université PARIS-SACLAY

Scalable Bayesian Non-Parametric Models for Clustering and Co-Clustering

Modèles Bayésiens non-paramétriques scalables pour le clustering et co-clustering

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique
Graduate School : Informatique et sciences du numérique
Référent: Université de Versailles-Saint-Quentin-en-Yvelines

Thèse préparée dans l'unité de recherche **DAVID (Université Paris-Saclay, UVSQ)**, sous la direction de **Mustapha LEBBAH**, Professeur des universités, et le co-encadrement de **Hanene AZZAG**, Professeure des universités à l'Université Sorbonne Paris Nord.

Thèse soutenue à Versailles, le 07 Novembre 2025, par

Reda KHOUFACHE

Composition du jury

Membres du jury avec voix délibérative

Tabea REBAFKA Professeure des universités, AgroParisTech, Université Paris Saclay	Présidente
Christophe BIERNACKI Professeur des universités, Inria, Université de Lille	Rapporteur & Examinateur
Guillaume CLEUZIOU Professeur des universités, Université d'Orléans	Rapporteur & Examinateur
Lazhar LABIOD Maître de conférences, Université Paris-Cité	Examinateur
Yuichi YOSHIDA Professeur, National Institute of Informatics (Japon)	Examinateur

THESE DE DOCTORAT

NNT : 2025UPASG069

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Modèles Bayésiens non-paramétriques scalables pour le clustering et co-clustering
Mots clés: Modèles de mélange gaussien; Modèles de mélange de processus de Dirichlet; Modèles à blocs latents non paramétriques; Calcul distribué; Algorithme EM; Sensibilité moyenne.

Résumé: Le *clustering* est une tâche fondamentale de l'apprentissage non supervisé, visant à regrouper des observations similaires en *clusters*. Les modèles de mélange gaussien (*GMM*) figurent parmi les approches probabilistes les plus utilisées en vertu de leur interprétabilité et de leur efficacité. Toutefois, ils nécessitent de spécifier à l'avance le nombre de *clusters* K, ce qui limite leur applicabilité en pratique. Les modèles de mélange à processus de Dirichlet représentent une extension bayésienne non paramétrique des GMM, permettant d'inférer automatiquement K à partir des données et offrant ainsi plus de flexibilité.

Lorsque les jeux de données tabulaires présentent une structure duale entre les observations (lignes) et les variables (colonnes), le *co-clustering*—qui partitionne simultanément les lignes et les colonnes en blocs homogènes—surpasse souvent les approches classiques de *clustering*, qui ne produisent qu'une partition des lignes. Le modèle de blocs latents non paramétrique étend les modèles classiques de mélange en blocs dans un cadre bayésien non paramétrique, et permet d'inférer automatiquement le nombre de *clusters* de lignes et de colonnes.

Bien que les méthodes de Monte Carlo par chaînes de Markov (*MCMC*), telles que l'échantillonneur de Gibbs, jouissent de bonnes propriétés asymptotiques et d'une grande précision, leur processus d'inférence devient très lent en raison de leur complexité computationnelle, ce qui limite leur applicabilité lorsque le nombre d'observations est grand. Pour relever ce défi, nous proposons DisCGS, un algorithme d'inférence distribué qui approxime l'échantillonneur de Gibbs. Conçu pour des données partitionnées horizontalement sur plusieurs *workers*, DisCGS permet un *clustering* efficace et scalable, aussi bien pour des données continues que discrètes.

Dans le cas continu, notre implémentation, lorsque les composantes sont supposées gaussiennes, atteint un temps d'exécution d'environ 3 minutes pour 100 itérations sur un jeu de données de 100 000 points, soit une accélération de plus de 200 fois par rapport à une approche centralisée qui nécessite 12 heures. Dans le cas discret, nous étendons notre méthode au cas où les composantes sont multinomiales, avec une application au *clustering* de textes. Enfin, notre méthode se généralise aisément aux distributions de la famille exponentielle.

Nous étendons également cette approche distribuée au *co-clustering* via DisNPLBM, un algorithme d'inférence scalable pour les modèles de blocs latents bayésiens non paramétriques. DisNPLBM adopte une architecture *master/worker*, où les données sont réparties par lignes, permettant une inférence parallèle sans communication entre les *workers*. Il capture efficacement les structures de *co-clustering*. Nous validons le passage à l'échelle et la précision de notre approche sur des données synthétiques, et présentons également une application pratique sur des données réelles d'expression de gènes.

Au-delà de la scalabilité, la fiabilité des algorithmes de clustering est essentielle pour garantir des résultats robustes. Nous étudions la stabilité de l'algorithme *Expectation-Maximization* (*EM*) appliqué aux *GMM* via la notion de sensibilité moyenne, qui mesure la variation des paramètres inférés lors de petites perturbations des données, comme la suppression d'un point. Nous démontrons théoriquement que *EM* est stable lorsque les proportions de mélange sont fixées et égales, et étendons ce résultat au cas plus général de proportions inconnues. Nos analyses supposent des clusters sphériques avec une covariance diagonale connue, sans hypothèse sur la séparabilité des clusters ni sur l'initialisation. Des expériences approfondies sur données synthétiques et réelles confirment nos résultats théoriques, soulignant la robustesse pratique de l'algorithme *EM*.

Title: Scalable Bayesian non-parametric models for clustering and co-clustering

Keywords: Gaussian mixture models; Dirichlet process mixture models; Bayesian non-parametric latent block models; Distributed computing; EM algorithm; Average sensitivity.

Clustering is a fundamental task in unsupervised learning, aiming to group similar observations into meaningful clusters. Gaussian Mixture Models (GMMs) are among the most popular probabilistic approaches for clustering, providing a flexible parametric framework. However, they assume a fixed and predefined number of clusters, which limits their applicability in practice. Dirichlet Process Mixture Models (DPMMs), a Bayesian non-parametric extension of GMMs, overcome this limitation by allowing the number of clusters to be inferred automatically from the data.

When a dataset exhibits a dual structure between observations and features, co-clustering—which simultaneously partitions both rows and columns into homogeneous blocks—outperforms traditional clustering methods that partition only the rows. The Non-Parametric Latent Block Model (NPLBM) extends parametric block mixture models to the Bayesian non-parametric setting, allowing the automatic estimation of the number of row and column clusters.

While inference using Markov Chain Monte Carlo (MCMC) methods, such as collapsed Gibbs sampling, provides strong asymptotic guarantees and high precision, it becomes computationally prohibitive for large datasets. To address this limitation, we introduce DisCGS, a distributed inference algorithm that approximates the collapsed Gibbs sampler using sufficient statistics. Designed for horizontally partitioned data across multiple workers, DisCGS enables efficient and scalable clustering for both continuous and discrete data.

For continuous data, our implementation with Gaussian components achieves a runtime of approximately 3 minutes for 100 iterations on a dataset with 100,000 points, yielding over a 200× speedup compared to a centralized sampler that requires 12 hours. For discrete data, we extend our framework to the case where the components are multinomial with an application to text clustering. Furthermore, our approach generalizes easily to the exponential family of distributions.

We further extend this distributed framework to co-clustering by proposing DisNPLBM, a scalable inference algorithm for Bayesian Non-Parametric Latent Block Models. DisNPLBM employs a master/worker architecture that distributes rows of the data matrix among workers, enabling parallel inference without inter-worker communication. It models latent multivariate Gaussian blocks to simultaneously partition rows and columns, effectively capturing complex co-cluster structures. We validate its scalability and accuracy on synthetic datasets and demonstrate its practical effectiveness on gene expression data.

Beyond scalability, algorithmic reliability is critical for trustworthy clustering results. We investigate the stability of the Expectation-Maximization (EM) algorithm for GMMs through the lens of average sensitivity, quantifying the sensitivity of inferred parameters to small data perturbations such as the removal of a single point. We theoretically prove that EM is stable under fixed, equal mixture proportions and extend this analysis to the more general setting with unknown proportions. Our results assume spherical clusters with known diagonal covariance and do not rely on cluster separability or initialization conditions. Extensive experiments on synthetic and real datasets corroborate our theoretical findings, underscoring the robustness of EM in practical scenarios.

Acknowledgement

First and foremost, I would like to express my deepest gratitude to my supervisors, Professor Mustapha Lebbah and Professor Hanene Azzag, for their invaluable guidance, constant encouragement, and unwavering support throughout this PhD journey. Their expertise, vision, and patience have been a continuous source of inspiration and growth.

I am also sincerely thankful to Professor Yuichi Yoshida, who supervised me during my five-month international research mobility at the National Institute of Informatics (NII), Japan. Working under his insightful guidance was both inspiring and intellectually enriching, and it stands out as an invaluable experience in my doctoral journey. His insightful feedback and generous mentorship greatly deepened my research perspective. I am also grateful to NII for hosting me. I also gratefully acknowledge DATAIA and GS ISN for making this mobility possible through their financial support.

I am indebted to Djamel Bouchaffra and Etienne Goffinet for the stimulating discussions, valuable insights, and kind help they shared along the way. My thanks also extend to the research environments that welcomed and supported me: Laboratoire DAVID and LIPN, where I had the privilege of being an associate member. I warmly acknowledge the administrative staff and secretaries from both labs for their efficiency and constant assistance, which greatly eased my daily work.

This work has further benefited from the support of the Paris Île-de-France Région, within the framework of DIM AI4IDF, whose funding was instrumental in carrying out my research. I would also like to thank Grid5000 for providing the essential computational resources that made large-scale experiments possible.

Finally, none of this would have been possible without the unwavering support of my colleagues, friends, and family. Their encouragement, patience, and kindness provided the balance and strength I needed to complete this journey. To all of them, I extend my heartfelt gratitude.

Contents

Introduction

Context and motivations

In recent years, the fast growth of data across domains such as text mining, bioinformatics, and computer vision has brought attention to clustering as a fundamental unsupervised learning task [58]. The goal of clustering is to partition data into meaningful clusters such that similar samples are grouped together while dissimilar samples are assigned to different clusters.

Model-based clustering assumes that observations are generated from a probabilistic model defined over the observation space. Typically, it is assumed that data points within the same cluster are independently drawn from the same distribution. A common choice in this framework is the family of mixture models [112], which assumes that the data are drawn from a mixture of weighted component distributions. This formulation provides a flexible and interpretable approach for analyzing complex datasets and has been widely adopted for its simplicity and efficiency. For parametric mixture models, the Expectation-Maximization (EM) [43] algorithm is widely used in practice to perform inference, owing to its simplicity.

However, a major limitation of these models lies in the requirement to specify the number of clusters in advance, which is unsuitable in a fully unsupervised setting where no labels are available. Moreover, in scenarios where the dataset size increases, the true number of clusters may also grow, making the assumption of a fixed number of clusters unrealistic.

Bayesian non-parametric (BNP) models, and particularly Dirichlet Process Mixture Models (DPMMs) [10], address this limitation by allowing the number of clusters to be automatically inferred from the data. This flexibility makes them an attractive choice for cluster analysis, especially when we have no prior knowledge of the underlying clustering structure.

Beyond traditional clustering, some tabular datasets exhibit a dual structure between observations and features, such as users and items in recommendation systems, or genes and samples in transcriptomics. In these settings, co-clustering[65]—which simultaneously groups the rows and columns of a data matrix to produce a block partition—offers a richer representation of the data and often outperforms traditional clustering that only infers row partitions. Bayesian non-parametric latent block models (NPLBMs) [113] naturally extend the DPMM framework to this setting, enabling the automatic inference of both the number of row and column clusters during the inference process.

The inference of parameters in BNP models such as DPMMs or NPLBMs can be performed using two main approaches: Markov Chain Monte Carlo

(MCMC) inference [121] and variational inference [19]. MCMC methods have the advantage of producing asymptotically exact samples from the target distribution [138], whereas variational inference does not offer such guarantees and instead finds a tractable distribution that approximates the target. Nevertheless, variational inference is often faster than MCMC methods [20]. A well-known MCMC technique for DPMMs is Gibbs sampling [121], which iteratively updates both the cluster assignments and the parameters of the component distribution corresponding to each cluster. In the case where the prior is conjugate to the likelihood, the parameters can be integrated out, and only the cluster assignments have to be estimated. This leads to the collapsed Gibbs sampling variant.

The collapsed Gibbs sampler is an efficient MCMC method, since it avoids sampling component parameters and has good asymptotic properties with high estimation accuracy. However, when the number of observations is large, the inference process becomes computationally expensive and thus very slow. This limitation restricts its applicability to massive datasets and real-world scenarios.

Several existing works on mixture models assume Gaussian components due to their computational convenience. While this assumption may be suitable for continuous data, it is inadequate in many scenarios, particularly when dealing with discrete or count data. For instance, Dirichlet-multinomial distributions are more appropriate for text clustering, and Poisson or Gamma–Poisson models are better suited for gene expression analysis. Thus, it is crucial to extend Bayesian non-parametric mixture models beyond Gaussian components to enhance their applicability across diverse domains.

Finally, the outputs of clustering algorithms often guide decision-making in practical applications, particularly in safety-critical domains such as healthcare [50] and network security [186]. It is therefore essential to evaluate not only their clustering quality but also their stability under small perturbations of the input data. In many real-world scenarios, data may be incomplete or subject to random deletions, raising the question of how sensitive the clustering outputs are to these small changes. Recent paradigms, such as average sensitivity [116], provide a principled means to quantify the stability of learning algorithms, offering valuable insights into their reliability in applied settings.

The contributions of this thesis directly address these challenges. In the first part, we develop scalable inference methods for Dirichlet Process Mixture Models capable of handling large-scale datasets without compromising accuracy. Our aim is twofold: on one hand, to design an approach that can be readily extended to co-clustering techniques, such as Non-Parametric Latent Block Models (NPLBMs); on the other hand, to make it flexible enough to accommodate a richer family of component distributions beyond the Gaussian

case. In the second part, we provide theoretical insights by investigating the stability of the classical Expectation–Maximization (EM) algorithm for clustering through the lens of average sensitivity.

Overview

The remainder of this manuscript is organized as follows:

Chapter 1: Related Work

This chapter provides an overview of clustering techniques, with a focus on model-based approaches. It first introduces Gaussian Mixture Models (GMMs) and the Expectation-Maximization algorithm, highlighting their connection to the K-means algorithm. It then presents Dirichlet Process Mixture Models as a Bayesian non-parametric extension of Gaussian Mixture Models, with the inference using the Collapsed Gibbs Sampler. The chapter also discusses the co-clustering problem, focusing on Non-Parametric Latent Block Models. Then, it reviews widely used distributed computing frameworks and existing distributed inference methods for Dirichlet Process Mixture Models and co-clustering algorithms. Finally, it describes techniques for evaluating clustering performance, both in terms of quality and stability.

