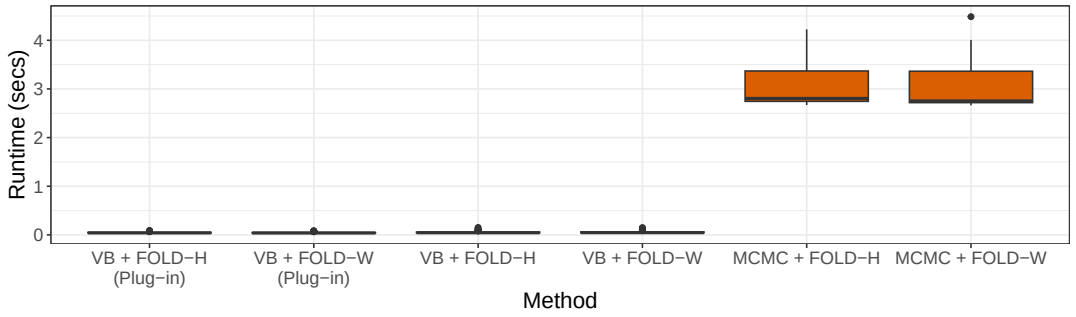


Figure 4.3: Comparison of runtime on 500 observations from a bivariate Gaussian mixture.

is typically smaller and easier to compute than the much larger, correlated sample produced by MCMC. This results in notably reduced runtime for the VB-based FOLD procedures.

4.2 Data generated from a skewed Gaussian mixture

In the second simulation study, data are generated from a bivariate skewed Gaussian mixture. In particular, the data generating process is

$$f_0 = \alpha_1 \mathcal{SN}_2(\boldsymbol{\mu}_1, \Sigma_1, \boldsymbol{\psi}_1) + \alpha_2 \mathcal{SN}_2(\boldsymbol{\mu}_2, \Sigma_2, \boldsymbol{\psi}_2) + \alpha_3 \mathcal{SN}_2(\boldsymbol{\mu}_3, \Sigma_3, \boldsymbol{\psi}_3)$$

where the mixing weights α_1, α_2, and α_3 are all equal to $1/3$, the locations are $\boldsymbol{\mu}_1 = (9, 2)$, $\boldsymbol{\mu}_2 = (0, 4)$, and $\boldsymbol{\mu}_3 = (-2, -2)$, the covariance matrices are $\Sigma_1 = 2\mathrm{I}_2$,

Method	No. of Clusters	ARI	FOLD runtime
VB (Pre)	4.30 (1.679)	0.95 (0.043)	–
VB + FOLD-H (Plug-in)	2.99 (0.333)	0.97 (0.072)	0.05 (0.013)
VB + FOLD-W (Plug-in)	2.96 (0.243)	0.97 (0.068)	0.05 (0.012)
VB + FOLD-H	2.96 (0.243)	0.97 (0.072)	0.05 (0.019)
VB + FOLD-W	2.96 (0.243)	0.97 (0.072)	0.06 (0.021)
MCMC (Pre)	3.95 (0.200)	0.98 (0.012)	–
MCMC + FOLD-H	3.05 (0.261)	0.98 (0.010)	3.01 (0.375)
MCMC + FOLD-W	3.02 (0.141)	0.98 (0.010)	2.95 (0.378)

Table 4.1: Averages and standard deviations (in parentheses) for the number of clusters, adjusted Rand index, and runtime of the FOLD procedure on 500 observations from a bivariate Gaussian mixture.

$\Sigma_2 = \text{diag}(5, 3)$, and $\Sigma_3 = 3I_2$, and the skewness parameters are $\psi_1 = (1, 1)$, $\psi_2 = (-10, 15)$, and $\psi_3 = (4, -17)$.

Figure 4.4, Figure 4.5, and Table 4.2 summarize the average number of clusters, ARI, and FOLD runtime for the different estimation and post-processing methods applied to the bivariate skewed Gaussian mixture.

In this setting, where the data slightly deviate from Gaussianity due to skewness in the mixture components, the performance of both VB and MCMC estimates without post-processing deteriorates compared to the well-specified case. Specifically, the unprocessed VB estimates yield an average of 5.38 clusters with an ARI of 0.91, while the unprocessed MCMC estimates tend to overcluster slightly less, producing an average of 4.63 clusters but with a lower ARI of 0.84. These results indicate that both methods are affected by the model misspecification, though VB appears to offer slightly more stable clustering performance in this case.

Applying the FOLD procedure leads to substantial improvements in both estimation methods. For all combinations of statistical distance and approximation method, FOLD reduces the number of clusters toward the true value and increases the ARI, often approaching perfect clustering accuracy. Notably, all Wasserstein-based FOLD methods, whether applied to VB or MCMC, consistently recover the true partition, with an average ARI of 1 and a very low standard deviation. Hellinger-based approaches also perform well, though with slightly higher variance in the number of clusters.

An exception is observed for the VB + FOLD-H (Plug-in) method. This variant tends to overestimate the number of clusters (mean of 4.43), likely due to limitations of the plug-in approximation. Specifically, the plug-in method can overestimate the distance between localized densities when using the Hellinger

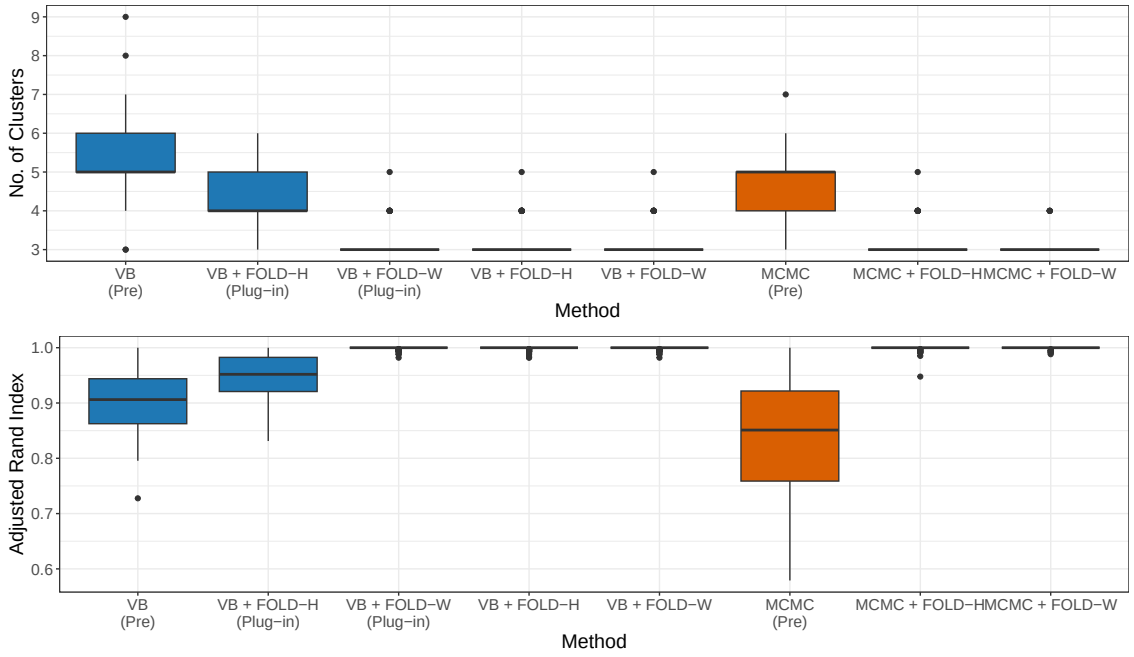


Figure 4.4: Comparison of the number of clusters and adjusted Rand index on 500 observations from a bivariate skewed Gaussian mixture.

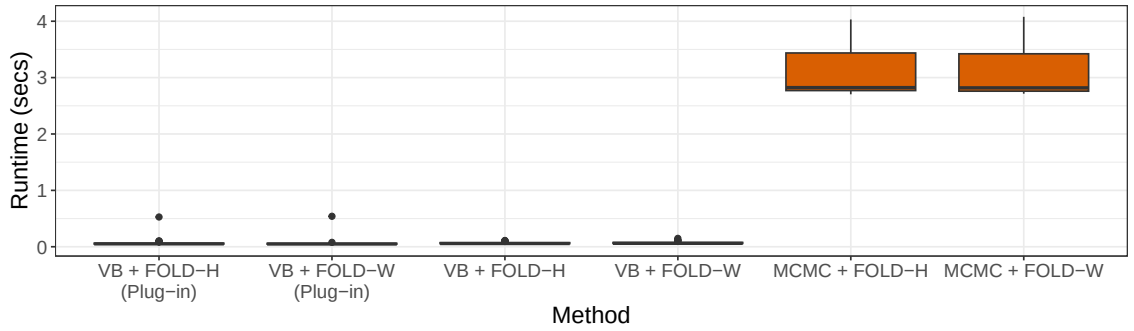


Figure 4.5: Comparison of runtime on 500 observations from a bivariate skewed Gaussian mixture.

distance, preventing the proper merging of nearby clusters. This effect is not present when using the Wasserstein metric, which is more robust to such approximation.