Chapter 2: Distributed inference for DPMMs with a Multivariate Gaussian components

This chapter presents a distributed inference algorithm for Dirichlet Process Mixture Models. In this chapter, the component distributions are assumed to be multivariate Gaussian. This distributed methodology is inspired by the Collapsed Gibbs Sampler algorithm and employs the master/worker architecture. The sufficient statistics make it possible to aggregate local results and estimate the global clustering structure.

Chapter 3: Distributed Inference for DPMMs with Multinomial Components and Generalization to the Exponential Family

This chapter adapts the distributed inference method presented in Chapter 2, where component distributions are assumed to be Gaussian, to the case where they are Multinomial, with an application to text clustering. Moreover, it demonstrates how the approach generalizes to the exponential family of distributions, thereby extending its applicability beyond Gaussian and Multinomial models.

Chapter 4: Distributed Inference for Bayesian Non-Parametric Latent Block Models

This chapter extends the proposed distributed inference framework for Dirichlet Process Mixture Models to co-clustering, introducing a distributed inference method for the Bayesian Non-Parametric Latent Block Model. The method also adopts a master–worker architecture, with data partitioned row-wise across workers.

Chapter 5: Average sensitivity analysis of the EM algorithm of Gaussian Mixture Models

The work presented in this chapter was conducted during an international mobility at the National Institute of Informatics (NII)[1], Japan. It provides a theoretical analysis of the stability of the Expectation–Maximization algorithm for Gaussian Mixture Models, from the perspective of average sensitivity, which quantifies how much the output changes under small perturbations of the input.

Chapter 6: Conclusion and perspectives

This chapter outlines possible extensions of the methods proposed in this thesis. In particular, it considers alternative implementations of the distributed approaches, such as deployment in federated environments, as well as enhancements to improve clustering quality. It also explores how these methods could be adapted to distribute other clustering techniques. Finally, it discusses potential extensions of the stability analysis of the Expectation–Maximization algorithm based on average sensitivity.

Publications and submissions

These contributions have been the subject of several publications and submissions, which are listed below:

Publications

Journals

- Reda Khoufache, Mustapha Lebbah, Hanene Azzag, Etienne Goffinet, Djamel Bouchaffra. A Distributed Inference Algorithm for Dirichlet Process Mixture Models with Exponential Family Components. Neurocomputing, 2025, 131119, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2025.131119

[1] https://www.nii.ac.jp/en/about/international/mouresearch/internship2024-1/

International conferences

- Reda Khoufache, Anisse Belhadj, Hanane Azzag, Mustapha Lebbah. Distributed MCMC inference for Bayesian Non-Parametric Latent Block Model. Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2024). Taipei, Taiwan. 7-10 May 2024.

- Reda Khoufache, Mustapha Lebbah, Hanene Azzag, Étienne Goffinet, and Djamel Bouchaffra. Distributed Collapsed Gibbs Sampler for Dirichlet Process Mixture Models in Federated Learning. In Proceedings of the 2024 SIAM International Conference on Data Mining (SDM 2024), April 18 - 20, Houston, Texas, United States.

National conferences

- Reda Khoufache, Anisse Belhadj, Hanane Azzag, Mustapha Lebbah. Inférence MCMC distribuée pour les modèles à blocs latents Bayésiens non-paramétriques. Société Francophone de Classification (SFC 2024). Marseille, France. 10-13 Septembre 2024.

- Reda Khoufache, Mustapha Lebbah, Hanene Azzag, Étienne Goffinet, and Djamel Bouchaffra. Inférence distribuée pour les modèles de mélange de processus de Dirichlet dans l'apprentissage fédéré, 55èmes Journées de Statistique de la SFdS, May 27 - 31, 2024, Bordeaux, France.

Submission

- Reda Khoufache, Mustapha Lebbah, Hanene Azzag, Yuichi Yoshida. Average Sensitivity Analysis of the Expectation Maximization Algorithm for the Gaussian Mixture Model. *Submitted to the International Conference on Artificial Intelligence and Statistics (AISTATS 2026).*

Open-source code

- **DisCGS for continuous data.** Implementation of the distributed inference method for the Dirichlet Process Mixture Model with Gaussian components proposed in Chapter 3. https://github.com/redakhoufache/DisCGS

- **DisCGS for discrete data.** Implementation of the distributed inference method for Dirichlet Process Mixture Models with Multinomial components proposed in Chapter 4. https://github.com/redakhoufache/DisCGS-for-Discrete-data

- **DisNPLBM.** Implementation of the distributed inference method for the Bayesian Non-Parametric Latent Block Model introduced in Chapter 5. https://github.com/redakhoufache/Distributed-NPLBM
- **Average Sensitivity Analysis of the EM algorithm.** Implementation of the experiments presented in Chapter 6. https://github.com/redakhoufache/Average-sensitivity-analysis-of-the-EM

Introduction

Contexte et motivations

Ces dernières années, la croissance rapide des données dans des domaines tels que l'exploration de textes, la bio-informatique et la vision par ordinateur a beaucoup attiré l'attention sur *le clustering* en tant que problème fondamental dans l'apprentissage non supervisé [58]. L'objectif du *clustering* est de partitionner les données en des groupes cohérents appelés *clusters*, de manière à ce que les observations similaires soient regroupés dans de mêmes *clusters* tandis que les observations dissemblables soient assignés à des *clusters* différents.

Les modèles probabilistes approchent le problème de *clustering* en supposant que les observations sont générées par une loi de probabilité définie sur l'espace d'observations. On suppose généralement que les observations appartenant au même cluster suivent indépendamment une même distribution. Un choix naturel dans ce cadre est la famille des modèles de mélange [112], qui représentent les données comme issues d'un mélange de distributions pondérées par un vecteur de probabilité. Cette modélisation offre une approche flexible et interprétable pour analyser des ensembles de données complexes et a été largement adoptée pour sa simplicité. Pour les modèles de mélange paramétriques, l'algorithme *Expectation–Maximization* [43] est largement utilisé en pratique pour inférer les paramètres.

Cependant, la limitation majeure de ces modèles réside dans la nécessité de spécifier à l'avance le nombre de *clusters*, ce qui est inadapté dans un contexte non supervisé où les données ne sont étiquetées. De plus, dans les situations où la taille de l'échantillon augmente, le nombre réel de *clusters* peut également croître, ainsi l'hypothèse d'un nombre fixe de *clusters* inadéquate et restrictive.

Les modèles bayésiens non paramétriques, et en particulier les modèles de mélange de processus de Dirichlet (*Dirichlet Process Mixture Models, DPMMs*) [10], surpassent cette limitation en permettant d'inférer automatiquement le nombre de clusters à partir des données. Cette flexibilité rend ces modèles plus attrayants et intéressants pour la tâche du *clustering*, particulièrement lorsqu'on ne dispose d'aucune information préalable sur la vraie structure de *clustering*.

Lorsque l'ensemble de données présente une relation duale entre les observations et les variables, comme par exemple les utilisateurs et les articles dans les systèmes de recommandation, ou les gènes et les échantillons en transcriptomique, le *co-clustering* [65]—qui regroupe de façon simultanée les lignes et les colonnes d'un tableau de données pour inférer une parti-

tion en blocs homogènes—offre une meilleure représentation des données et surpasse souvent les méthodes de *clustering* classiques qui ne partitionnent que les lignes sans tenir compte de cette relation. Les modèles à blocs latents bayésiens non paramétriques (*Non-Parametric Latent Block Models*) [113] représentent une extension naturelle des *DPMMs* au contexte du *co-clustering*, permettant ainsi l'inférence automatique du nombre de clusters de lignes et de colonnes durant l'inférence.

Deux approches principales permettent l'inférence des paramètres dans ces modèles : l'inférence par chaînes de Markov de Monte Carlo [121] et l'inférence variationnelle [19]. Les méthodes *MCMC* présentent l'avantage de produire des échantillons asymptotiquement exacts de la distribution cible [138], tandis que l'inférence variationnelle n'offre pas de telles garanties, mais construit plutôt une distribution qui approxime la distribution cible. Néanmoins, l'inférence variationnelle est souvent plus rapide que les méthodes *MCMC* [20]. Une technique *MCMC* bien connue pour les *DPMMs* est l'échantillonneur de Gibbs [121], qui met à jour itérativement les affectations de *clusters* ainsi que les paramètres de la distribution composante correspondant à chaque *cluster*. Lorsque la distribution a priori est conjuguée, les paramètres peuvent être intégrés, et seules les affectations de clusters doivent être estimées, ce qui conduit à une variante de l'échantillonneur de Gibbs (*collapsed Gibbs sampling*).

L'échantillonneur de Gibbs est une méthode *MCMC* efficace, puisqu'il n'est plus nécessaire d'échantillonner les paramètres associés à chaque composantes. Cette approche est largement utilisée en raison de ses bonnes propriétés asymptotiques. Cependant, lorsque le nombre d'observation est grand, le processus d'inférence devient considérablement lent à cause du coût computationnel élevé, ce qui entrave le passage à l'échelle et restreint son applicabilité aux jeux de données massifs.

Par ailleurs, de nombreux travaux existants sur les modèles de mélange supposent que les distributions associées à chaque *cluster* sont gaussiennes, pour des raisons de simplicité de calcul. Bien que cette hypothèse soit adaptée aux données continues, elle devient inadéquate dans de nombreux scénarios réels, par exemple lorsqu'on traite des données discrètes ou des données de comptage. Par exemple, dans le *clustering* de documents, les distributions Dirichlet–Multinomiales sont plus appropriées, tandis qu'en analyse d'expression de gènes, les modèles Gamma–Poisson sont mieux adaptés. Il est donc crucial d'étendre les modèles de mélange bayésiens non paramétriques au-delà des composantes gaussiennes, afin d'élargir leur applicabilité à divers domaines.

Enfin, les résultats des algorithmes de *clustering* jouent un rôle important dans la prise de décision dans des applications réelles, notamment dans des domaines tels que la santé [50] et la cybersécurité [186]. Il est donc essentiel

d'évaluer non seulement leur qualité de *clustering*, mais aussi leur stabilité face à de petites perturbations dans les données d'entrée. Dans de nombreux contextes réels, les données peuvent être incomplètes ou soumises à des suppressions aléatoires, ce qui soulève la question de la stabilité des algorithmes face à ces perturbations. Des outils récents, tels que la sensibilité moyenne [116], offrent un cadre formel et rigoureux pour quantifier la stabilité des algorithmes d'inférence et fournir des informations sur leur fiabilité dans des contextes appliqués.

Les contributions de cette thèse répondent directement à ces enjeux. Dans une première partie, nous développons des méthodes d'inférence distribuées pour les modèles de mélange à processus de Dirichlet, capables de traiter de grands ensembles de données sans compromettre la qualité du *clustering*. Notre objectif est, d'une part, de concevoir une approche facilement extensible aux techniques de co-clustering, telles que les modèles à blocs latents non paramétriques, et, d'autre part, de garantir une flexibilité suffisante pour considérer une famille plus riche de distributions de composantes, au-delà du cas gaussien. Dans une seconde partie, nous menons une analyse théorique en étudiant la stabilité de l'algorithme classique *Expectation–Maximization* pour le *clustering*, à travers la sensibilité moyenne.

Plan de la thèse

La suite de ce manuscrit est organisée comme suit :

Chapitre 2: Related Work

Ce chapitre propose une synthèse des techniques de *clustering* existantes, en mettant l'accent sur les approches basées sur les modèles probabilistes. Il introduit d'abord les modèles de mélange gaussiens et l'algorithme *Expectation–Maximization* et sa connexion avec l'algorithme des K-moyennes. Il présente ensuite les modèles de mélange à processus de Dirichlet comme une extension bayésienne non paramétrique des modèles de mélange gaussiens, ainsi que l'inférence via l'échantillonnage de Gibbs. Ce chapitre aborde également le problème du *co-clustering*, et rappelle la définition des modèles à blocs latents non paramétriques. Il passe ensuite en revue les outils de calcul distribué les plus utilisés ainsi que les méthodes d'inférence distribuée existantes pour les modèles de mélange à processus de Dirichlet et les algorithmes de *co-clustering*. Enfin, il décrit les techniques d'évaluation des algorithmes de *clustering*, tant en termes de qualité de *clustering* et de stabilité.

Chapitre 3: Distributed inference for DPMMs with a Multivariate Gaussian components

Ce chapitre présente un algorithme d'inférence distribuée pour des modèles de mélange de processus de Dirichlet en utilisant les statistiques suffisantes. Dans ce chapitre, les distributions associées à chaque *cluster* sont supposées être gaussiennes multivariées. Cette approche distribuée a pour objectif d'approximeer l'algorithme *collapsed Gibbs sampler* et est basée sur une architecture *master/worker*.

Chapitre 4: Distributed Inference for DPMMs with Multinomial Components and Generalization to the Exponential Family

Ce chapitre introduit une méthode d'inférence distribuée pour les modèles de mélange de processus de Dirichlet dans le cas où les distributions composantes sont multinomiales, avec une application au *clustering* de textes. Il montre également comment cette méthode se généralise au cas où les distributions associées à chaque *cluster* sont supposées appartenir à la famille exponentielle, étendant ainsi son applicabilité au-delà des modèles gaussiens et multinomiaux.

Chapitre 5: Distributed Inference for Bayesian Non-Parametric Latent Block Models

Ce chapitre étend le cadre d'inférence distribuée proposé pour les modèles de mélange de processus de Dirichlet au *co-clustering*, en introduisant une méthode d'inférence distribuée pour le modèle à blocs latents bayésien non paramétrique. La méthode adopte également une architecture *master/worker*, avec une disribution des données par lignes sur les différents *workers*.

Chapitre 6: Average sensitivity analysis of the EM algorithm of Gaussian Mixture Models

Ce chapitre présente une analyse théorique de la stabilité de l'algorithme *Expectation–Maximization* pour les modèles de mélange gaussiens, du point de vue de la sensibilité moyenne, qui quantifie dans quelle mesure les paramètres estimés par l'algorithme *EM* varient sous de petites perturbations des données d'entrée.

Chapitre 7: Conclusion and perspectives

Ce chapitre expose les extensions possibles des méthodes proposées dans cette thèse. Il aborde notamment des implémentations alternatives des approches distribuées, comme leur déploiement en environnement fédéré, ainsi

que des améliorations visant à améliorer la qualité du *clustering*. Il explore également comment ces méthodes pourraient être adaptées pour distribuer d'autres techniques de *clustering*. Enfin, il discute des extensions potentielles de l'analyse de stabilité de l'algorithme *Expectation–Maximization* basée sur la sensibilité moyenne.

Notation

Table 1: Table of Notation

General	
Linear Algebra	
$\mathbb{R}$	Real numbers
I	Identity matrix
$\mathrm{d}(\cdot, \cdot)$	Distance
$\|\cdot\| = \|\cdot\|_2$	Euclidean norm
$(\cdot)^T$	Transpose operator
$\det(\cdot)$	Determinant
Probability and Statistics	
$\mathbb{1}$	Indicator function
M	Iteration number
$\Gamma(\cdot)$	Gamma function
$x \sim \mathcal{D}(\theta)$	x is drawn from probability distribution $\mathcal{D}(\theta)$ parameterized by θ
$\mathrm{p}(x \mid \theta) := \mathcal{D}(x \mid \theta)$	Probability density at x
$\ell(\cdot)$	Data log-likelihood
$\ell_c(\cdot)$	Complete-data log-likelihood
$\mathcal{Q}$	Expected complete-data log-likelihood
$\mathbb{H}(\cdot)$	Entropy
Dataset	
Dimensions	
n	Number of observations
p	Number of variables
d	Representation space dimension
V	Vocabulary size
Data Structures	
$\mathbf{x} \in \mathbb{R}^{n \times d}$	Univariate dataset
$X \in \mathbb{R}^{n \times p \times d}$	Multivariate dataset
X_{-i}/X_{-j}	X without row i / column j
$\mathbf{x}_{i,\cdot}/\mathbf{x}_{\cdot,j}$	Row i / column j of X
Mixture Model and Dirichlet Process Mixture Model	
Model Structure	
K	Number of components/clusters
$\mathbf{z}$	Membership vector
Θ	Parameter space
$\theta_k \in \Theta$	Parameter of component k

$f(\cdot, \theta_k)$	Component distribution parametrized by θ_k
	Parameters and Hyperparameters
$\boldsymbol{\pi}$	Mixture proportions
$(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate Gaussian parameters
$\boldsymbol{\theta}$	Component parameters
$\hat{\boldsymbol{\theta}}$	Estimate of $\boldsymbol{\theta}$
$\tilde{\boldsymbol{\theta}}$	Estimate from EM with deleted point
σ^2	Variance parameter
α	Concentration parameter
G_0	Base distribution
(Λ_0, τ_0)	Parameters of G_0
Ω	Hyperparameter set
$\lambda_0 = (\mu_0, \kappa_0, \Psi_0, \nu_0)$	NIW prior parameters
$\lambda_k = (\mu_k, \kappa_k, \Psi_k, \nu_k)$	Posterior hyperparameters for cluster k
	Counts and Sufficient Statistics
n_k	Size of cluster k
(T_k, S_k)	Sufficient statistics for Gaussian cluster k
$\boldsymbol{\gamma}$	Dirichlet distribution parameter
$\ell_{x,w}$	Number of times word w appears in document x
ℓ_x	Length of document x
$n_{k,w}$	Number of times word w appears in cluster k
N_k	Total number of word occurrences in cluster k
$\eta(\theta)$	Natural parameter vector
$S(x)$	Sufficient statistics
ψ	Log-partition function
ϕ	Base measure
	Co-clustering
	Structure
$\mathbf{w}$	Coclustering variable partition
$\boldsymbol{\rho}$	Coclustering column proportions
L	Number of column clusters
p_l	Column-cluster size
$(n_{k,l})$	Block-cluster size
$f(\cdot, \theta_{k,l})$	Block component distribution parametrized by $\theta_{k,l}$
	Parameters
β	Column-cluster concentration parameter
$\theta_{k,l}$	Block distribution parameters
$\mu_{k,l}, \kappa_{k,l}, \Psi_{k,l}, \nu_{k,l}$	Block cluster (k, l) posterior distribution parameters

1 - Related Work

1.1 . Clustering

Clustering, or cluster analysis, is a fundamental task in unsupervised machine learning that aims to automatically assign labels to unlabelled data by grouping similar instances together. The goal is to partition the data so that observations in the same cluster are more similar to each other than to observations in other clusters. In this way, clustering methods seek to maximize intra-cluster similarity while minimizing inter-cluster similarity. Clustering technique has been extensively studied over the past decades and has found utility across wide range of domains, including healthcare [105, 151], biology [2, 33], aviation [102, 55], text mining [176, 108], and recommender systems [13, 169]. Numerous clustering algorithms have been proposed to address various data types and structures. These methods have been continuously reviewed and compared in survey papers [171, 147, 141, 53, 126, 155]. Clustering techniques can be categorized based on the underlying assumptions and strategies used to identify homogeneous clusters in data.

1.1.1 . Hierarchical Clustering

Hierarchical clustering technique [84] aims to build a hierarchy of nested clusters, usually represented as a tree-like structure called a dendrogram (see Figure 1.1). In this representation, each leaf node corresponds to a single data point, while internal nodes denote clusters formed by merging or splitting other clusters. Meaningful partitions can be extracted depending on the desired level of granularity by cutting the dendrogram at a given depth level.

- **Agglomerative hierarchical clustering (bottom-up):** Agglomerative hierarchical clustering starts by considering each data point as an individual cluster. It then progressively merges the most similar clusters until all data points are contained in a single cluster. The main challenge in hierarchical clustering resides in determining which clusters to merge at each step. This process is performed using a similarity measure between clusters. Various hierarchical methods have been proposed based on different types of similarity metrics. For instance, distance-based similarity measures often utilize a linkage criterion, such as single linkage [8], complete linkage [156], average linkage [80], and centroid linkage [101]. Algorithms that adopt these criteria include CURE [67], Generic_Linkage [119], and HC-OT [30]. Other approaches use different similarity measures, such as nearest neighbor similarity [120, 35, 142] or the quality evaluation of partitions post-

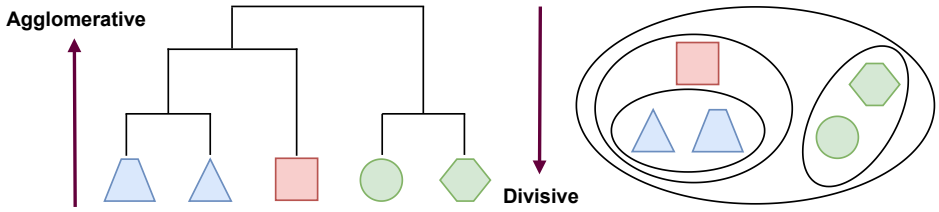

Figure 1.1: The representation of the dendrogram produced by hierarchical clustering (left) and the corresponding cluster structure (right).

merging [146, 74, 139, 124]. Additional agglomerative clustering algorithms comprise AGNES [87], BIRCH [185], and ROCK [68].

- **Divisive hierarchical clustering (top-down):** In contrast, the divisive method begins with all data points grouped into a single cluster. It then recursively splits this cluster into smaller sub-clusters until a termination condition is satisfied, such as reaching a desired number of clusters or when the data within each cluster is sufficiently homogeneous. Divisive clustering is typically composed of three steps: computing the dissimilarity matrix, selecting and splitting clusters based on this measure, and determining the level or depth of newly formed clusters in the dendrogram. Unlike agglomerative methods, where cluster levels arise naturally, divisive approaches often require custom strategies to define node levels. Divisive hierarchical clustering techniques comprise DIANA [87], DHCC [170] and DHClus [160].

Hierarchical clustering presents several strengths. It does not require specifying the number of clusters in advance, it handles arbitrary similarity or dissimilarity measures, and allows flexible partitioning at different levels of granularity by cutting the dendrogram at various levels. Despite these advantages, it also has limitations. For instance, once a merge or split operation is performed, it cannot be reversed, potentially leading to suboptimal clustering. Additionally, hierarchical methods are computationally intensive in both time and memory, with a time complexity that grows quadratically with the number of data points[145, 130].

1.1.2 . Partitional clustering

Unlike hierarchical clustering, partitional clustering [29], also known as partition-based clustering, begins with an initial partition of the data and iteratively updates it by optimizing some objective function [100]. Partitional clustering is illustrated in Figure 1.2. In these methods, each cluster is typically represented by a prototype or centroid. For this reason, they are sometimes called prototype-based clustering techniques [22].

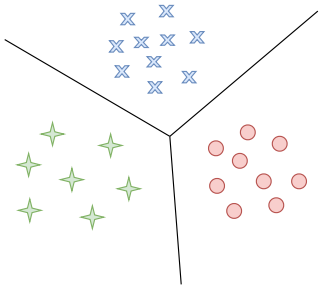

Figure 1.2: Illustration of partitional clustering with $K = 3$ clusters.

Among partitional clustering methods, the K-means algorithm [109] is one of the most widely used due to its simplicity and efficient implementation. The algorithm begins by selecting K initial centroids. Each data point is then assigned to the closest centroid based on a given distance. After the assignment step, the centroids of the clusters are updated by calculating the mean of the points within each cluster. These two steps are repeated iteratively until the centroids do not change. While K-means is guaranteed to converge to a local minimum [149], minimizing its objective function is known to be an NP-hard problem [79]. Other variants of the K-means algorithm include K-medoids [86], CLARA [85], CLARANS [123], K-modes [77], K-prototypes [78], and K-means++ [11].

In the K-means algorithm, each point is assigned to one and only one cluster. This type of assignment is known as a hard assignment, which leads to a type of clustering that is known as hard clustering. In contrast, soft (or fuzzy) clustering techniques allow each point to belong to more than one cluster. An example is the Fuzzy c-means algorithm [48, 15].

Partitional clustering algorithms offer several advantages. They are relatively scalable, simple, and easy to implement. These methods are particularly effective for datasets with compact, spherical, and well-separated clusters. However, they also have notable limitations. First, they tend to discover only spherical-shaped clusters and are thus not suited for identifying clusters with more complex shapes. Additionally, these methods often assume that clusters have approximately similar sizes, as data points are assigned to the nearest centroid. This can result in inaccurate cluster boundaries [72]. Moreover, partitional clustering algorithms, such as K-means, are highly sensitive to initialization; different initial centroid choices can yield significantly different results. Finally, their most critical limitation is the need to predefine the number of clusters K. In a real unsupervised learning setting, this number is often unknown due to the absence of labels. Recent works that have surveyed partitional clustering techniques include [157, 29, 133, 97].

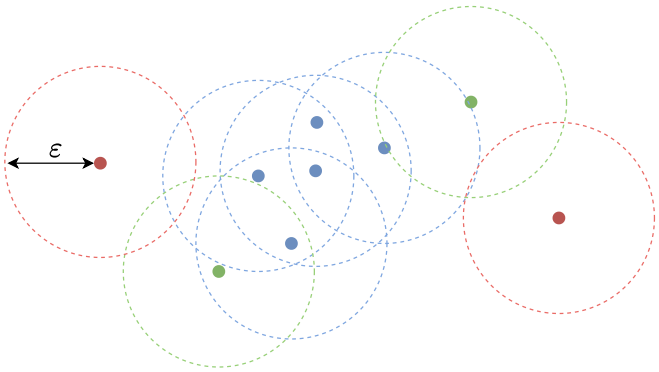

Figure 1.3: Illustration of density-based clustering approach with $MinPts = 4$. Core points are shown in blue, border points in green, and noise points in red.

1.1.3 . Density-based clustering

In density-based clustering approaches, clusters are identified as areas in the observation space with a high concentration of points, separated by regions of low density. The most popular algorithm is DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [52]. Given two parameters: a distance threshold ε and a density threshold $MinPts$. A point is said to be a core point if it has at least $MinPts$ neighbors within its ε-radius. If a point lies within the neighborhood of a core point but is not itself a core point, it is labeled as a border point. Any point that is neither a core nor a border point is labeled as noise. This clustering structure is illustrated in Figure 1.3.

DBSCAN begins by randomly selecting a point and checking whether it is a core point. If so, the algorithm expands the cluster by connecting it to all reachable core points. Border points are then assigned to the clusters of their associated core points, and noise points remain unclustered. This process repeats until all points have been visited and classified.

This algorithm offers several advantages: it does not assume any specific cluster shape, allowing it to identify clusters of arbitrary shapes; it does not require prior knowledge of the number of clusters; and it includes an explicit notion of noise, distinguishing outliers from clusters. However, its main drawback is its sensitivity to the choice of the two parameters, ε and $MinPts$, which can be difficult to set appropriately. Other notable density-based clustering algorithms include DENCLUE [76], OPTICS [9], HDBSCAN [27], DENCLUE 2.0 [75], and the density peaks clustering method [140]. Comprehensive surveys on these methods can be found in [94, 26].

1.1.4 . Model-based clustering

Model-based clustering approach assumes that the observations are generated from a probabilistic model defined over the observation space. Typi-

Figure 1.4: Illustration of model-based clustering using a Gaussian Mixture Model with three components

cally, it assumes that each cluster corresponds to a distinct component distribution and that data points within the same cluster are independently drawn from the same distribution. The most widely used model in this framework is the mixture model [112], which assumes that the data follow a mixture of weighted component distributions. Figure 1.4 illustrates data generated from three Gaussian components.

This formulation provides a flexible and interpretable way to analyze complex datasets. Once the model is specified, given a set of observations, the goal is to estimate its parameters and assign data points to clusters based on the components they are most likely generated from. This is commonly achieved through the Expectation-Maximization [43] algorithm, which iteratively estimates the parameters attempting to maximize the data likelihood.

One major drawback of traditional mixture models lies in the need to specify the number of clusters K. This poses challenges in real-world applications, as these models are sensitive to the number of clusters. Determining an appropriate number of clusters requires prior knowledge of the dataset, which is not easy since the true labels are not provided in unsupervised scenarios. Ideally, an algorithm should autonomously determine the most suitable number of clusters. This can be achieved using Dirichlet Process Mixture Models, which extend classical mixture models into the Bayesian non-parametric framework, as will be discussed later.

1.1.5 . Deep learning based clustering

Recent approaches combine clustering methods with deep neural networks, resulting in significant performance improvements. These deep clustering methods can be divided into two categories. The first category includes two-stage methods, which first extract features from the input data using

deep neural networks, such as variational autoencoders [91], and then apply a clustering algorithm to the extracted features. Notable works in this category include DCC [153], DDC [32], DDIC [136], and N2D [111]. The second category takes into account this separation. Therefore, these approaches propose a unified framework that jointly learns both deep representations and clusters. Examples of such works include VADE [82], SC-EDAE [5], IMDGC [175], and GamMM-VAE [69]. For comprehensive surveys on deep clustering, refer to [134, 165, 187, 135].

1.2 . Gaussian mixture models

In this section, we formally introduce Gaussian Mixture Models as probabilistic generative models. Then, we describe the inference process based on the Expectation-Maximization algorithm, and outline its connection to the K-means algorithm.

Mixture models [112] assume that the data are generated from a probabilistic generative process. Specifically, observations belonging to the same cluster are independently drawn from the same component distribution, and the cluster assignment of each observation is sampled from a categorical distribution parameterized by the mixture proportions (also known as the vector of weights). As a result, the overall data distribution can be expressed as a convex combination of the component distributions, where each component is weighted by its corresponding mixture proportion.