Finally, the computational efficiency of the FOLD procedure is once again evident: runtimes for VB-based methods remain below 0.1 seconds on average, while MCMC-based versions require over 3 seconds due to their reliance on larger, correlated posterior samples.

4.3 Data generated from a mixture of mixtures

In the third simulation study, data are generated from bivariate mixture of three components $f^0 = \alpha_1 g_1^0 + \alpha_2 g_2^0 + \alpha_3 g_3^0$ with mixing weights α_1, α_2, and α_3 all equal

Method	No. of Clusters	ARI	FOLD Runtime
VB (Pre)	5.38 (0.908)	0.91 (0.053)	–
VB + FOLD-H (Plug-in)	4.43 (0.807)	0.95 (0.041)	0.06 (0.049)
VB + FOLD-W (Plug-in)	3.13 (0.367)	1 (0.003)	0.06 (0.050)
VB + FOLD-H	3.19 (0.419)	1 (0.004)	0.06 (0.014)
VB + FOLD-W	3.13 (0.367)	1 (0.003)	0.07 (0.017)
MCMC (Pre)	4.63 (0.928)	0.84 (0.111)	–
MCMC + FOLD-H	3.14 (0.377)	1 (0.006)	3.06 (0.382)
MCMC + FOLD-W	3.05 (0.219)	1 (0.002)	3.05 (0.367)

Table 4.2: Averages and standard deviations (in parentheses) for the number of clusters, adjusted Rand index and runtime of the FOLD procedure on 500 observations from a bivariate skewed Gaussian mixture.

to 1/3. Two components, g_1^0 and g_3^0, are mixtures of two skewed Gaussian kernels:

$$\begin{aligned} g_1^0 = 0.5\,\mathcal{SN}_2\left(\begin{bmatrix}0\\4\end{bmatrix}, I_2, \begin{bmatrix}15\\-5\end{bmatrix}\right) \\ + 0.5\,\mathcal{SN}_2\left(\begin{bmatrix}0\\6\end{bmatrix}, \begin{bmatrix}5 & 3\\3 & 2\end{bmatrix}, \begin{bmatrix}7\\-3\end{bmatrix}\right), \end{aligned}$$

$$\begin{aligned} g_3^0 = 0.5\,\mathcal{SN}_2\left(\begin{bmatrix}12\\6\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 3\end{bmatrix}, \begin{bmatrix}11\\0\end{bmatrix}\right) \\ + 0.5\,\mathcal{SN}_2\left(\begin{bmatrix}0\\6\end{bmatrix}, \begin{bmatrix}5 & 3\\3 & 2\end{bmatrix}, \begin{bmatrix}9\\-7\end{bmatrix}\right). \end{aligned}$$

Instead, the second component g_2^0 is a mixture of two skewed Gaussian kernels and a Gaussian kernel:

$$\begin{aligned} g_2^0 = 1/3\,\mathcal{SN}_2\left(\begin{bmatrix}3\\15\end{bmatrix}, 3I_2, \begin{bmatrix}-3\\2\end{bmatrix}\right) \\ + 1/3\,\mathcal{N}_2\left(\begin{bmatrix}6\\15\end{bmatrix}, \begin{bmatrix}5 & -2\\-2 & 1\end{bmatrix}\right) \\ + 1/3\,\mathcal{SN}_2\left(\begin{bmatrix}8\\15\end{bmatrix}, I_2, \begin{bmatrix}3\\-5\end{bmatrix}\right). \end{aligned}$$

Figure 4.6, Figure 4.7, and Table 4.3 summarize the average number of clusters, ARI, and FOLD runtime for the different estimation and post-processing methods applied to the bivariate mixture of mixtures.

First, we observe that the VB estimate without post-processing results in a number of clusters close to the expected value (6.24), with a high ARI of 0.71.

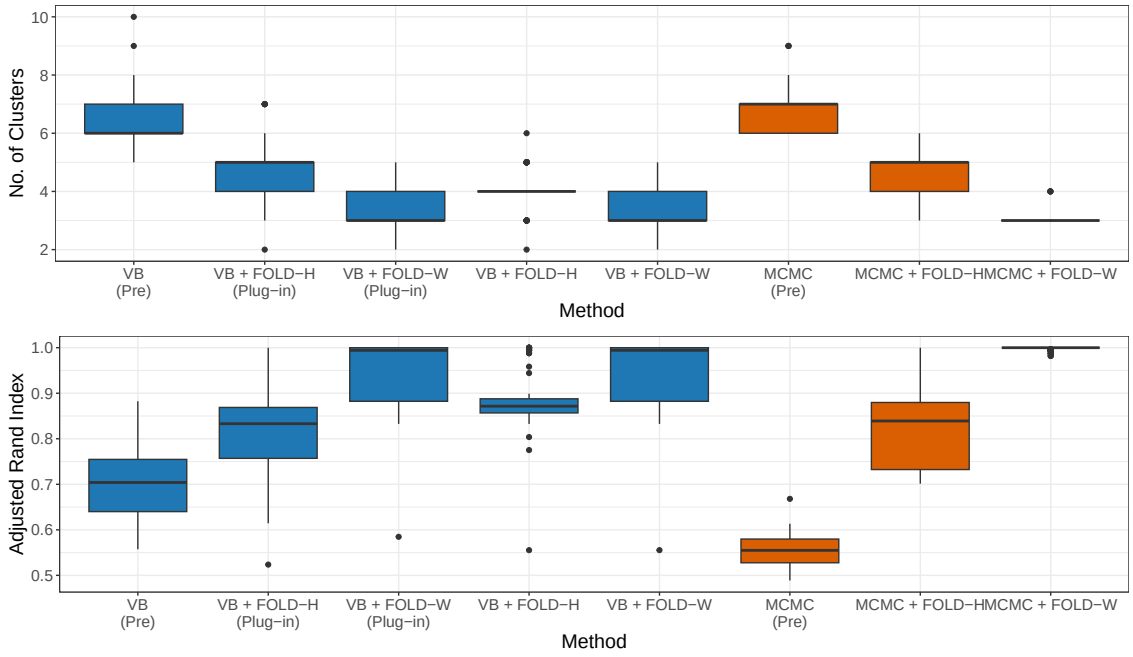


Figure 4.6: Comparison of the number of clusters and adjusted Rand index on 500 observations from a mixture of mixtures.

Applying FOLD improves the clustering quality significantly: both the plug-in and theoretic versions of FOLD-W reduce the number of clusters to about 3.45 and achieve the highest ARI (0.95), indicating near-perfect alignment with the true partition. The Hellinger-based versions also reduce the number of clusters while improving ARI, though to a lesser extent than Wasserstein. Notably, the plug-in methods perform comparably to the theoretic versions, but with slightly faster runtime.

In contrast, the MCMC-based estimate without post-processing overestimates the number of clusters (6.91) and yields a lower ARI of 0.55, suggesting a less accurate clustering. After applying FOLD, the results improve substantially: both FOLD-H and FOLD-W reduce the number of clusters to more interpretable levels (around 4.52 and 3.04, respectively), with FOLD-W achieving a perfect ARI of 1, suggesting exact recovery of the true clustering structure. However, these improvements come at the cost of higher computational time, compared to VB-based FOLD procedures.

Overall, the results show that applying FOLD, especially with the Wasserstein metric, substantially improves the accuracy and parsimony of clustering for both VB and MCMC estimates. While MCMC + FOLD-W achieves the best clustering accuracy, VB + FOLD-W offers a favorable trade-off between performance and computational efficiency.

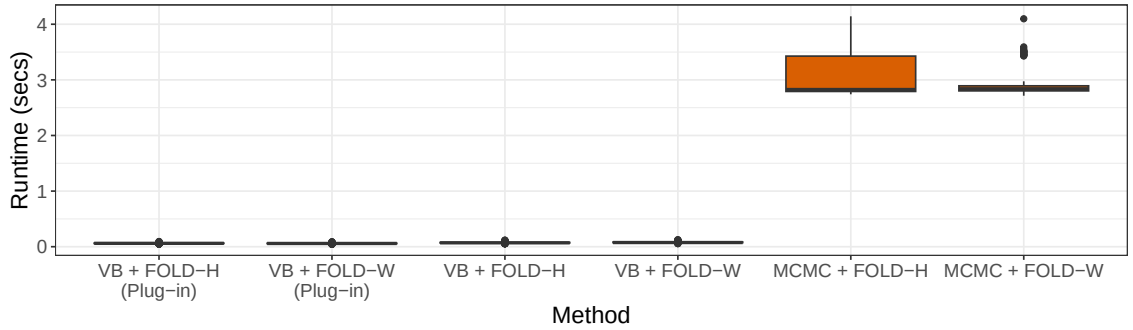


Figure 4.7: Comparison of runtime on 500 observations from a mixture of mixtures.