The most widely used mixture model is the Gaussian Mixture Model (GMM), also known as the mixture of Gaussians (MoG). In this framework, each component distribution is assumed to follow a multivariate Gaussian distribution. GMMs have a range of applications, such as data compression [182], outlier detection [132], generative classification [103], and density estimation [1]. However, their most common use is in clustering [174, 34], where they form the foundation of model-based approaches.

1.2.1 . Model definition

Let n, d and K be positive integers, $\mathbf{x} = (x_1, \dots, x_n)^T \in \mathbb{R}^{n \times d}$ the observed data, with $(\cdot)^T$ the transpose operator. Let $[n] = \{1, \dots, n\}$ and $[K] = \{1, \dots, K\}$ denote the sets of observation and cluster indices, respectively. Let $\mathcal{D}(\theta)$ be a probability distribution over $\mathbb{R}^d$ parametrized by θ. For a vector $x \in \mathbb{R}^d$, we write $\mathcal{D}(x \mid \theta)$ to denote its probability density at x. Let $\boldsymbol{\mu} = (\mu_k)_{k=1}^K$, where $\mu_k \in \mathbb{R}^d$ and $\boldsymbol{\Sigma} = (\Sigma_k)_{k=1}^K$, where $\Sigma_k \in \mathbb{R}^{d \times d}$ a symmetric positive semi-definite matrix. Let $\mathbf{z} = (z_i)_{i=1}^n$ be the *membership vector*, where z_i is a *latent variable* such that $z_i = k$ signifies that the x_i belongs to the k-th cluster. Finally, let $\boldsymbol{\pi} = (\pi_k)_{k=1}^K$ be a probability vector, i.e., it verifies

$$\forall k \in [K], \quad 0 \leq \pi_k \leq 1,$$

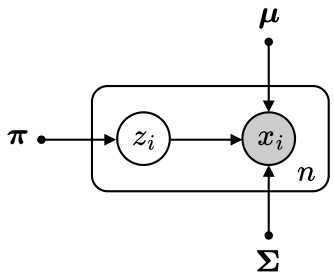

Figure 1.5: Graphical representation of a Gaussian Mixture Model.

and

$$\sum_{k=1}^{K} \pi_k = 1.$$

The *Gaussian mixture model* assumes that the observations are generated as follows

$$z_i \overset{\text{i.i.d.}}{\sim} \mathrm{Cat}\left(\boldsymbol{\pi}\right), \qquad \forall i \in [n],$$
$$x_i \mid \{z_i = k, \mu_k, \Sigma_k\} \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(\mu_k, \Sigma_k\right), \qquad \forall i \in [n].$$

Each observation x_i is sampled first by sampling its membership z_i from a categorical distribution $\mathrm{Cat}(\boldsymbol{\pi})$ parameterized by the *mixture weights* $\boldsymbol{\pi}$ (also called *mixture proportions*). Then, given the membership z_i and its corresponding component parameters $(\mu_{z_i}, \Sigma_{z_i})$, the observation $x_i|\{z_i = k, \mu_k, \Sigma_k\}$ is sampled from the normal distribution $\mathcal{N}(\mu_k, \Sigma_k)$. The distribution $\mathcal{N}(\mu_k, \Sigma_k)$ represents the *component distribution*. The graphical representation of GMMs is illustrated in Figure 1.5.

According to this definition, the probability distribution of x_i parameterized by $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{\pi}$ is given by

$$\begin{aligned}
\mathrm{p}(x_i|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \sum_{k=1}^{K} \mathrm{p}(x_i, z_i = k|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}), \\
&= \sum_{k=1}^{K} \mathrm{p}(z_i = k|\boldsymbol{\pi})\mathrm{p}(x_i|z_i = k, \boldsymbol{\mu}, \boldsymbol{\Sigma}), \\
&= \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k).
\end{aligned}$$

An illustration of a mixture of Gaussian densities and their corresponding i.i.d. generated data is shown in Figure 1.6.

The quantity π_k can be seen as a prior probability of $z_i = k$. The corresponding posterior probability is given by Bayes' rule

$$r_{ik} = \mathrm{p}(z_i = k|x_i, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

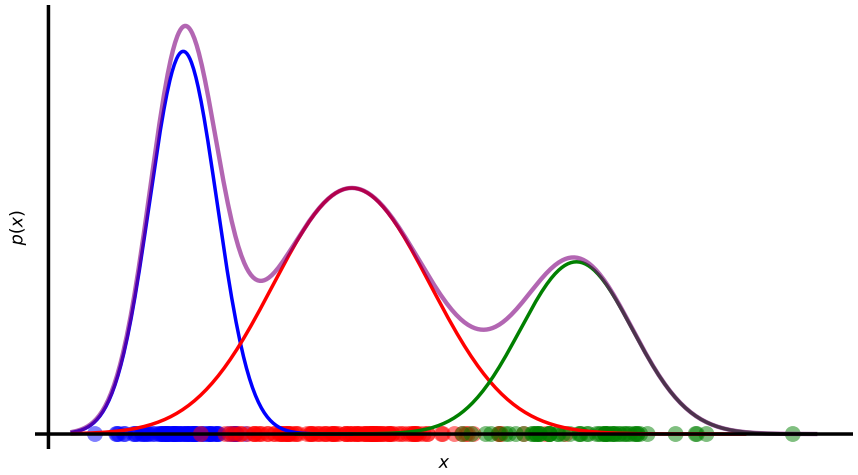

Figure 1.6: The mixture density (in purple) formed by three different Gaussian components (blue, red, and green), along with their corresponding generated clusters.

$$= \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{l=1}^{K} \pi_l \mathcal{N}(x_i|\mu_l, \Sigma_l)}. \tag{1.1}$$

The quantity r_{ik} is called a *responsibility*. We emphasize that only $\mathbf{x}$ is known. The membership vector $\mathbf{z}$, the parameters $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$ are unknown.

In the next section, we discuss how to estimate these unknown quantities from the data using the Expectation-Maximization algorithm. For convenience, we denote the full set of parameters by $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$.

1.2.2 . Inference: Expectation-Maximization

Given the observed data $\mathbf{x}$, the goal is to find a good estimate of the unknown parameters $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$. The data log-likelihood is given by

$$\begin{aligned} \ell(\boldsymbol{\theta}) &= \log \mathrm{p}(\mathbf{x}|\boldsymbol{\theta}), \\ &= \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\theta_k, \Sigma_k) \right). \end{aligned}$$

The maximization of the data log-likelihood with respect to $\boldsymbol{\theta}$ is a complex problem due to the presence of the summation over k inside the $\log$ function. The *Expectation-Maximization* (EM) algorithm is an alternative to finding the maximum likelihood estimator (MLE). In this approach, we consider the log-likelihood of the complete data $\{\mathbf{x}, \mathbf{z}\}$ defined by

$$\begin{aligned} \ell_c(\boldsymbol{\theta}) &= \log \mathrm{p}(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}), \\ &= \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}_{\{z_i=k\}} \left[\log \pi_k + \log \mathcal{N}(x_i|\mu_k, \Sigma_k) \right], \end{aligned} \tag{1.2}$$

where $\mathbb{1}$ is the indicator function. Maximizing such a function is much easier than the data log-likelihood. However, in practice, only $\mathbf{x}$ is observed (which is referred to as the incomplete data). Hence, in the E step of the EM algorithm, we evaluate the expectation of the complete data log-likelihood where this expectation is taken under the posterior distribution $\mathrm{p}(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta})$. Afterward, we maximize this expectation, which consists of the M step of the EM algorithm. Specifically, given some initial parameters $\hat{\boldsymbol{\theta}}^{(0)} = \left(\hat{\boldsymbol{\pi}}^{(0)}, \hat{\boldsymbol{\mu}}^{(0)}, \hat{\boldsymbol{\Sigma}}^{(0)}\right)$, the EM algorithm alternates between these two steps at each iteration $m \in [M]$:

1. **E step:** Evaluate $\mathrm{p}\left(\mathbf{z}|\mathbf{x}, \hat{\boldsymbol{\theta}}^{(m-1)}\right)$ and set:

$$\begin{aligned}\mathcal{Q}\left(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}^{(m-1)}\right) &= \mathbb{E}_{\mathbf{z}\sim\mathrm{p}\left(\mathbf{z}|\mathbf{x}, \hat{\boldsymbol{\theta}}^{(m-1)}\right)}\left[\ell_c(\boldsymbol{\theta})\right], \\ &= \sum_{\mathbf{z}} \mathrm{p}\left(\mathbf{z}|\mathbf{x}, \hat{\boldsymbol{\theta}}^{(m-1)}\right) \log \mathrm{p}\left(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}\right), \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} \hat{r}_{ik}^{(m)} \left[\log \pi_k + \log \mathcal{N}(x_i|\mu_k, \Sigma_k)\right],\end{aligned}$$

   where $\hat{r}_{ik}^{(m)} = \mathrm{p}\left(z_i = k|x_i, \hat{\boldsymbol{\pi}}^{(m-1)}, \hat{\boldsymbol{\mu}}^{(m-1)}, \hat{\boldsymbol{\Sigma}}^{(m-1)}\right)$ is given by the equation 1.1.

2. **M step:** Maximize $\mathcal{Q}\left(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}^{(m-1)}\right)$ with respect to $\boldsymbol{\theta}$ and set:

$$\hat{\boldsymbol{\theta}}^{(m)} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}\left(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}^{(m-1)}\right).$$

In the Gaussian case, the parameters associated with each component k are updated as follows

$$\hat{\mu}_k^{(m)} = \frac{1}{\hat{r}_k^{(m)}} \sum_{i=1}^{n} \hat{r}_{ik}^{(m)} x_i, \tag{1.3}$$

$$\hat{\Sigma}_k^{(m)} = \frac{1}{\hat{r}_k^{(m)}} \sum_{i=1}^{n} \hat{r}_{ik}^{(m)} \left(x_i - \hat{\mu}_k^{(m)}\right)\left(x_i - \hat{\mu}_k^{(m)}\right)^T, \tag{1.4}$$

$$\hat{\pi}_k^{(m)} = \frac{\hat{r}_k^{(m)}}{n}, \tag{1.5}$$

where $\hat{r}_k = \sum_{i=1}^{n} \hat{r}_{ik}$ and $(\cdot)^T$ denotes the transpose operator. The EM algorithm monotonically increases the data log-likelihood until it reaches a local maximum. The pseudo-code of the EM algorithm is given in Algorithm 1.

1.2.3 . Derivation of the K-means Algorithm

Let us consider the case where the mixture proportions are uniform, i.e., $\pi_1 = \cdots = \pi_K = 1/K$, fixed and known. Let us also assume spherical clusters

Algorithm 1 Expectation-Maximization algorithm

Input: Dataset $\mathbf{x}$,
    Component number K,
    Initial parameters $\hat{\boldsymbol{\pi}}^{(0)}$, $\hat{\boldsymbol{\mu}}^{(0)}$, and $\hat{\boldsymbol{\Sigma}}^{(0)}$,
    Iteration number M.
For $m \leftarrow 1$ **to** M **do**:
    // E-step: Compute the responsibilities
    For $i \leftarrow 1$ **to** n **do**:
        For $k \leftarrow 1$ **to** K **do**:
            Compute responsibility $\hat{r}_{ik}^{(m)}$ using Equation (1.7).
    // M-step: update component parameter
    For $k \leftarrow 1$ **to** K **do**:
        Compute the new parameters $\hat{\mu}_k^{(m)}$, $\hat{\Sigma}_k^{(m)}$, and $\hat{\pi}_k^{(m)}$ using Equations (1.3) to (1.5).
Output: The estimated components $\hat{\boldsymbol{\mu}}^{(M)}$, $\hat{\boldsymbol{\Sigma}}^{(M)}$, and $\hat{\boldsymbol{\pi}}^{(M)}$.

with known and fixed covariances, $\Sigma_1 = \cdots = \Sigma_K = \sigma^2 I$ where I is a $d \times d$ identity matrix. Hence, the goal of the EM algorithm is to estimate the means $\mu_1, \ldots, \mu_K \in \mathbb{R}^d$. For this purpose, given the initial parameters $(\hat{\mu}_1^{(0)}, \ldots, \hat{\mu}_K^{(0)})$, at each iteration m, the EM algorithm performs the following two steps:

1. **E step:** For each pair $(i, k) \in [n] \times [K]$ set:

$$\begin{aligned}\hat{r}_{ik}^{(m)} &= \frac{\mathcal{N}\left(x_i \mid \hat{\mu}_k^{(m-1)}, \sigma^2 I\right)}{\sum_{\ell \in [K]} \mathcal{N}\left(x_i \mid \hat{\mu}_\ell^{(m-1)}, \sigma^2 I\right)}, \\ &= \frac{\exp\left(-\frac{1}{2\sigma^2}\|x_i - \hat{\mu}_k^{(m-1)}\|^2\right)}{\sum_{\ell \in [K]} \exp\left(-\frac{1}{2\sigma^2}\|x_i - \hat{\mu}_\ell^{(m-1)}\|^2\right)}.\end{aligned}$$

2. **M step:** For each $k \in [K]$, update the parameter $\hat{\mu}_k^{(m-1)}$ associated with the component k as follows:

$$\hat{\mu}_k^{(m)} = \frac{1}{\hat{r}_k^{(m)}} \sum_{i \in [n]} \hat{r}_{ik}^{(m)} x_i.$$

We can easily notice that $\hat{r}_{ik}^{(m)} \to \hat{\delta}_{ik}^{(m)}$ when $\sigma \to 0$ where

$$\hat{\delta}_{ik}^{(m)} = \begin{cases} 1 & \text{if } k = \arg\min_{k=1,\ldots,K} \|x_i - \hat{\mu}_k^{(m)}\|, \\ 0 & \text{otherwise.} \end{cases} \tag{1.6}$$

Hence, the E-step of the EM algorithm is equivalent to an assignment function that assigns each data point to its closest centroid. Moreover, as $\sigma \to 0$, the

responsibility $\hat{r}_k^{(m)}$ converges to $n_k^{(m)}$, where $n_k^{(m)}$ denotes the cardinal of the k-th cluster at iteration m. Finally, the M-step is equivalent to the following update

$$\hat{\mu}_k^{(m)} = \frac{1}{n_k^{(m)}} \sum_{i=1}^{n} \hat{\delta}_{ik}^{(m)} x_i, \tag{1.7}$$

The EM algorithm alternates between the assignment and update steps. This process is sometimes called hard EM, and is commonly known as the K-*means* algorithm. Moreover, maximizing the expected complete-data log-likelihood Equation (1.2) becomes equivalent to maximizing

$$-\frac{1}{2} \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{1}_{\{z_i=k\}} \|x_i - \mu_k\|^2.$$

This is equivalent to minimizing the objective function of the K-means algorithm. The pseudo code of the K-means algorithm is given in Algorithm 2.

Algorithm 2 K-means Algorithm

Input: Dataset $\mathbf{x}$,
    Number of clusters K,
    Initial centroids $\hat{\boldsymbol{\mu}}^{(0)}$,
    Iteration number M.
for $m \leftarrow 1$ to M **do**
    // E-step: Cluster assignment
    for $i \leftarrow 1$ to n **do**
        Assign x_i to the nearest centroid using Equation (1.6).
    // M-step: Centroid update
    for $k \leftarrow 1$ to K **do**
        Update centroid $\hat{\mu}_k^{(m)}$ using Equation (1.7).
Output: Final centroids $\hat{\boldsymbol{\mu}}^{(M)}$ and membership vector $\hat{\mathbf{z}}^{(M)}$.

1.3 . Model Selection

In parametric model-based clustering methods, the number of clusters K is assumed to be known and fixed. In addition, this number must be specified in advance. However, in practice, the true number of clusters is often unknown in a fully unsupervised setting and difficult to determine in real-world scenarios.

To address this limitation, several model selection criteria have been proposed to estimate the most appropriate K by penalizing the likelihood function. Common examples include the Akaike Information Criterion (AIC) [6],

the Bayesian Information Criterion (BIC) [148], and the Integrated Completed Likelihood (ICL)[16]. These criteria attempt to balance model fit and complexity, but they typically require evaluating the model for multiple candidate values of K and selecting the one with the best score. This approach implicitly requires that the true number of clusters lies within the set of candidate values.

Despite their utility, these techniques come with important limitations. First, they often assume some prior knowledge of the underlying data structure. Second, they may fail to accurately recover the number of clusters in complex or high-dimensional data. Furthermore, the notion of a "true" or optimal number of clusters is not well defined and may not be unique in real-world and fully unsupervised scenarios [118].

The challenge becomes even more pronounced when dealing with complex, increasing, or streaming data, where the number of clusters may grow or change dynamically. In such cases, assuming a fixed number of clusters is restrictive. Instead, it is more appropriate to let the model infer the number of clusters during the inference process, enabling it to discover new clusters or remove unnecessary ones. This flexibility is one of the key motivations for adopting Bayesian non-parametric models such as Dirichlet Process Mixture Models, which will be discussed in the next section.

1.4 . Dirichlet process mixture models

Bayesian non-parametric modelling allows us to overcome the limitations of parametric mixture models, particularly the assumption of a fixed and known number of clusters. Dirichlet Process Mixture Models extend mixture models to BNP models. In this framework, the number of clusters is assumed to be infinite, shifting the problem from a parametric to a non-parametric, and the model parameters are assumed to be random and follow a prior distribution, shifting the problem from a frequentist estimation to a Bayesian estimation. This framework allows the estimation of the number of clusters during the inference process.

DPMMs have been widely applied across diverse domains. For instance, [177] introduced Bayesian parametric multinomial mixture models applied to short text clustering, which was then extended to a non-parametric setting in [179]. In the medical field, DPMMs have been used for brain tumor segmentation in imaging [107] and for clustering single-cell RNA-seq data [3]. Beyond biomedicine, [25] have used DPMMs to model user preferences within content-based recommender systems.

In this section, we first recall the Chinese Restaurant Process and the stick-breaking process, which are the two most widely used representations of the Dirichlet Process. We then present the formulation of the Dirichlet Process

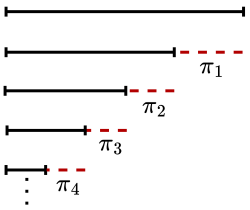

Figure 1.7: Representation of the Stick-Breaking process

Mixture Model. Finally, we describe the inference procedure based on the collapsed Gibbs sampling algorithm.

1.4.1 . Preliminaries

Stick breaking process

We let $\alpha > 0$ and $\pi = (\pi_k)_{k=1}^{\infty}$ an infinite probability vector derived from the following process:

$$v_k \overset{\text{i.i.d.}}{\sim} \text{Beta}\,(1, \alpha)\,,$$
$$\pi_k = v_k \prod_{k'=1}^{k-1} (1 - v_{k'})\,.$$

π is said to follow the *Stick-Breaking* process (SB), also called the *Griffiths-Engen-McCloskey* (GEM) process, parameterized by $\alpha > 0$. This is denoted by $\pi \sim SB(\alpha)$ or $\pi \sim GEM(\alpha)$. We have $\pi_k > 0$ for every $k > 1$ and

$$\sum_{k=1}^{\infty} \pi_k = 1.$$

With this definition, when $\boldsymbol{\theta} = (\theta_k)_{k \geq 1}$ are sampled i.i.d. according to some distribution G_0, the distribution

$$G(\boldsymbol{\theta}) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}(\boldsymbol{\theta}),$$

where δ is the indicator function, follows the Dirichlet Process with a *concentration parameter* α and a *base distribution* G_0 [150]. This is denoted by

$$G \sim DP(\alpha, G_0).$$

An illustration of the stick-breaking process is given in Figure 1.7.

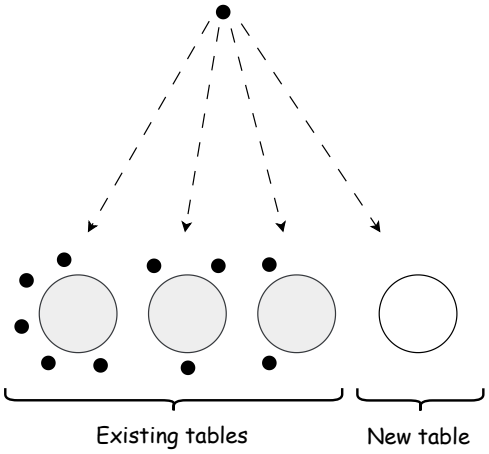

Figure 1.8: Representation of the Chinese Restaurant Process

Chinese restaurant process

The Chinese Restaurant Process (CRP) [10] defines a distribution over the set of partitions that serves as an intuitive metaphor (see Figure 1.8) for representing the clustering structure in Dirichlet Process Mixture Models. Consider a restaurant with an infinite number of circular tables (representing clusters) and customers (observations) entering one by one. Each customer chooses a table either by joining an existing table, with probability proportional to the number of customers already there, or by selecting a new one with probability that is proportional to a given concentration parameter $\alpha > 0$.

Formally, let n denote the number of customers currently seated, and let n_k be the number of customers at table k. The cluster assignment (i.e., table choice) for customer i is denoted by z_i. The probability of customer i choosing a particular table under the *Chinese Restaurant Process* is given by:

$$\mathrm{p}(z_i = k \mid \mathbf{z}_{-i}, \alpha) \propto \begin{cases} \frac{n_k}{n-1+\alpha} & \text{if } k \text{ is an existing cluster,} \\ \frac{\alpha}{n-1+\alpha} & \text{if } k \text{ is a new cluster,} \end{cases}$$

where $\mathbf{z}_{-i}$ denotes the assignments of all other customers. The concentration parameter α controls the tendency to create new clusters. On the other hand, already occupied tables are more likely to attract other customers. This fact is known as the *"rich get richer"* phenomenon. This process defines a distribution over all possible partitions of n elements. If the resulting partition contains K clusters with cardinalities $(n_1, \ldots, n_K)$, the joint distribution of $\mathbf{z} = (z_1, \ldots, z_n)$ under the CRP is:

$$\mathrm{p}(z_1, \ldots, z_n) = \frac{\alpha^K}{\prod_{i=1}^{n}(\alpha + i - 1)} \prod_{k=1}^{K}(n_k - 1)!$$

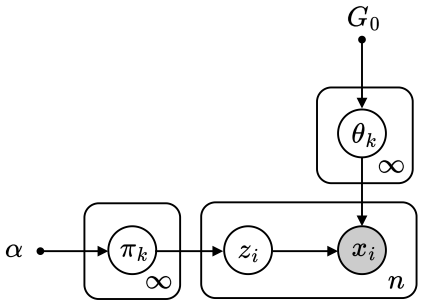

Figure 1.9: Graphical representation of a Dirichlet Process Mixture Model.

The expected number of clusters observed under the CRP grows approximately as $\alpha \log(n)$ as the number of observations $n \to \infty$.

The Chinese Restaurant Process is also related to the stick-breaking process, as shown in [115]; each process can be derived from the other.

1.4.2 . Model definition

Given the observed dataset $\mathbf{x} = (x_1, \ldots, x_n)^T$ and the corresponding membership vector $\mathbf{z} = (z_1, \ldots, z_n)$. The *Dirichlet Process Mixture Model* is defined as follows:

$$
\begin{aligned}
&\pi \sim SB(\alpha), \\
&z_i \overset{\text{i.i.d.}}{\sim} \text{Mult}\left(\pi\right), && \forall i \in \{1, \ldots, n\}, \\
&\theta_k \overset{\text{i.i.d.}}{\sim} G_0, && \forall k \in \{1, 2, \ldots\}, \\
&x_i \mid \{z_i = k, \theta_k\} \overset{\text{i.i.d.}}{\sim} f(x_i, \theta_k), && \forall i \in \{1, \ldots, n\}.
\end{aligned}
$$

This definition assumes a generative process over the observation space. To generate an observation x_i, we first generate an infinite vector of mixture weights π, also called mixture proportions, following the stick-breaking process. Then, we draw z_i from the multinomial distribution parameterized by π. Finally, given an unbounded number of parameters $\boldsymbol{\theta} = (\theta_k)_{k \geq 1}$ generated from the base distribution G_0 (prior), the observation $x_i | \{z_i = k, \theta_k\}$ that belongs to the cluster k is drawn from the component distribution $f(\cdot, \theta_k)$. Here, the component parameters $\theta_k \in \Theta$ are unknown. We denote by $\Omega = (\alpha, G_0)$ the hyper-parameter set. We assume that G_0 is the conjugate prior on θ_k. The representation of Dirichlet Process mixture models is illustrated in Figure 1.9.

1.4.3 . Inference

Given a set of observations $\mathbf{x}$, the goal is to infer the membership vector $\mathbf{z}$, which represents the cluster assignments. Formally, we want to sample from the joint posterior distribution $\mathrm{p}(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\theta}, \Omega)$, which represents the target density. However, direct sampling from this distribution is generally intractable.

To address this, two main classes of approximate inference methods are typically used: Markov Chain Monte Carlo methods [121] and variational inference [19].

MCMC methods provide asymptotically exact samples from the target distribution [138], but they can be computationally intensive. In contrast, variational inference does not provide such guarantees and instead finds a density that approximates the target [20]. A widely used MCMC method is the Gibbs sampling algorithm [121], which iteratively updates the membership vector and parameters associated with each cluster.

When the model satisfies conjugacy assumptions, the parameters can be integrated out analytically, leading to the collapsed Gibbs sampler [122], a more efficient variant of standard Gibbs sampling. In this section, we focus on recalling this approach.

The standard Gibbs sampler (Algorithm 2 in [122]) alternates between two steps. First, it samples each membership variable z_i from the conditional distribution $\mathrm{p}(z_i \mid \mathbf{x}, \mathbf{z}_{-i}, \boldsymbol{\theta}, \Omega)$, where $\mathbf{z}_{-i} = \{z_l : l \neq i\}$ denotes the current assignment of all other observations. Second, it updates the parameters of each cluster θ_k by sampling from the posterior distribution $\mathrm{p}(\theta_k \mid \mathbf{x}_k, G_0)$, where $\mathbf{x}_k = \{x_i : z_i = k\}$ is the set of data points assigned to cluster k.

In the *collapsed Gibbs sampler* (Algorithm 3 in [122]), the model parameters θ_k are integrated out, skipping the parameter sampling step. This results in a more efficient algorithm that samples each membership variable z_i directly from its marginal

$$\mathrm{p}(z_i \mid \mathbf{z}_{-i}, \mathbf{x}, \Omega) \propto \begin{cases} \dfrac{n_k}{n-1+\alpha}\mathrm{p}(x_i \mid z_i = k, \mathbf{x}_k, G_0), & \text{existing cluster } k, \quad (1.8) \\ \dfrac{\alpha}{n-1+\alpha}\mathrm{p}(x_i \mid G_0), & \text{new cluster}, \quad (1.9) \end{cases}$$

where n_k is the cardinality of cluster k. The posterior predictive Equation 1.8 and prior predictive Equation 1.9 can be obtained by integrating over θ:

$$\begin{aligned} \mathrm{p}(x_i \mid z_i = k, \mathbf{x}_k, G_0) &= \int_\Theta \mathrm{p}(x_i, \theta \mid \mathbf{x}_k, G_0) d\theta, \\ &= \int_\Theta \mathrm{p}(x_i \mid \theta)\mathrm{p}(\theta \mid \mathbf{x}_k, G_0) d\theta, \end{aligned} \quad (1.10)$$

and

$$\begin{aligned} \mathrm{p}(x_i \mid \Omega) &= \int_\Theta \mathrm{p}(x_i, \theta \mid G_0) d\theta, \\ &= \int_\Theta \mathrm{p}(x_i \mid \theta)\mathrm{p}(\theta \mid G_0) d\theta. \end{aligned} \quad (1.11)$$

Under the conjugacy assumption, the integrals in Equations 1.10 and 1.11 can be computed analytically. In Chapters 2 and 3, we detail the computation of these terms when the component distributions are assumed to be

multivariate Gaussian and multinomial, with the associated priors being the Normal–Inverse–Wishart and Dirichlet distributions, respectively.

The collapsed Gibbs sampler is an efficient Markov Chain Monte Carlo method that improves sampling efficiency by integrating out the component parameters. This technique, known as Rao-Blackwellization, is based on the Rao-Blackwell theorem [18], which guarantees that the variance of an estimator obtained by marginalizing over latent variables (such as $\boldsymbol{\theta}$) is always less than or equal to that of an estimator based on direct sampling. This property holds in the context of collapsed Gibbs sampling as well [104]. The complete inference procedure is summarized in Algorithm 3.

Algorithm 3 Collapsed Gibbs sampler for DPMM inference

Input: Dataset $\mathbf{x}$,
    Concentration parameter α,
    Prior G_0,
    Iteration number M.
For $m \leftarrow 1$ **to** M **do**:
    For $i \leftarrow 1$ **to** n **do**:
        Remove x_i from its cluster.
        Compute $\mathrm{p}(z_i \mid \mathbf{z}_{-i}, \mathbf{x}, \Omega)$. ▷ *Using Equations 1.8 and 1.9*
        Sample z_i. ▷ *Draw new membership*
        Add x_i to its new cluster.
Output: The last sampled partition $\mathbf{z}$.