Method	No. of Clusters	ARI	FOLD Runtime
VB (Pre)	6.24 (0.830)	0.71 (0.079)	—
VB + FOLD-H (Plug-in)	4.82 (0.936)	0.82 (0.080)	0.06 (0.008)
VB + FOLD-W (Plug-in)	3.46 (0.610)	0.95 (0.071)	0.06 (0.008)
VB + FOLD-H	4.01 (0.559)	0.88 (0.060)	0.07 (0.012)
VB + FOLD-W	3.45 (0.609)	0.95 (0.073)	0.08 (0.014)
MCMC (Pre)	6.91 (0.793)	0.55 (0.033)	—
MCMC + FOLD-H	4.52 (0.731)	0.81 (0.090)	3.05 (0.339)
MCMC + FOLD-W	3.04 (0.197)	1 (0.004)	2.95 (0.279)

Table 4.3: Averages and standard deviations (in parentheses) for the number of clusters, adjusted Rand index, and runtime of the FOLD procedure on 500 observations from a bivariate mixture of mixtures.

Chapter 5

Real data applications

In this chapter, we use the DPGMM with a normal-Wishart prior to address clustering in different real scenarios. As in the previous chapter, the model is first estimated via variational Bayes, using the `R` function in Appendix A, and via marginal sampler, using the `R` package `BNPmix`. The CAVI algorithm is run by fixing the maximum number of components at 100, the maximum number of iterations at 100, and a relative increase in the ELBO between successive iterations at 10^{-4}. The marginal sampler is run by letting the number of auxiliary clusters equal to the default value (100). The optimal partition derived from the MCMC estimates belongs to a set of candidates obtained using hierarchical clustering with average linkage, and minimize the posterior expectation of the variation of information.

The FOLD procedure is then applied to each model estimate. In particular, to apply FOLD to VB estimates, we generate 1 000 samples from the variational posterior.

The elbow plot diagnostic is used to select the number of clusters to consider. In each elbow plot, we seek the "elbow" point, that is a point where the metric stops decreasing significantly with the addition of more clusters. The number of clusters at the "elbow" is often the best balance between performance and model complexity.

Finally, each data partition is evaluated with the adjusted Rand index.

5.1 Yeast dataset

The Yeast dataset (Nakai, 1991) contains data on protein localization sites in the yeast Saccharomyces cerevisiae. Each observation corresponds to a protein, and the goal is to predict the subcellular location where the protein functions, based on its sequence-derived features. We consider a subset of this dataset.

Specifically, we aim to distinguish the two localization sites, CYT (cytosolic or cytoskeletal) and ME3 (membrane protein, no N-terminal signal), by considering three variables: McGeoch's method for signal sequence recognition (mcg), the score of the ALOM membrane spanning region prediction program (alm), and the score of discriminant analysis of the amino acid content of vacuolar and extracellular proteins (vac). This subset of the dataset contains a total of 626 observations, with 463 labeled as CYT and 163 as ME3. Figure 5.1 shows the pairwise scatterplots of the selected variables, with points colored by cluster assignment, illustrating the data partition.

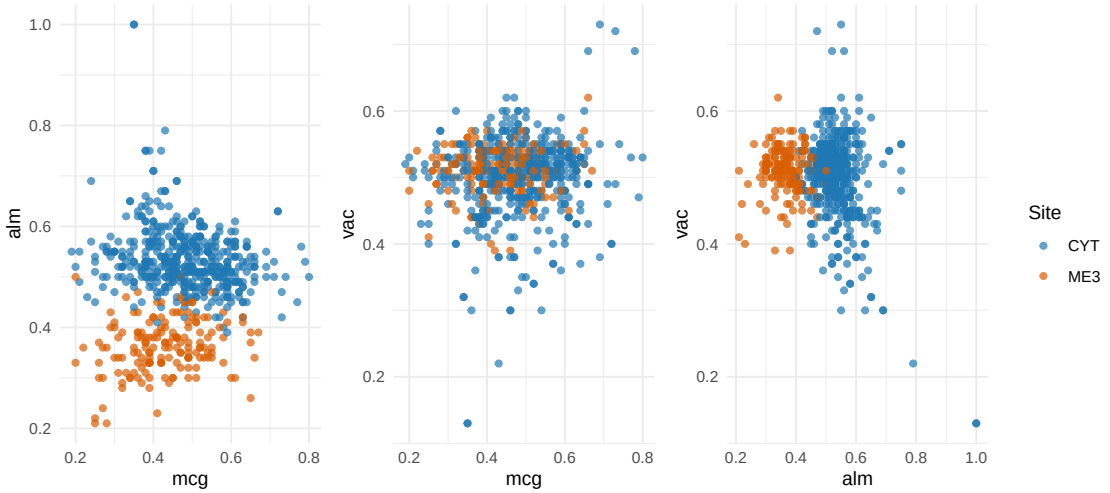


Figure 5.1: Pairwise scatterplots of selected features (mcg, alm, vac) from the Yeast dataset, showing two classes of protein localization sites: CYT (cytosolic or cytoskeletal) and ME3 (membrane protein, no N-terminal signal). Each point represents a protein, colored according to its class.

The following hyperparameters are used for the estimations of the DPGMM with a normal-Wishart prior. The prior mean $\mathbf{m}_0$ of the Gaussian components is set to the empirical mean of the observed data. The scalar precision parameter β_0 that controls the strength of the prior on the component means is set to 10. The scale matrix W_0^{-1} for the prior on the covariance matrices of the Gaussian components is set to the sample covariance of the data. The degrees of freedom ν_0 for the Wishart prior is set to 5, which ensures that the corresponding inverse-Wishart distribution has a finite expectation. The concentration parameter α of the Dirichlet process is set to 1, implying a prior expectation of 6.44 clusters.

The CAVI algorithm is run multiple times with different initializations to identify the starting values that yield the highest value of the ELBO. Figure 5.2 shows the progression of the ELBO across iterations for the best run.

The MCMC sampling is run for 10 000 iterations after a burn-in of 10 000 iterations.

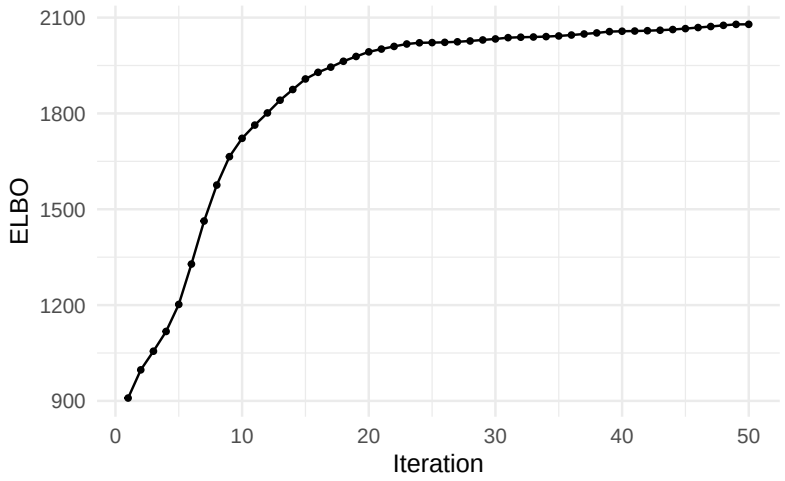


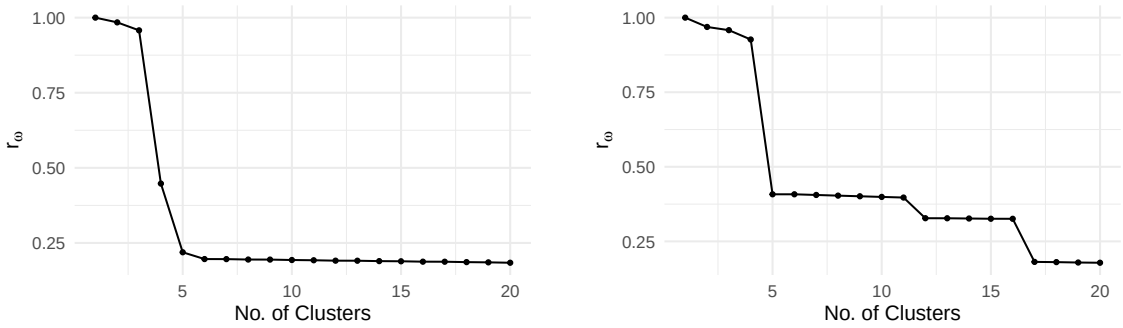
Figure 5.2: Progression of the ELBO during the estimation of the DPGMM with a normal-Wishart prior on the Yeast dataset.

FOLD is applied to post-process the estimates obtained from both VB and MCMC, resulting in the elbow plots shown in Figure 5.3.

Table 5.1 presents the number of clusters identified by the raw VB and MCMC estimates, along with those obtained after post-processing with FOLD. Each clustering result is evaluated with the ARI.

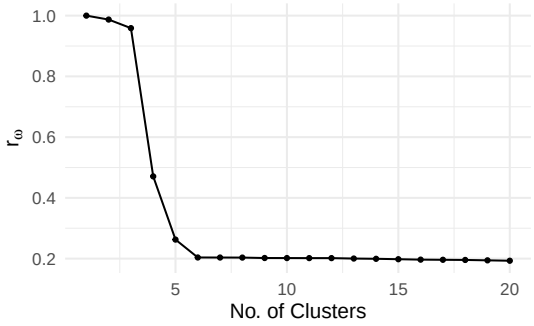
The raw VB estimate overestimates the number of clusters, identifying 6 clusters with a relatively low ARI of 0.362. Post-processing this estimate using FOLD with both the Hellinger distance and the Wasserstein metric reduces the number of clusters to 5. When using the Hellinger distance, the improvement in ARI is marginal (0.373 and 0.389, respectively, for the standard and plug-in implementations). In contrast, employing the Wasserstein metric substantially enhances the partitioning, achieving an ARI of 0.715. Notably, the highest performance is obtained by combining the plug-in approximation with the Wasserstein metric, which reduces the number of clusters to 3 while achieving an ARI of 0.726.

Similarly, the raw MCMC estimate tends to overestimate the number of clusters, identifying 5 and achieving an ARI of 0.441. While this result is slightly better than the raw VB estimate, it still reflects a suboptimal partitioning of the data. Applying FOLD post-processing improves clustering quality in all cases. Specifically, using the Hellinger distance reduces the number of clusters to 4 and raises the ARI to 0.593. Employing the Wasserstein metric maintains the original number of clusters but achieves a higher ARI of 0.653, indicating a better alignment with the true data partition.

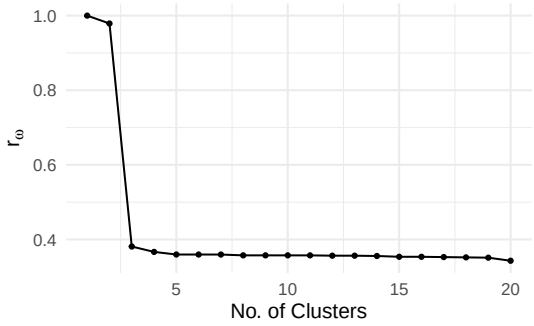


(a) VB + FOLD-H

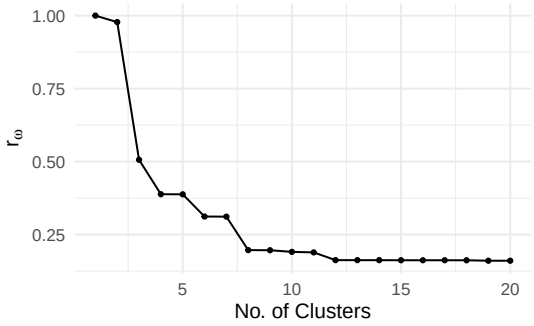
(b) VB + FOLD-W



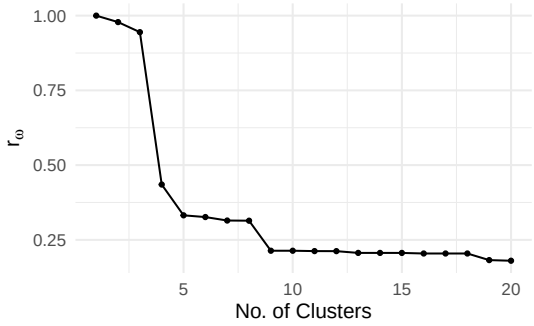
(c) VB + FOLD-H (Plug-in)



(d) VB + FOLD-W (Plug-in)



(e) MCMC + FOLD-H



(f) MCMC + FOLD-W

Figure 5.3: Elbow plots used to select the number of clusters for each combination of estimation method and FOLD implementation in the DPGMM with a normal-Wishart prior, applied to the Yeast dataset.

Despite improvements in number of clusters and ARI, none of the methods successfully recovers the true number of clusters.

5.2 Flea beetles dataset

The Flee beetles dataset (Lubischew, 1962) consists of six physical measurements of 74 flea beetles from three different species: concinna, heptapotamica, and heikertingeri. Both the original and the standardized datasets are available in the

Method	No. of Clusters	ARI
VB (Pre)	6	0.362
VB + FOLD-H	5	0.373
VB + FOLD-W	5	0.715
VB + FOLD-H (Plug-in)	5	0.389
VB + FOLD-W (Plug-in)	3	0.726
MCMC (Pre)	5	0.441
MCMC + FOLD-H	4	0.593
MCMC + FOLD-W	5	0.653

Table 5.1: Comparison of the number of clusters and adjusted Rand index for each combination of estimation method and FOLD implementation in the DPGMM with a normal-Wishart prior, applied to the Yeast dataset.

R package `tourr` (Wickham et al., 2011). The whole standardized dataset is used in the analysis.

The following hyperparameters are used for the estimations of the DPGMM with a normal-Wishart prior. The prior mean $\mathbf{m}_0$ of the Gaussian components is set to the origin. The scalar precision parameter β_0 that controls the strength of the prior on the component means is set to 1. The scale matrix W_0^{-1} for the prior on the covariance matrices of the Gaussian components is set to the identity matrix. The degrees of freedom ν_0 for the Wishart prior is set to 8, which ensures that the corresponding inverse-Wishart distribution has a finite expectation. The concentration parameter α of the Dirichlet process is set to 1, implying a prior expectation of 4.31 clusters.

The CAVI algorithm is run multiple times with different initializations to identify the starting values that yield the highest value of the ELBO. Figure 5.4 shows the progression of the ELBO across iterations for the best run.

The MCMC sampling is run for 3 000 iterations after a burn-in of 3 000 iterations.

FOLD is applied to post-process the estimates obtained from both VB and MCMC, resulting in the elbow plots shown in Figure 5.5.

Table 5.2 presents the number of clusters identified by the raw VB and MCMC estimates, along with those obtained after post-processing with FOLD. Each clustering result is evaluated with the ARI.

The raw VB estimate oversegments the data, identifying 8 clusters, yet it achieves a relatively high-quality partition with an ARI of 0.803. All FOLD implementations successfully recover the true number of clusters (3). Among them, the plug-in approximation with the Hellinger distance yields an ARI of 0.875, while all other FOLD-based methods perfectly match the ground truth with an ARI of 1.

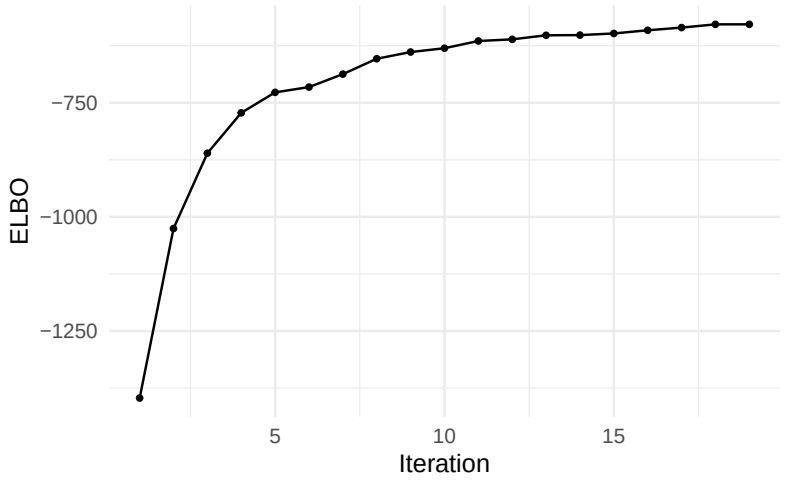


Figure 5.4: Progression of the ELBO during the estimation of the DPGMM with a normal-Wishart prior on the Flea beetles dataset.

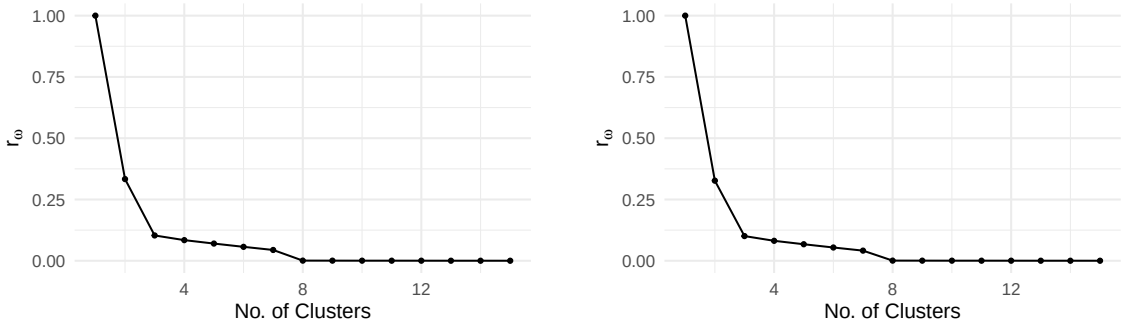
In contrast, the raw MCMC estimate already recovers the correct clustering structure, identifying exactly 3 clusters with an ARI of 1. Applying FOLD in this case does not alter the results, preserving both the number of clusters and the perfect clustering accuracy.