1.5 . Bayesian Non-parametric Latent Block Models

Given a data matrix, where rows represent observations and columns represent variables or features, co-clustering aims to infer a row partition and a column partition simultaneously. The resulting partition is composed of homogeneous blocks (see Figure 1.10). When a dataset exhibits a dual structure between observations and variables, co-clustering outperforms conventional clustering algorithms, which only infer a row partition without considering the relationships between observations and variables. Co-clustering is a powerful data mining tool for two-dimensional data and is widely applied in various fields such as document–term coclustering, recommender systems [73], Bioinformatics [70, 92], and ADAS systems[63]. For a comprehensive overview of existing methods and applications, refer to the following survey [17, 163].

1.5.1 . Model definition

Let n, p, and d be positive integers, and let $X = (x_{i,j})_{n,p} \in \mathbb{R}^{n \times p \times d}$ be the observed dataset. Here, n represents the number of rows, p is the number

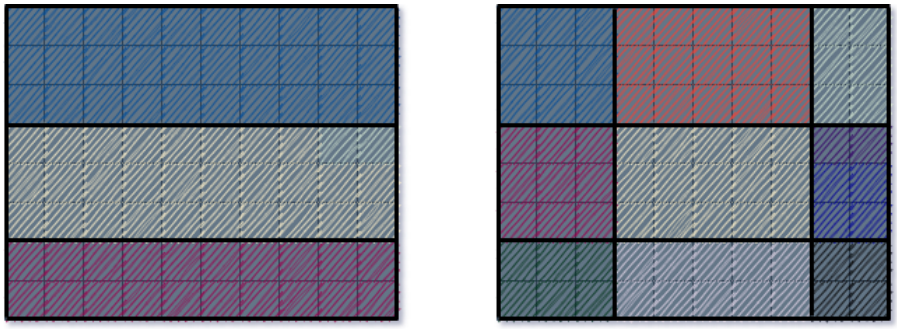

Figure 1.10: Clustering (left) and co-clustering (right)

of columns, and d denotes the dimension of the observation space. Let $\mathbf{z} = (z_i)_{i=1}^n$ be the *row membership* vector (also called *row partition*), where each z_i is a latent variable such that $z_i = k$ means that the i-th row $x_{i,\cdot}$ belongs to the row cluster k. Similarly, let $\mathbf{w} = (w_j)_{j=1}^p$ be the *column membership* vector (also called *column partition*), where $w_j = l$ indicates that the j-th column $x_{\cdot,j}$ belongs to the column cluster l. The *Non-Parametric Latent Bloc Model* is defined as follows:

$$\begin{gathered}\pi \sim \mathrm{SB}(\alpha),\ \rho \sim \mathrm{SB}(\beta),\\ \theta_{z_i,w_j} \sim G_0,\ z_i \,|\pi \sim \mathrm{Mult}(\pi),\ w_j|\, \rho \sim \mathrm{Mult}(\rho),\\ x_{i,j} \mid \{z_i, w_j, \theta_{z_i,w_j}\} \sim f\left(x_{i,j}, \theta_{z_i,w_j}\right).\end{gathered}$$

According to this definition, each observation $x_{i,j}$ is generated through the following process. First, *row proportions* $\boldsymbol{\pi}$ and *column proportions* $\boldsymbol{\rho}$ are drawn independently from Stick-Breaking processes parameterized by concentration parameters $\alpha > 0$ and $\beta > 0$, respectively. Next, the row and column membership vectors $\mathbf{z}$ and $\mathbf{w}$ are sampled from multinomial distributions parametrized by $\boldsymbol{\pi}$ and $\boldsymbol{\rho}$, respectively. Given the block component parameters θ_{z_i,w_j} sampled from the base distribution G_0, the cell $x_{i,j} \mid \{z_i, w_j, \theta_{z_i,w_j}\}$ is drawn from $f(x_{i,j}, \theta_{z_i,w_j})$. A graphical representation of this Bayesian non-parametric latent block model is shown in Figure 1.11.

1.5.2 . Inference

The goal is to estimate the row and column partitions $\mathbf{z}$ and $\mathbf{w}$ given the observed dataset X, the prior G_0, and the concentration parameters α and β, by sampling from the joint posterior distribution $\mathrm{p}(\mathbf{w}, \mathbf{z} \mid X, G_0, \alpha, \beta)$. Direct sampling from this distribution is intractable, but it can be performed using the collapsed Gibbs sampler.

Given initial values for the row and column memberships, the inference procedure alternates between updating the row partition given the current column partition and vice versa. For instance, the update of the row partition $\mathbf{z}$ is performed by sampling each z_i sequentially from its conditional posterior

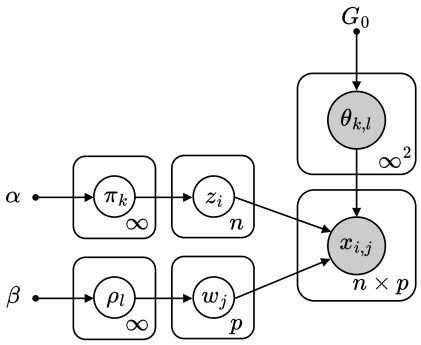

Figure 1.11: Graphical representation of a Bayesian Non-parametric Latent Block Model.

distribution $\mathrm{p}(z_i \mid \mathbf{z}_{-i}, \mathbf{w}, X, G_0, \alpha) \propto$

$$\begin{cases} \dfrac{n_k}{n-1+\alpha}\, \mathrm{p}\left(\mathbf{x}_{i,\cdot} \mid \mathbf{w}, X_{-i}, \mathbf{z}_{-i}, G_0\right), & \text{existing row cluster } k, \quad (1.12) \\ \dfrac{\alpha}{n-1+\alpha}\, \mathrm{p}\left(\mathbf{x}_{i,\cdot} \mid \mathbf{w}, G_0\right), & \text{new row cluster.} \quad (1.13) \end{cases}$$

Here, $\mathbf{z}_{-i} = \{z_r : r \neq i\}$ denotes the row partition excluding observation i, and n_k is the number of rows currently assigned to cluster k. The joint posterior predictive distribution term in Equation 1.12 is given by

$$\mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}, X_{-i}, \mathbf{z}_{-i}, G_0) = \prod_{l=1}^{L} \mathrm{p}(\mathbf{x}_{i,l} \mid G_{k,l}), \quad (1.14)$$

where L is the number of inferred column clusters, $\mathbf{x}_{i,l} = \{x_{i,j} : w_j = l\}$ denotes the subset of elements in row i associated with column cluster l, and $G_{k,l}$ represents the block posterior distribution updated from the prior. Finally, the joint prior predictive distribution in Equation 1.13 is expressed as:

$$\mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}, G_0) = \prod_{l=1}^{L} \mathrm{p}(\mathbf{x}_{i,l} \mid G_0). \quad (1.15)$$

When the conjugacy assumption holds, the terms $\mathrm{p}(\mathbf{x}_{i,l} \mid G_{k,l})$ and $\mathrm{p}(\mathbf{x}_{i,l} \mid G_0)$ required in Equations (1.14) and (1.15) admit closed-form expressions. In Chapter 4, we detail the computation of these terms in the case where the block component distribution is multivariate Gaussian and the prior is the conjugate Normal–Inverse–Wishart distribution. The update for the column partition $\mathbf{w}$ is carried out analogously, conditioned on the current row partition. The complete inference algorithm for the NPLBM is provided in Algorithm 4.

Algorithm 4 Collapsed Gibbs Sampler for NPLBM Inference

Input: Dataset X,
    Concentration parameters α (rows) and β (columns),
    Prior distribution G_0,
    Number of iterations M.
for $m = 1$ to M **do**
  for $i = 1$ to n **do** ▷ *Update row cluster assignments*
    Remove row $\mathbf{x}_{i,\cdot}$ from its current row cluster.
    Compute $\mathrm{p}(z_i \mid \mathbf{z}_{-i}, X, \mathbf{w}^{(m)}, \alpha, G_0)$. ▷ *Using Equations 1.13 and 1.12*
    Sample a new assignment $z_i^{(m+1)}$.
    Add row $\mathbf{x}_{i,\cdot}$ to the new row cluster.
  for $j = 1$ to p **do** ▷ *Update column cluster assignments*
    Remove column $\mathbf{x}_{\cdot,j}$ from its current column cluster.
    Compute $\mathrm{p}(w_j \mid \mathbf{w}_{-j}, X, \mathbf{z}^{(m)}, \beta, G_0)$. ▷ *Computed similarly to the conditional distribution for the rows*
    Sample a new assignment $w_j^{(m+1)}$.
    Add column $\mathbf{x}_{\cdot,j}$ to the new column cluster.
Output: Estimated block partition $(\hat{\mathbf{z}}, \hat{\mathbf{w}})$

1.6 . Parallel and distributed inference

DPMMs and NPLBMs can automatically estimate the number of clusters and co-clusters, respectively, and identify new structures. However, their inference process through MCMC methods becomes prohibitively slow and non-scalable with large datasets. This scalability issue limits its practical use in handling extensive data in real-world applications.

Distributed computing consists of distributing data evenly across multiple workers. The workers may represent components in edge devices, cores in a cluster, computers in a network, processors in a computer, etc. This distribution allows the execution of parallel computations. Distributed computing presents a good framework to significantly accelerate computations and circumvent memory limitations, making it particularly suitable for algorithms to handle large datasets.

In this section, we first present the most widely used parallel and distributed frameworks for big data learning. We then review existing approaches in the literature for parallel and distributed inference for Dirichlet Process Mixture Models and co-clustering.

1.6.1 . Distributed framework

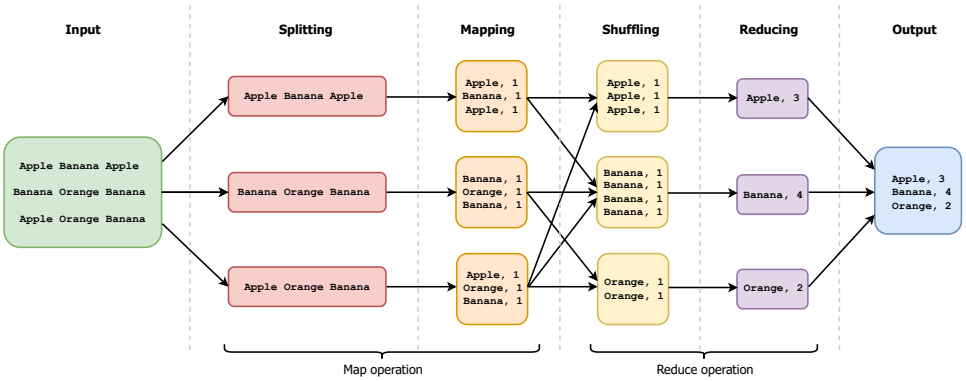

Figure 1.12: Workflow of the MapReduce Process for Word Count

Distributed computing frameworks are a fundamental component of distributed computing systems. Many parallel and distributed frameworks have been proposed to allow the implementation and execution of parallel and distributed algorithms that can handle large datasets. The most famous ones are MapReduce [42], Spark [184], Storm [158], and Flink [28]. Next, we describe the MapReduce and Spark frameworks.

MapReduce

The MapReduce programming model [42] facilitates large-scale data processing by splitting computations into independent tasks that operate on distributed subsets of the data. Initially introduced by Google, it was later adopted by the Apache Hadoop project [166], which enabled developers to adapt sequential algorithms for parallel execution across cluster environments.

A MapReduce program is composed of two main operations: Map and Reduce. During the Map operation, each node processes its assigned data blocks independently to produce local results, typically in the form of key-value pairs. These operations are run in parallel across all participating nodes, enhancing computational performance. After mapping is complete, the intermediate data is shuffled and routed to specific nodes for the Reduce operation. The Reduce phase aggregates the local results to produce the final output (as illustrated in Figure 1.12). This entire workflow is managed under a master-worker architecture, where the master coordinates the job execution, and the workers execute the Map and Reduce tasks in parallel.

Spark

Researchers from the University of California introduced Apache Spark [184], a cluster computing framework designed as an extension of the MapReduce model to support more complex workloads, including interactive queries and

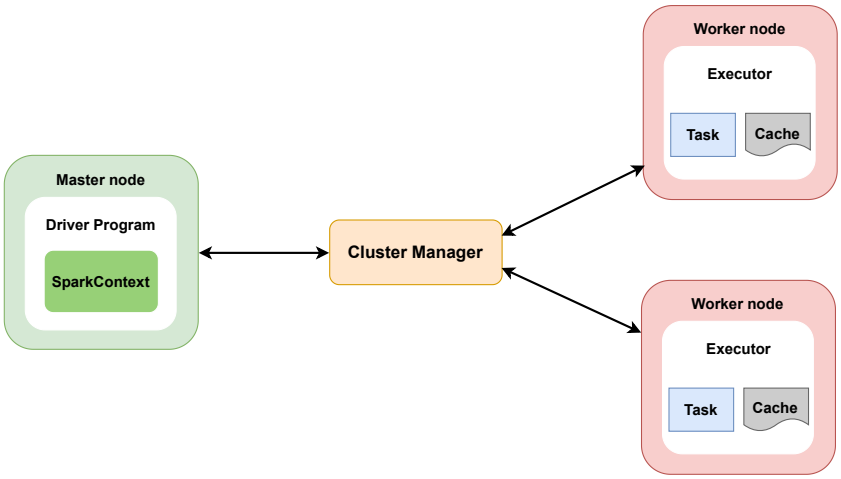

Figure 1.13: Spark architecture

stream processing. A central abstraction in Spark is the Resilient Distributed Dataset (RDD) [183], a fault-tolerant, read-only collection of objects distributed across multiple nodes. If any partition of an RDD is lost, it can be recomputed automatically using lineage information. Spark supports parallel computation through two types of operations: Transformations (e.g., map, filter) and Actions (e.g., reduce, collect, foreach), where transformations are operations applied to RDDs that produce new RDDs, and Actions perform computations on RDDs and return the final result to the driver program.

Spark is implemented in the high-level programming language Scala[1], and allows the construction of RDDs from various sources, such as the Hadoop Distributed File System (HDFS) [152] or parallelized Scala collections. The framework follows a master/worker architecture, where a central component called the driver orchestrates the computation and communicates with multiple distributed executors, as illustrated in Figure1.13. Spark applications can be launched using various cluster managers, including the Standalone cluster manager, Hadoop YARN[2], Apache Mesos[3], and Kubernetes[4].

1.6.2 . Parallel and distributed inference for DPMM

Parallel inference

Several parallel approaches have been proposed to address the slow inference of DPMMs. In [106], the authors introduced a reparameterization of the Dirichlet Process that facilitates the learning of clusters representing the

[1] https://www.scala-lang.org/
[2] https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html
[3] https://mesos.apache.org/
[4] https://kubernetes.io/

data and super-clusters defining the granularity of parallelization. In [167], the authors incorporated auxiliary variables into the Dirichlet Process and Hierarchical Dirichlet Process to establish a conditional independence structure, enabling a parallel Gibbs sampler without the need for approximations. However, [57] demonstrated that the auxiliary variable approach proposed in these works leads to an extremely unbalanced distribution of data across computing nodes during parallel inference, making these methods impractical for real-world applications. Another parallelized MCMC sampler was proposed by [31], which combines a non-ergodic, restricted Gibbs sampler in the workers with split/merge proposals [81] at the master level to ensure ergodicity. However, this method was implemented only in a single-machine setting, which limits its scalability and does not fully leverage distributed memory resources.

Distributed Inference

In [46], the authors proposed to extend the parallel approach proposed in [31] to a distributed version across multiple machines using a distributed-memory model. Despite this improvement, the main limitation of this approach is the use of a restricted Gibbs sampler at the worker level, where the number of clusters is fixed. As new components can only be discovered at the master level, convergence becomes significantly slower. In fact, many iterations are needed before the model discovers a sufficient number of clusters to capture the underlying data distribution. In [59], the authors proposed a distributed inference algorithm for DPMMs based on a slice sampler developed by [162]. They addressed the scalability by developing a distributed architecture that estimates the Dirichlet Process and Hierarchical Dirichlet Process mixture models. [164] presented a scalable estimation method for DPMMs in a distributed architecture. Their approach allows the creation of new components locally in individual computing nodes and proposes a probabilistic consolidation scheme for merging the created components. In the [114], the authors introduced another distributed Markov Chain Monte Carlo inference method based on the Gibbs sampler. Their approach performs a local Gibbs sampler to infer new local clusters and a new Gibbs sampler on the means of each cluster to synchronize the clustering. However, this method assumes a Dirichlet Process Gaussian Mixture Model with known variances. This assumption of fixed variances both within and between clusters imposed by the prior significantly restricts the model's flexibility and limits its applicability to real-world scenarios where variances are typically unknown and must be inferred from the data. In addition, assuming known variance of each cluster implicitly assumes the knowledge of the number of clusters, which makes one of the key advantages of the DPMM, its ability to estimate the number of clusters, meaningless or even obsolete.

1.6.3 . Parallel and distributed inference for NPLBM

Numerous scalable co-clustering algorithms have been proposed in the literature. The first distributed co-clustering (DisCo) using Hadoop is introduced in [128]. In [54], authors devised a parallelized co-clustering approach, specifically designed to tackle the high-order co-clustering problem with heterogeneous data. Their methodology extends the approach initially proposed in [66], enabling the computation of co-clustering solutions in a parallel fashion, leveraging a Map-Reduce infrastructure. In [45], a parallel simultaneous co-clustering and learning (SCOAL) approach is introduced, also harnessing the power of Map-Reduce. This work focuses on predictive modeling for bi-modal data. In [36], the authors introduce a distributed framework for data co-clustering with sequential updates (Co-ClusterD). The authors propose two distinct approaches to parallelize sequential updates for alternate minimization co-clustering algorithms. However, it's worth noting that these approaches are parametric and assume knowing a priori the true numbers of row and column clusters, respectively, which are unknowable in real-life applications. One of the main challenges in distributing Bayesian Non-Parametric co-clustering lies in efficiently handling and discovering new block components.

1.7 . Evaluation

1.7.1 . Clustering quality

Evaluating the quality of the output produced by a clustering algorithm is a challenging problem, as the clustering solution is not unique. While probabilistic models allow the evaluation of likelihood, this metric does not necessarily reflect the quality of the inferred clustering structure. In addition, likelihood evaluation is limited only to probabilistic models. To address this, various evaluation criteria have been proposed. They are mainly grouped into two main categories:

1. **Internal criteria**, which do not require ground-truth labels and instead rely on intrinsic properties of the data (e.g., compactness and separation between clusters). Common examples include the Dunn Index [49], Calinski-Harabasz Index [24], Davies-Bouldin Index [41], and the Silhouette Coefficient [144].

2. **External criteria,** which assume access to ground-truth labels. These metrics compare the inferred clustering labels against the true labels. Widely used examples include Purity [137], Adjusted Rand Index (ARI) [131], Normalized Mutual Information (NMI) [7], Adjusted Mutual Information (AMI) [161], Clustering Accuracy (ACC) [56], and Variation of Information (VI) [143].

In this manuscript, we mainly use ARI, NMI, and ACC to evaluate and compare clustering methods. We provide a brief description of these metrics in what follows.

Rand Index

Let $\hat{\boldsymbol{P}} = \{\hat{P}_1, \ldots, \hat{P}_{\hat{K}}\}$ denote the estimated clustering partition induced by the estimated membership vector $\hat{\mathbf{z}}$, and $\boldsymbol{P} = \{P_1, \ldots, P_K\}$ the reference (ground-truth) partition derived from the true labels $\mathbf{z}$. The Rand Index (RI) [131] measures the similarity between the two partitions and is defined as:

$$RI(\hat{\boldsymbol{P}}, \boldsymbol{P}) := \frac{TP + TN}{TP + FP + FN + TN},$$

where TP is the number of True Positives. TN is the number of True Negatives. FP is the number of False Positives. FN is the number of False Negatives. The RI ranges between 0 and 1, with higher values indicating better clustering quality. However, the major drawback of the RI score is that even if $\hat{\boldsymbol{P}}$ is significantly different from $\boldsymbol{P}$, the RI score doesn't guarantee to be close to 0.

Adjusted Rand Index

To overcome the limitations of the RI, the Adjusted Rand Index adjusts for chance by subtracting the expected similarity of random clusterings and normalizing by the maximum possible RI. It is defined as:

$$ARI(\hat{\boldsymbol{P}}, \boldsymbol{P}) = \frac{RI(\hat{\boldsymbol{P}}, \boldsymbol{P}) - \mathbb{E}[RI(\hat{\boldsymbol{P}}, \boldsymbol{P})]}{\max(RI(\hat{\boldsymbol{P}}, \boldsymbol{P})) - \mathbb{E}[RI(\hat{\boldsymbol{P}}, \boldsymbol{P})]}.$$

The ARI ranges from -1 to 1. An ARI close to 0 indicates random clustering, while 1 indicates a perfect match.

Normalized Mutual Information

The Mutual Information (MI) [39] measures the amount of shared information between two partitions. Given $\hat{\boldsymbol{P}}$ and $\boldsymbol{P}$, the MI is defined as:

$$MI(\hat{\boldsymbol{P}}, \boldsymbol{P}) = \sum_{i=1}^{\hat{K}} \sum_{j=1}^{K} P(i,j) \log \left(\frac{P(i,j)}{\hat{P}(i) P(j)} \right),$$

where $P(i,j)$ is the probability that a randomly chosen data point belongs to both $\hat{P}_i$ and P_j; $\hat{P}(i)$ and $P(j)$ are the marginal probabilities for the estimated and true clusters, respectively.

The Normalized Mutual Information is a normalized version of MI and is defined as:

$$NMI(\hat{\boldsymbol{P}}, \boldsymbol{P}) = \frac{MI(\hat{\boldsymbol{P}}, \boldsymbol{P})}{\left(\mathbb{H}(\hat{\boldsymbol{P}}) + \mathbb{H}(\boldsymbol{P})\right)/2},$$

where $\mathbb{H}(\hat{\boldsymbol{P}})$ and $\mathbb{H}(\boldsymbol{P})$ denote the entropies of the predicted and true partitions, respectively. The entropy of a partition $\boldsymbol{P}$ is given by:

$$\mathbb{H}(\boldsymbol{P}) = -\sum_{j=1}^{K} P(j) \log P(j).$$

The NMI score ranges between 0 and 1, with higher values indicating greater similarity between the partitions.

Clustering Accuracy

Clustering Accuracy [56] is a set-matching-based metric that compares predicted and true labels after finding the optimal one-to-one label correspondence. Let $\hat{\boldsymbol{z}}$ and $\boldsymbol{z}$ be the membership vectors corresponding to $\hat{\boldsymbol{P}}$ and $\boldsymbol{P}$, respectively. The ACC is defined as:

$$ACC(\hat{\boldsymbol{P}}, \boldsymbol{P}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{z_i = \mathrm{map}(\hat{z}_i)\}},$$

where map is a permutation function that maps the predicted labels to the true labels. In other words, clustering accuracy measures the proportion of data points that are correctly assigned to their corresponding clusters, up to a permutation of labels. Since clustering algorithms assign arbitrary labels to clusters, we seek an optimal mapping function map that permutes the predicted labels to best align with the ground truth labels. The optimal mapping is typically obtained using the Kuhn-Munkres (Hungarian) algorithm [127]. ACC ranges from 0 to 1, with higher values indicating a better match.

1.7.2 . Stability

In practical applications, it is essential that clustering results remain stable under small perturbations to the input data. Inconsistent outcomes can lead to unreliable interpretations and potentially costly errors, emphasizing the need for robust and stable clustering methods. Average sensitivity, introduced by Murai and Yoshida et al. [116], formally quantifies an algorithm's stability with respect to small changes in its input. Average sensitivity was introduced to analyze the stability of network centralities and was later formally defined for (graph) optimization problems by Varma and Yoshida [159]. It can also be adapted to evaluate the stability of clustering algorithms. We now define average sensitivity in the context of clustering.

For a dataset $\mathbf{x} = \{x_1, \ldots, x_n\}$, let $\mathbf{x}_{-i}$ denote the dataset obtained by removing the observation x_i (i.e., $\mathbf{x} \setminus \{x_i\}$). For a deterministic algorithm A, let $A(\mathbf{x})$ denote the clustering output when A is run on $\mathbf{x}$. The *average sensitivity* of A on $\mathbf{x}$ is defined as:

$$\frac{1}{n} \sum_{i=1}^{n} \mathsf{d}(A(\mathbf{x}), A(\mathbf{x}_{-i})),$$

where $\mathsf{d}(\cdot, \cdot)$ is a distance that quantifies the dissimilarity between two clustering outputs. An algorithm is considered stable on average if its average sensitivity is small.

The stability of some clustering algorithms has been specifically investigated. For instance, [129] analyzed the average sensitivity of spectral clustering. Yoshida [180] studied the average sensitivity of k-clustering methods, including the k-means algorithm. However, in this work, the authors analyzed the D^2-sampling procedure used in the initialization step of the k-means++ algorithm [12]. Average sensitivity has been studied in other various contexts, including dynamic programming [95, 96], decision tree learning [71], and graph problems [181]. In Chapter 5 of this manuscript, we will analyze the average sensitivity of the Expectation-Maximization algorithm.

2 - Distributed inference for DPMMs with a Multivariate Gaussian components

In this chapter, we present a distributed inference algorithm for Dirichlet Process Mixture Models using sufficient statistics. We assume that the distribution associated with each cluster is a multivariate Gaussian. The proposed method is inspired by the collapsed Gibbs sampling approach introduced in [51, 122]. Its main objective is to improve the scalability of the inference process while preserving the accuracy of Markov Chain Monte Carlo techniques and the flexibility offered by DPMMs. This algorithm also serves as a foundational component for distributing the inference process in Bayesian nonparametric latent block models, as discussed in Chapter 4. The research presented in this chapter was originally published in [89].

2.1 . Introduction

The Dirichlet Process Mixture Model is a flexible Bayesian non-parametric framework for clustering that automatically infers the number of clusters during the inference process. While traditional inference methods based on MCMC, such as collapsed Gibbs sampling [51, 122], offer high accuracy, they suffer from high computational cost and poor scalability when applied to large datasets.

To address these limitations, we propose a novel distributed inference method based on sufficient statistics that significantly improves computational efficiency and reduces execution time, while maintaining high clustering accuracy. The idea of using sufficient statistics for distributed DPMM inference has been explored in several works [59, 164, 114, 46]. However, to the best of our knowledge, it has not yet been applied to distribute the Collapsed Gibbs Sampler , which is the core focus of this work. Moreover, none of the existing approaches is readily applicable for improving the scalability and distribution of Bayesian non-parametric co-clustering methods [113].

We demonstrate the effectiveness of this approach on both synthetic and real-world datasets. For instance, with a dataset of 100000 observations, the centralized algorithm requires approximately 12 hours to complete 100 iterations, while our approach achieves the same number of iterations in just 3 minutes, reducing the execution time by a factor of 200 without compromising clustering performance.

The remainder of this chapter is organized as follows. In Section 2.2, we review the definition of the Dirichlet Process Mixture Model with Gaussian component distributions. In Section 2.3, we present our distributed inference

algorithm, including the collapsed Gibbs sampler performed at the worker and master levels. Section 2.4 reports empirical results on synthetic and real-world datasets. Finally, Section 2.5 offers concluding remarks and directions for future work.

2.2 . Model definition

We assume that the observations that belong to the same cluster are independent and identically distributed according to a multivariate Gaussian distribution. More specifically, for a cluster k, we denote the parameters associated with its component distribution by $\theta_k = (\mu_k, \Sigma_k)$ where $\mu_k \in \mathbb{R}^d$ and $\Sigma_k \in \mathbb{R}^{d\times d}$ is a positive semi-definite matrix. We assume that the component distribution associated with the cluster k is given by

$$f(x_i, \theta_k) = \mathcal{N}(x_i \mid \mu_k, \Sigma_k).$$

Here, μ_k and Σ_k are unknown. Hence, a natural conjugate prior is the Normal-Inverse Wishart (NIW) distribution parameterized by $\lambda_0 = (\mu_0, \kappa_0, \Psi_0, \nu_0)$. The parameter μ_0 can be interpreted as our prior for μ, and κ_0 is how confident we are in this prior. Similarly, Ψ_0 is our prior on Σ, and ν_0 is how confident we are in this prior. Under these assumptions, the Dirichlet Process Mixture Model can be formally defined as follows:

$$\begin{aligned}
&\pi \sim SB(\alpha), \\
&z_i \overset{\text{i.i.d.}}{\sim} \mathrm{Mult}\,(\pi)\,, && \forall i \in \{1, \cdots, n\}, \\
&\theta_k \overset{\text{i.i.d.}}{\sim} \mathrm{NIW}(\lambda_0), && \forall k \in \{1, 2, \cdots\}, \\
&x_i \mid \{z_i = k, \theta_k\} \overset{\text{i.i.d.}}{\sim} \mathcal{N}\,(\mu_k, \Sigma_k)\,, && \forall i \in \{1, \cdots, n\}.
\end{aligned}$$

As stated in 1.6, the inference process of DPMM parameters using the collapsed Gibbs sampler is an efficient MCMC method because it avoids component parameter sampling and provides high accuracy. However, this approach still becomes considerably slow and fails to scale when the number of observations is large, especially in the Gaussian case, due to the intensive computations involved. The following sections introduce our novel distributed inference approach designed to overcome this limitation.