Method	No. of Clusters	ARI
VB (Pre)	8	0.803
VB + FOLD-H	3	1
VB + FOLD-W	3	1
VB + FOLD-H (Plug-in)	3	0.875
VB + FOLD-W (Plug-in)	3	1
MCMC (Pre)	3	1
MCMC + FOLD-H	3	1
MCMC + FOLD-W	3	1

Table 5.2: Comparison of the number of clusters and adjusted Rand index for each combination of estimation method and FOLD implementation in the DPGMM with a normal-Wishart prior, applied to the Flea beetles dataset.

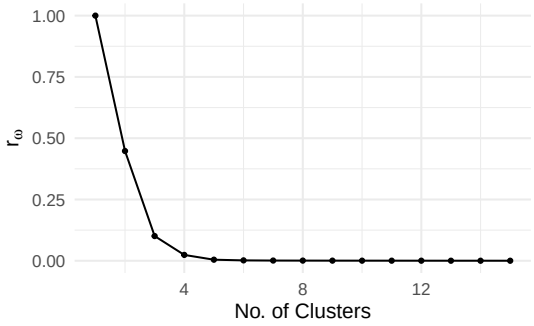
5.3 Australian Institute of Sports dataset

The Australian Institute of Sports dataset (Cook & Weisberg, 1994) contains physical measurements and blood measurements from 202 high performance athletes. The sex and the practiced sport of each athlete are known. This dataset is available in the `R` package `locfit` (Loader, 2025). We consider a subset of the dataset. Specifically, we aim to distinguish the sex of the athlete, by considering three physical measurements: the body mass index (BMI) in kilograms, the lean

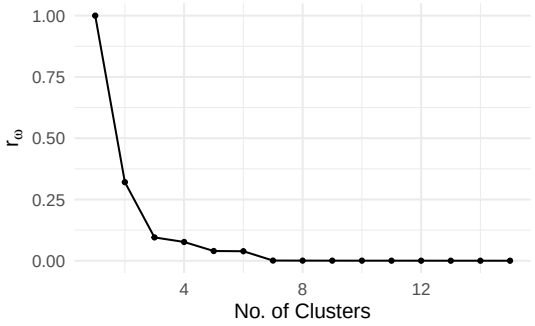


(a) VB + FOLD-H

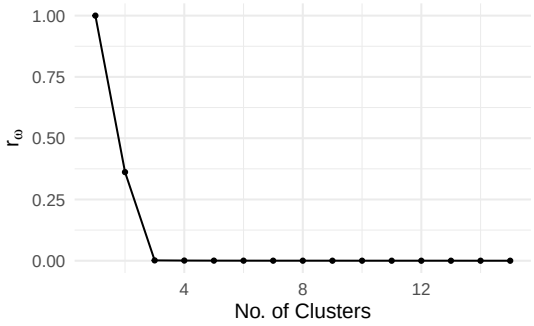
(b) VB + FOLD-W



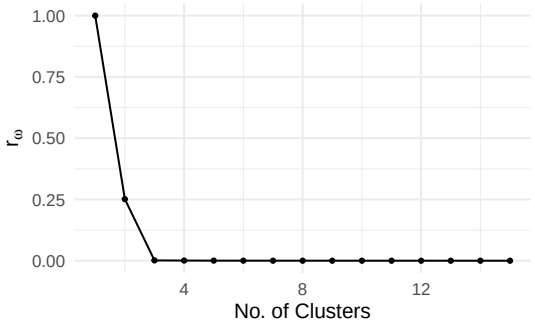
(c) VB + FOLD-H (Plug-in)



(d) VB + FOLD-W (Plug-in)



(e) MCMC + FOLD-H



(f) MCMC + FOLD-W

Figure 5.5: Elbow plots used to select the number of clusters for each combination of estimation method and FOLD implementation in the DPGMM with a normal-Wishart prior, applied to the Flea beetles dataset.

body mass (LBM) in kilograms, and the percentage of body fat (BFat). Figure 5.6 shows the pairwise scatterplots of the selected variables, with points colored by cluster assignment, illustrating the data partition.

The following hyperparameters are used for the estimations of the DPGMM with a normal-Wishart prior. The prior mean $\mathbf{m}_0$ of the Gaussian components is set to the origin. The scalar precision parameter β_0 is set to 1. The scale matrix W_0^{-1} for the prior on the covariance matrices of the Gaussian components is set to the identity matrix. The degrees of freedom ν_0 for the Wishart prior is set to

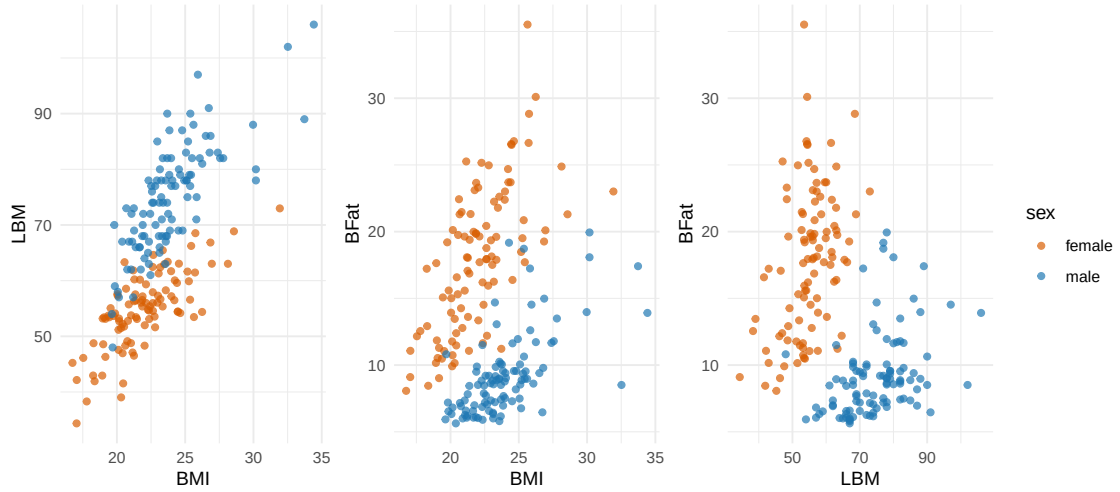


Figure 5.6: Pairwise scatterplots of selected features (BMI, LBM, and BFat) from the Australian Institute of Sports dataset. Each point represents an athlete, colored according to their sex.

5. The concentration parameter α of the Dirichlet process is set to 1, implying a prior expectation of 5.13 clusters.

The CAVI algorithm is run multiple times with different initializations to identify the starting values that yield the highest value of the ELBO. Figure 5.2 shows the progression of the ELBO across iterations for the best run.

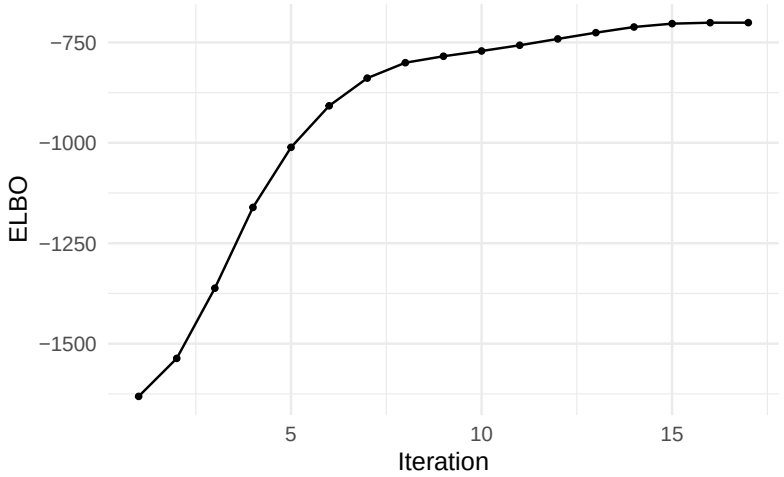
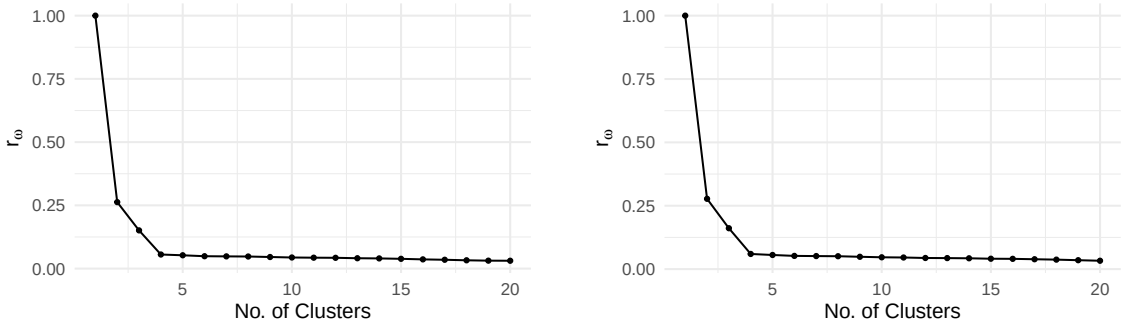


Figure 5.7: Progression of the ELBO during the estimation of the DPGMM with a normal-Wishart prior on the Australian Institute of Sports dataset.

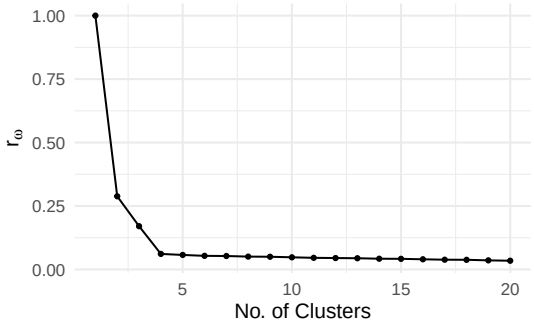
The MCMC sampling is run for 4 000 iterations after a burn-in of 4 000 iterations.

FOLD is applied to post-process the estimates obtained from both VB and MCMC, resulting in the elbow plots shown in Figure 5.8.

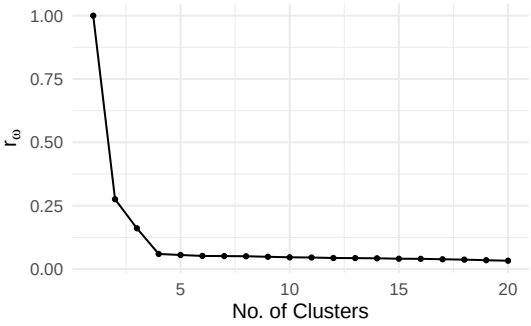


(a) VB + FOLD-H

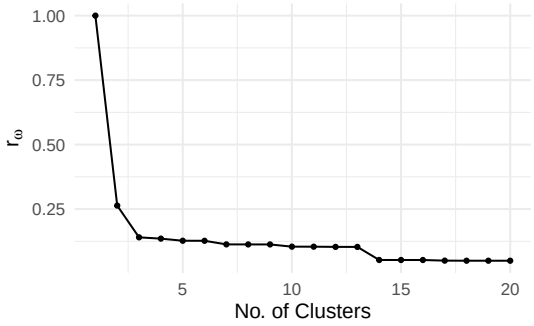
(b) VB + FOLD-W



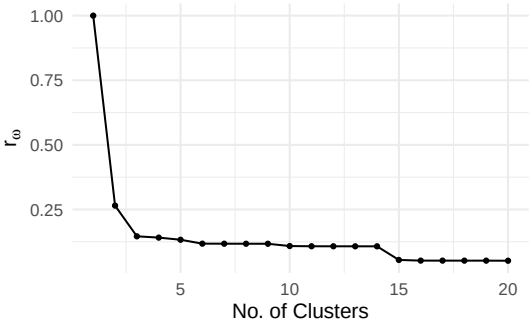
(c) VB + FOLD-H (Plug-in)



(d) VB + FOLD-W (Plug-in)



(e) MCMC + FOLD-H



(f) MCMC + FOLD-W

Figure 5.8: Elbow plots used to select the number of clusters for each combination of estimation method and FOLD implementation in the DPGMM with a normal-Wishart prior, applied to the Australian Institute of Sports dataset.

Table 5.3 presents the number of clusters identified by the raw VB and MCMC estimates, along with those obtained after post-processing with FOLD. Each clustering result is evaluated with the ARI.

Both the raw VB and MCMC estimates identify 4 clusters and their clusterings attain ARI values of 0.602 and 0.760, respectively. Each FOLD method recovers the true number of clusters (2). However, using the Wasserstein metric yields to higher quality clusterings than using the Hellinger distance.

Method	No. of Clusters	ARI
VB (Pre)	4	0.602
VB + FOLD-H	2	0.724
VB + FOLD-W	2	0.829
VB + FOLD-H (Plug-in)	2	0.707
VB + FOLD-W (Plug-in)	2	0.829
MCMC (Pre)	4	0.760
MCMC + FOLD-H	2	0.690
MCMC + FOLD-W	2	0.829

Table 5.3: Comparison of the number of clusters and adjusted Rand index for each combination of estimation method and FOLD implementation in the DPGMM with a normal-Wishart prior, applied to the Australian Institute of Sports dataset.

5.4 Wisconsin Diagnostic Breast Cancer dataset

The Wisconsin Diagnostic Breast Cancer dataset (Wolberg et al., 1993) contains data obtained from digitized images of fine needle aspirates of breast masses. Each observation corresponds to a tumor mass, which is categorized either as benign or malignant. We consider a subset of this dataset. Specifically, we aim to distinguish the diagnosis, B (benign) and M (malignant), by considering three variables: the mean texture, the "worst" area, and the "worst" smoothness. Figure 5.9 shows the pairwise scatterplots of the selected variables, with points colored by cluster assignment, illustrating the data partition.

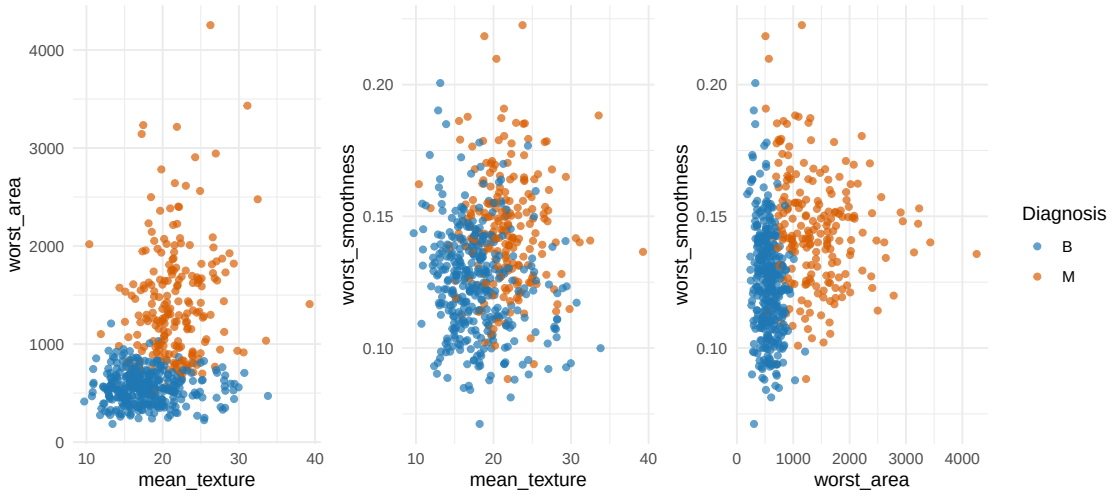


Figure 5.9: Pairwise scatterplots of selected features (mean_texture, worst_area, and worst_smoothness) from the Wisconsin Diagnostic Breast Cancer dataset. Each point represents a tumor sample, colored according to its diagnostic category (benign or malignant).

The DPGMM with a normal-Wishart prior is estimated on the standardized data. The following hyperparameters are used for the estimations. The prior mean $\mathbf{m}_0$ of the Gaussian components is set to the origin. The scalar precision parameter β_0 is set to 1. The scale matrix W_0^{-1} for the prior on the covariance matrices of the Gaussian components is set to the identity matrix. The degrees of freedom ν_0 for the Wishart prior is set to 5. The concentration parameter α of the Dirichlet process is set to 1, implying a prior expectation of 5.13 clusters.

The CAVI algorithm is run multiple times with different initializations to identify the starting values that yield the highest value of the ELBO. Figure 5.10 shows the progression of the ELBO across iterations for the best run.

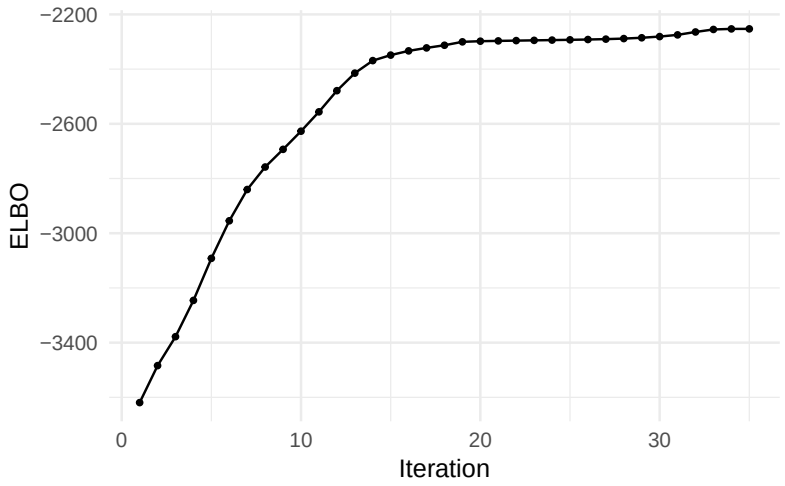


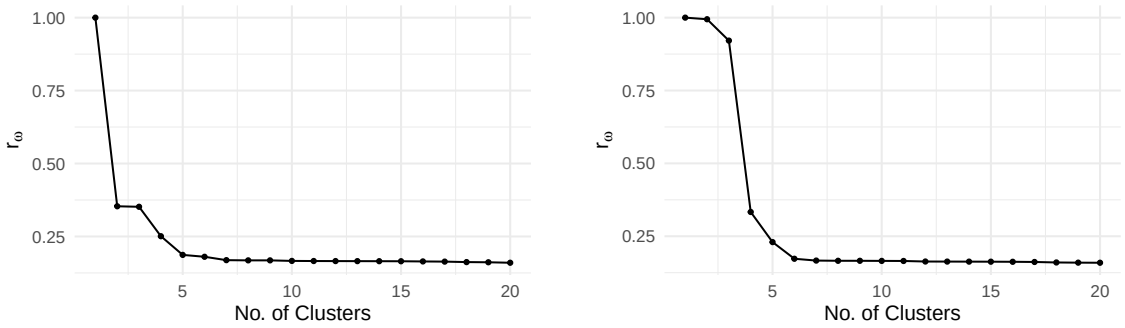
Figure 5.10: Progression of the ELBO during the estimation of the DPGMM with a normal-Wishart prior on the Wisconsin Diagnostic Breast Cancer dataset.

The MCMC sampling is run for 5 000 iterations after a burn-in of 5 000 iterations.

Finally, FOLD is applied to post-process the estimates obtained from both VB and MCMC, resulting in the elbow plots shown in Figure 5.11.

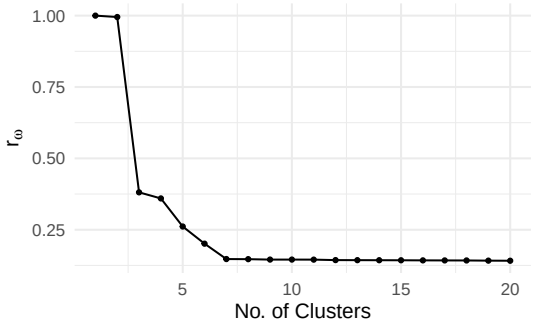
Table 5.4 presents the number of clusters identified by the raw VB and MCMC estimates, along with those obtained after post-processing with FOLD. Each clustering result is evaluated with the ARI.

The raw VB estimate detects 7 clusters, considerably oversegmenting the data and yielding a moderate ARI of 0.534. Post-processing with FOLD significantly improves performance. Using the Hellinger distance reduces the number of clusters to 2 and raises the ARI to 0.837. The plug-in version of the Hellinger-based post-processing further improves the clustering quality, achieving the highest ARI of 0.842 with 3 clusters. The Wasserstein-based methods (the original version and its plug-in counterpart) also improve the clustering compared to the raw estimate, identifying 4 clusters with ARI values of 0.805 and 0.812, respectively.

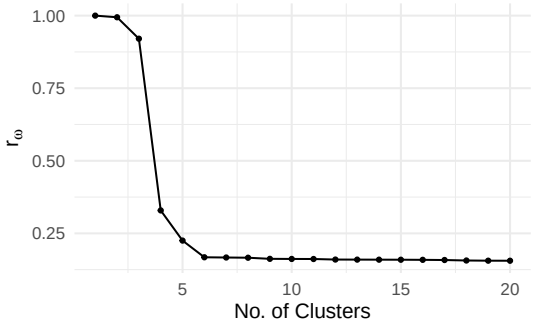


(a) VB + FOLD-H

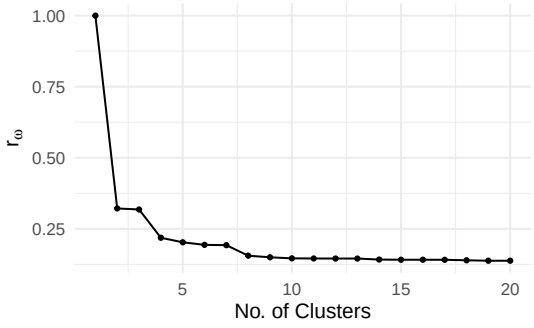
(b) VB + FOLD-W



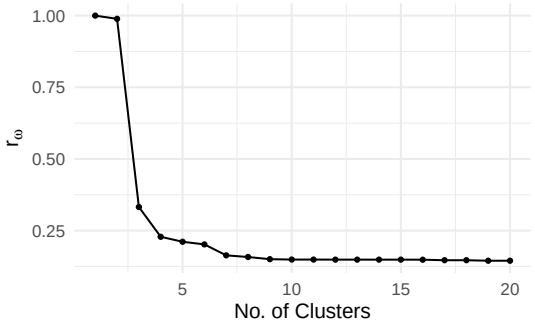
(c) VB + FOLD-H (Plug-in)



(d) VB + FOLD-W (Plug-in)



(e) MCMC + FOLD-H



(f) MCMC + FOLD-W

Figure 5.11: Elbow plots used to select the number of clusters for each combination of estimation method and FOLD implementation in the DPGMM with a normal-Wishart prior, applied to the Wisconsin Diagnostic Breast Cancer dataset.

For the MCMC approach, the raw estimate yields 4 clusters and an ARI of 0.738. Post-processing with FOLD again leads to improved clustering performance. Using the Hellinger distance, the number of clusters is reduced to 2, with a resulting ARI of 0.824. The Wasserstein-based post-processing identifies 3 clusters and achieves an ARI of 0.813. These results confirm that FOLD, particularly in combination with plug-in approximations and the Hellinger distance, effectively improves both the clustering accuracy and alignment with the ground truth.

Method	No. of Clusters	ARI
VB (Pre)	7	0.534
VB + FOLD-H	2	0.837
VB + FOLD-W	4	0.805
VB + FOLD-H (Plug-in)	3	0.842
VB + FOLD-W (Plug-in)	4	0.812
MCMC (Pre)	4	0.738
MCMC + FOLD-H	2	0.824
MCMC + FOLD-W	3	0.813

Table 5.4: Comparison of the number of clusters and adjusted Rand index for each combination of estimation method and FOLD implementation in the DPGMM with a normal-Wishart prior, applied to the Wisconsin Diagnostic Breast Cancer dataset.

Chapter 6

Conclusions

In this manuscript, we presented the Bayesian nonparametric framework with a particular focus on the Dirichlet process. We discussed its theoretical properties and demonstrated its role in constructing flexible mixture models. Two major inference strategies were examined in depth: Markov chain Monte Carlo methods and variational inference. We presented the foundational theory behind MCMC, along with the widely used Neal's Algorithm 8, and then transitioned to the variational framework, where we derived the coordinate ascent variational inference algorithm.

We then introduced the Fusing of Localized Densities procedure, originally proposed as a post-processing method for MCMC outputs. Our contribution extended this method to the variational inference setting. In particular, we proposed two implementations of FOLD that differ in how they estimate expected distances between localized densities: a plug-in approximation and a Monte Carlo approximation. Both approaches offer substantial improvements in runtime compared to the original MCMC-based FOLD procedure.

These general techniques were applied to the Dirichlet process Gaussian mixture model with a normal-Wishart prior, one of the most widely used Bayesian nonparametric models for continuous data. With proper modifications, however, they can be applied to any Bayesian mixture model.

We conducted simulation studies and real data applications to compare clustering results from MCMC and VI estimates under well-specified and misspecified models. We used the FOLD procedure, employing both the Hellinger distance and the Wasserstein metric, to post-process these results and evaluated the resulting partitions. Notably, VI provided clustering results comparable to those obtained using MCMC, suggesting that the known drawback of underestimating posterior variance may be acceptable in exchange for substantially faster inference in certain scenarios. Interestingly, the use of the Wasserstein metric in the FOLD procedure

often led to fewer clusters being identified, providing the advantage of more interpretable clusterings.

There are several avenues for future research. It would be interesting to further investigate the behavior of MCMC and VI estimates, both before and after FOLD post-processing, in more complex settings. This includes exploring different sample sizes and higher-dimensional data. A particular focus should be placed on the trade-off between computational efficiency and inference accuracy. Additionally, the sensitivity of FOLD to different statistical distances could be examined, paying special attention to the tendency of the Wasserstein metric to yield more parsimonious clusterings. Finally, the CAVI algorithm could be extended to accommodate models with more relaxed specifications, and the previous analyses could be replicated under these alternative model choices. Specifically, hyperpriors over key model hyperparameters could be incorporated. Notably, placing a hyperprior over the Dirichlet process concentration parameter (Escobar & West, 1995) would provide a more data-driven determination of the number of clusters, reducing the reliance on subjective specifications. These directions are left for future work.

Bibliography

Antoniak, C. E. (1974). Mixtures of Dirichlet Processes with Applications to Bayesian Nonparametric Problems. *The Annals of Statistics* **2**, 1152 – 1174.

Binder, D. A. (1978). Bayesian cluster analysis. *Biometrika* **65**, 31–38.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag.

Blei, D. M. & Jordan, M. I. (2006). Variational inference for Dirichlet process mixtures. *Bayesian Analysis* **1**, 121 – 143.

Blei, D. M., Kucukelbir, A. & McAuliffe, J. D. (2017). Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association* **112**, 859–877.

Cook, R. D. & Weisberg, S. (1994). Transforming a response variable for linearity. *Biometrika* **81**, 731–737.

Corradin, R., Canale, A. & Nipoti, B. (2021). BNPmix: An R package for Bayesian nonparametric modeling via Pitman-Yor mixtures. *Journal of Statistical Software* **100**, 1–33.

Dahl, D. B., Johnson, D. J. & Mueller, P. (2021). Search Algorithms and Loss Functions for Bayesian Clustering.

Dombowsky, A. & Dunson, D. B. (in press). Bayesian clustering via fusing of localized densities. *Journal of the American Statistical Association* .

Escobar, M. D. & West, M. (1995). Bayesian Density Estimation and Inference Using Mixtures. *Journal of the American Statistical Association* **90**, 577–588.

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

Ghosal, S. & van der Vaart, A. (2017). *Fundamentals of Nonparametric Bayesian Inference*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Hubert, L. & Arabie, P. (1985). Comparing partitions. *Journal of Classification* **2**, 193–218.

Ishwaran, H. & Zarepour, M. (2002). Exact and Approximate Sum Representations for the Dirichlet Process. *The Canadian Journal of Statistics / La Revue Canadienne de Statistique* **30**, 269–283.

Loader, C. (2025). *locfit: Local Regression, Likelihood and Density Estimation*. R package version 1.5-9.12.

Lubischew, A. A. (1962). On the Use of Discriminant Functions in Taxonomy. *Biometrics* **18**, 455–477.

Meilă, M. (2007). Comparing clusterings—an information based distance. *Journal of Multivariate Analysis* **98**, 873–895.

Nakai, K. (1991). Yeast. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5KG68.

Neal, R. M. (2000). Markov Chain Sampling Methods for Dirichlet Process Mixture Models. *Journal of Computational and Graphical Statistics* **9**, 249–265.

Peyré, G. & Cuturi, M. (2019). Computational optimal transport. *Foundations and Trends in Machine Learning* **11**, 355–607.

Robert, C. P. & Casella, G. (2004). *Monte Carlo Statistical Methods*. Springer Verlag.

Sethuraman, J. (1994). A constructive definition of Dirichlet priors. *Statistica Sinica* **4**, 639–650.

Wade, S. & Ghahramani, Z. (2018). Bayesian Cluster Analysis: Point Estimation and Credible Balls (with Discussion). *Bayesian Analysis* **13**, 559 – 626.

Wickham, H., Cook, D., Hofmann, H. & Buja, A. (2011). tourr: An R package for exploring multivariate data with projections. *Journal of Statistical Software* **40**, 1–18.

Wolberg, W., Mangasarian, O., Street, N. & Street, W. (1993). Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5DW2B.

Appendix A

Implementation of the CAVI algorithm

Listing A.1: Implementation of the CAVI algorithm for Dirichlet process Gaussian mixture models with normal-Wishart priors.

```
cavi_dpgmm_nw <- function(data, # Data matrix
                          m_0, # Prior mean of the Gaussian components
                          beta_0, # Strength of the prior on the mean
                          W_0, # Scale matrix of the prior on the precision matrix
                          nu_0, # Degrees of freedom for the prior on the precision matrix
                          alpha_0, # Concentration parameter
                          H = 100, # Maximum number of mixture components
                          max_iter = 50, # Maximum number of iterations
                          threshold = 1e-3, # Maximum relative elbo decrement
                          print_message = TRUE) {
  t_0 <- Sys.time()
  n <- nrow(data)
  p <- ncol(data)

  # Initialization
  convergence <- FALSE
  iter <- 0
  elbo <- numeric(max_iter)

  r <- bmixture::rdirichlet(n, rep(alpha_0 / H, H))
  W_0_inv <- qr.solve(qr(W_0))
  log_det_W_0 <- determinant(W_0)$modulus[1]

  while (!convergence) {
    iter <- iter + 1
    if (print_message) cat(sprintf("\rIteration %3d", iter))

    # Compute statistics on observed data
    N <- colSums(r)
    means <- matrix(0, nrow = H, ncol = p)
    S <- array(0, dim = c(p, p, H))
    for (h in which(N != 0)) {
      means[h, ] <- crossprod(r[, h], data) / N[h]
      centered_data <- sweep(data, 2, means[h, ], "-")
      S[, , h] <- crossprod(centered_data, (centered_data * r[, h])) / N[h]
    }

```

```r
    # Update hyperparameters of the Dirichlet distribution
    alpha <- alpha_0 / H + N

    # Update strength parameters
    beta <- beta_0 + N

    # Update means
    m <- (beta_0 * matrix(m_0, nrow = H, ncol = p, byrow = TRUE) + N * means) / beta

    # Update scale matrices
    W_inv <- sapply(1:H, function(h) {
      means_h <- matrix(means[h, ])
      W_0_inv + N[h] * S[, , h] + beta_0 * N[h] / beta[h] * tcrossprod(means_h - m_0)
    }, simplify = FALSE)
    W_inv <- abind::abind(W_inv, along = 3)

    W <- apply(W_inv, 3, function(x) qr.solve(qr(x)), simplify = FALSE)
    W <- abind::abind(W, along = 3)

    # Update degrees of freedom
    nu <- nu_0 + N

    # Compute useful expected values
    alpha_plus <- sum(alpha)
    E_log_pi <- digamma(alpha) - digamma(alpha_plus)
    log_det_W <- apply(W, 3, function(x) determinant(x)$modulus[1])
    E_log_det_Lambda <- CholWishart::mvdigamma(nu / 2, p) + p * log(2) + log_det_W
    Nu <- matrix(nu, nrow = n, ncol = H, byrow = TRUE)
    quad_form_x <- sapply(1:H, function(h) {
      centered_data <- sweep(data, 2, m[h, ], "-")
      rowSums(tcrossprod(centered_data, W[, , h]) * centered_data)
    })
    E_quad_form_lik <- matrix(p / beta, nrow = n, ncol = H, byrow = TRUE) + Nu * quad_form_x

    # Update responsibilities
    log_rho <- tcrossprod(matrix(rep(1, n)), E_log_pi + 0.5 * E_log_det_Lambda) - 0.5 * E_quad_form_lik
    max_log_rho <- apply(log_rho, 1, max)
    r <- exp((log_rho - max_log_rho) - log(rowSums(exp(log_rho - max_log_rho))))

    # Compute ELBO
    quad_form_m <- vapply(1:H, function(h) {
      diff_vec <- m[h, ] - m_0
      weighted_diff <- crossprod(W[, , h], diff_vec)
      as.numeric(crossprod(diff_vec, weighted_diff))
    }, FUN.VALUE = numeric(1))
    trace_W_0_inv_W <- apply(W, 3, function(x) sum(W_0_inv * x))

    elbo[iter] <- c(
      sum(lgamma(alpha)) - lgamma(alpha_plus)
      - H * lgamma(alpha_0 / H) + lgamma(alpha_0)
      + 0.5 * p * H * log(beta_0) - 0.5 * p * sum(log(beta))
      - sum(r * log(r + .Machine$double.eps))
      - 0.5 * beta_0 * sum(nu * quad_form_m)
      - 0.5 * sum(rowSums(r * Nu * quad_form_x))
      - 0.5 * sum(nu * trace_W_0_inv_W)
      + 0.5 * sum(nu * log_det_W) - 0.5 * nu_0 * H * log_det_W_0 +
```

```
        sum(CholWishart::lmvgamma(nu / 2, p)) - H * CholWishart::lmvgamma(nu_0 / 2, p)
      + 0.5 * p * sum(nu) - 0.5 * p * n * log(pi)
    )

    # Convergence check
    rel_growth <- ifelse(iter > 1, (elbo[iter] - elbo[iter - 1]) / abs(elbo[iter - 1]), threshold + 1)
    convergence <- iter == max_iter | rel_growth <= threshold
  }
  t_1 <- Sys.time()
  tot_time <- as.numeric(difftime(t_1, t_0, units = "secs"))

  if (print_message) {
    cat(sprintf("\rConvergence reached at iteration %d in %.3f seconds.\nELBO: %.3f\n",
                iter, tot_time, elbo[iter]))
  }

  # Output
  list(elbo = elbo[1:iter], tot_time = tot_time,
    z = apply(r, 1, which.max),
    m = m,
    beta = beta,
    W = W,
    W_inv = W_inv,
    nu = nu,
    r = r
  )
}
```