2.3 . Distributed inference

The collapsed version of the Gibbs sampling inference inspires our approach and is based on a Master/Worker architecture. In this approach, the data is evenly partitioned across multiple workers to ensure a balanced workload. Each worker performs independently a collapsed Gibbs sampler on its

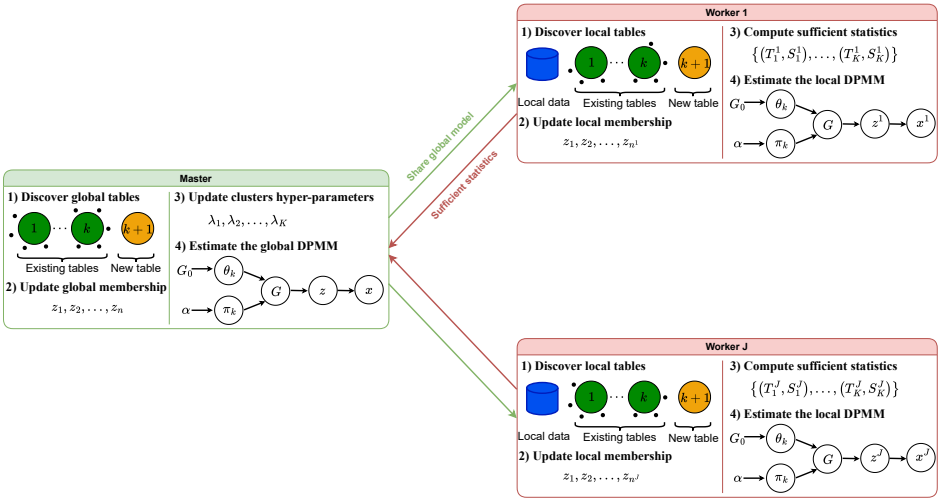

Figure 2.1: General workflow of DisCGS

local data to discover local clusters. Then, the sufficient statistics associated with each cluster are computed and sent to the master node.

At the master level, the global clustering structure is inferred by only using sufficient statistics. The master aggregates local information to estimate a global DPMM and updates the hyperparameters associated with each global cluster. The global clustering structure is then shared with the workers. It is important to note that the workers do not exchange information between them, and the master never accesses the raw local data, but only the aggregated statistics. More specifically, our method alternates between these two steps:

1 **Worker level:** In this step, a local DPMM is inferred on each worker using the collapsed Gibbs sampler. The observations inside each worker are assigned one by one to their local clusters. This identifies local clusters. After sufficient statistics and sizes associated with each cluster are sent to the master.

2 **Master level:** At this level, instead of assigning the observations one by one to their global cluster, we assign a group of observations that already share the same local cluster at the worker level to a global cluster at the master level. Then, all the observations with the same global cluster will share the same label. This step circumvents the label-switching problem, allowing the estimation of the global membership vector and the cluster posterior hyperparameters. Then, these updates are shared with the workers. The overall workflow of our model is illustrated in Figure 2.1.

Next, we provide the theoretical details of the collapsed Gibbs samplers executed at each level in the multivariate Gaussian case.

2.3.1 . Worker level

We denote by $\mathbf{x}^e = (x_1^e, \cdots, x_{n^e}^e)$ the observations assigned to the worker e, n^e the cardinal of $\mathbf{x}^e$, and $\mathbf{z}^e = (z_1^e, \cdots z_{n^e}^e)$ the local membership vector such that $z_i^e = k$ means that the observation x_i^e (assigned to the worker e) belongs to the local cluster k.

At this level, the local memberships are updated one by one using the collapsed version of Gibbs sampling. Each z_i^e is updated by sampling from

$$\mathrm{p}(z_i^e \mid \mathbf{z}_{-i}^e, \mathbf{x}^e, \Omega) \propto \begin{cases} \dfrac{n_k^e}{n^e - 1 + \alpha} \mathrm{p}(x_i^e \mid z_i^e = k, \mathbf{x}_k^e, G_0), & \text{existing cluster } k, \quad (2.1) \\ \dfrac{\alpha}{n^e - 1 + \alpha} \mathrm{p}(x_i^e \mid G_0), & \text{new cluster}, \quad (2.2) \end{cases}$$

where n_k^e is the size of cluster k in worker e and $\mathbf{x}_k^e$ is the content of cluster k in worker e. The posterior predictive Equation (2.1) and prior predictive Equation (2.2) can be obtained by integrating over θ:

$$\begin{aligned} \mathrm{p}(x_i^e \mid z_i = k, \mathbf{x}_k^e, G_0) &= \int_\Theta \mathrm{p}(x_i^e, \theta \mid \mathbf{x}_k^e, G_0) d\theta, \\ &= \int_\Theta \mathrm{p}(x_i^e \mid \theta) \mathrm{p}(\theta \mid \mathbf{x}_k^e, G_0) d\theta, \end{aligned} \quad (2.3)$$

and

$$\begin{aligned} \mathrm{p}(x_i^e \mid \Omega) &= \int_\Theta \mathrm{p}(x_i^e, \theta \mid G_0) d\theta, \\ &= \int_\Theta \mathrm{p}(x_i^e \mid \theta) \mathrm{p}(\theta \mid G_0) d\theta. \end{aligned} \quad (2.4)$$

Finally, the integrals in Equations (2.3) and (2.4) can be computed analytically. In fact, according to [117], the integral in Equation (2.4) is reduced to the following multivariate t-distribution:

$$t_{\nu_0 - d + 1}\left(x_i^e \mid \mu_0, \frac{(\kappa_0 + 1)\Psi_0}{\kappa_0(\nu_0 - d + 1)}\right),$$

and the integral in Equation (2.3) is reduced to the following t-distribution:

$$t_{\nu_k^e - d + 1}\left(x_i^e \mid \mu_k^e, \frac{(\kappa_k^e + 1)\Psi_k^e}{\kappa_k^e(\nu_k^e - d + 1)}\right),$$

where $\lambda_k^e = (\mu_k^e, \kappa_k^e, \Psi_k^e, \nu_k^e)$ are the posterior hyper-parameters associated to cluster k in worker e, updated as follows:

$$\kappa_k^e = \kappa_0 + n_k^e,\ \nu_k^e = \nu_0 + n_k^e,\ \mu_k^e = \frac{\kappa_0 \nu_0 + \sum_{x \in \mathbf{x}_k^e} x}{\kappa_k^e},$$

$$\Psi_k^e = \Psi_0 + \sum_{x \in \mathbf{x}_k^e} x x^T + \kappa_0 \mu_0 \mu_0^T - \kappa_k^e \mu_k^e \mu_k^{eT}.$$

After having updated the local membership vector, we compute the sufficient statistics [154] associated with each cluster. In the multivariate Gaussian case, the sufficient statistics (T_k^e, S_k^e) for a cluster $\mathbf{x}_k^e$ are given by [60]:

$$T_k^e = \frac{1}{n_k^e} \sum_{x \in \mathbf{x}_k^e} x \in \mathbb{R}^d, \tag{2.5}$$

$$S_k^e = \sum_{x \in \mathbf{x}_k^e} (x - T_k^e)(x - T_k^e)^T \in \mathbb{R}^{d \times d}. \tag{2.6}$$

Sufficient statistics and the sizes of each cluster are sent to the master. The DPMM inference process at the worker level is described in Algorithm 5.

Algorithm 5 Inference at worker level

Input: Dataset $\mathbf{x}^e$,
    Concentration parameter α,
    Prior G_0.
For $i \leftarrow 1$ **to** n^e **do**:
    Remove x_i^e from its local cluster.
    Compute $\mathrm{p}(z_i^e \mid \mathbf{z}_{-i}^e, \mathbf{x}^e, \Omega)$. ▷ *Using Equations 2.1 and 2.2*
    Sample z_i^e. ▷ *Draw new local membership*
    Add x_i^e to its new cluster.
For $k \leftarrow 1$ **to** K **do**:
    Compute the sufficient statistics (T_k^e, S_k^e). ▷ *Using Equations 2.5 and 2.6*
Output: Sufficient statistics $\{(T_1^e, S_1^e), \cdots, (T_K^e, S_K^e)\}$ and cluster sizes $(n_1^e, \cdots, n_K^e)$.

2.3.2 . Master level

The master receives from each worker the sample size of each cluster and its associated sufficient statistics. The objective is to estimate the membership vector $\mathbf{z} = (z_1, \cdots, z_n)$ and update the prior hyperparameters of each cluster. In this level, the observations are assigned by batch; a batch corresponds to a set of observations that belong to the same cluster. In fact, instead of assigning the observations one by one to their clusters, we assign a group of observations that already share the same local cluster (i.e., at the worker level) to a global cluster at the master level. Hence, the memberships of the observations that are assigned to the same global cluster k will share the same label

k. We sample the global membership z_h^e of the cluster $\mathbf{x}_h^e$ according to

$$\mathrm{p}(z_h^e \mid \mathbf{z}_{-h}^{-e}, \mathbf{x}, \Omega) \propto \begin{cases} \dfrac{n_k}{n-1+\alpha}\mathrm{p}(\mathbf{x}_h^e \mid z_h^e = k, \mathbf{x}_k, G_0), & \text{existing cluster } k, \quad (2.7) \\ \dfrac{\alpha}{n-1+\alpha}\mathrm{p}(\mathbf{x}_h^e \mid G_0), & \text{new cluster.} \quad (2.8) \end{cases}$$

In practice, the joint posterior predictive and the joint prior predictive distributions in Equations (2.7) and (2.8), respectively, are computed analytically by only using sufficient statistics, i.e., it is not necessary to have access to the content of each cluster. In fact, we have:

$$\mathrm{p}\left(\mathbf{x}_h^e \mid G_0\right) = \pi^{-n_h^e \frac{d}{2}} \cdot \frac{\kappa_0^{d/2}}{\left(\kappa_h^e\right)^{d/2}} \cdot \frac{\Gamma_d\left(\nu_h^e/2\right)}{\Gamma_d\left(\nu_0/2\right)} \cdot \frac{\det(\Psi_0)^{\nu_0/2}}{\det\left(\Psi_h^e\right)^{\nu_h^e/2}},$$

where $\det(\cdot)$ denotes the matrix determinant and the hyper-parameter values $(\mu_h^e, \kappa_h^e, \Psi_h^e, \nu_h^e)$ are obtained as follows:

$$\mu_h^e = \frac{\kappa_0\mu_0 + n_h^e T_h^e}{\kappa_h^e}, \quad \kappa_h^e = \kappa_0 + n_h^e, \quad \nu_h^e = \nu_0 + n_h^e,$$

$$\Psi_h^e = \Psi_0 + S_h^e + \frac{\kappa_0 n_h^e}{\kappa_h^e}\left(\mu_0 - T_h^e\right)\left(\mu_0 - T_h^e\right)^T,$$

where T_h^e and S_h^e are the sufficient statistics obtained from the workers. Moreover, we have

$$\mathrm{p}(\mathbf{x}_h^e \mid z_h^e = k, \mathbf{x}_k, G_0) = \pi^{\frac{-dn_h^e}{2}} \cdot \frac{\kappa_k^{d/2}}{\left(\kappa_h^e\right)^{d/2}} \cdot \frac{\Gamma_d\left(\nu_h^e/2\right)}{\Gamma_d\left(\nu_k/2\right)} \cdot \frac{\det\left(\Psi_k\right)^{\nu_k/2}}{\det\left(\Psi_h^e\right)^{\nu_h^e/2}},$$

and the posterior distribution parameters $(\mu_k, \kappa_k, \Psi_k, \nu_k)$ associated to the global cluster k, updated from the prior:

$$\mu_k = \frac{\kappa_0\mu_0 + n_k T_k}{\kappa_k}, \quad \kappa_k = \kappa_0 + n_k, \quad \nu_k = \nu_0 + n_k,$$

$$\Psi_k = \Psi_0 + S_k + \frac{\kappa_0 n_k}{\kappa_k}\left(\mu_0 - T_k\right)\left(\mu_0 - T_k\right)^T,$$

with T_k and S_k, the aggregated sufficient statistics obtained when local clusters are assigned to the same global cluster k are computed as follows:

$$T_k = \frac{1}{n_k}\sum_{j,h|\,\mathbf{z_h^e}=k} n_h^e \cdot T_h^e,$$

$$S_k = \sum_{j,h|\,\mathbf{z_h^e}=k} S_h^e + \sum_{j,h|\,\mathbf{z_h^e}=k}\left(n_j^h \cdot T_h^e \cdot T_h^{eT}\right) - n_k \cdot T_k \cdot T_k^T.$$

The inference process at the master level is described in Algorithm 6.

Algorithm 6 Inference at master level

Input: Sufficient statistics,
    Cluster sizes,
    Concentration parameter α,
    Prior G_0.
For each (e, h) **do**:
    Remove $\mathbf{x}_h^e$ from its global cluster.
    Compute $\mathrm{p}(z_h^e \mid \mathbf{z}_{-i}^{-h}, \mathbf{x}_h^e, \Omega)$. ▷ *Using Equations 2.7 and 2.8*
    Sample z_h^e. ▷ *Draw new global membership*
    Add $\mathbf{x}_h^e$ to its new global cluster.
Update the membership vector $\mathbf{z}$.
Output: Membership vector $\mathbf{z}$.

2.4 . Experiments

To evaluate the effectiveness of our approach, we conduct three types of experiments on synthetic and real-world datasets. Firstly, we compare our distributed algorithm with other state-of-the-art clustering algorithms in terms of clustering performance and convergence rate. Secondly, we compare the execution time and clustering performance of the distributed and the centralized collapsed Gibbs sampler from [121] on synthetic datasets of different sizes. Lastly, we investigate the scalability of our distributed algorithm by increasing the number of nodes while keeping the number of observations fixed. For this purpose, we execute our distributed approach on 10^6 data points multiple times, varying the number of cores.

2.4.1 . Implementation settings and distributed environment

In the following experiments, we use an uninformative prior NIW for both centralized and distributed collapsed Gibbs sampler algorithms . Therefore, both methods are implemented by setting the NIW hyperparameters as follows: μ_0 and the matrix precision Ψ_0 are respectively set to be the empirical mean vector and the covariance matrix of all data, as we want them as uninformative as possible. κ_0 and ν_0 represents our confidence in μ_0 and Ψ_0, are set to their lowest values, which are 1 and $d + 1$, respectively, where d is the dimension of the observation space. The initial state is a one-cluster partition. Finally, we have executed the distributed algorithms using the Neowise machine (1 CPU AMD EPYC 7642, 48 cores/CPU) hosted by the cluster grid5000 [1]. The centralized algorithm is executed on the same machine using one core. Both the centralized and distributed algorithms were implemented in Scala using Apache Spark version 3.3.0 and Scala version 2.13.0 to ensure

[1] https://www.grid5000.fr/w/Grid5000:Home

reproducibility. Apache Spark is a distributed computing system based on a master/worker architecture. A key concept in Spark is the resilient distributed dataset, which is a read-only collection of objects partitioned across a group of machines that can be rebuilt if necessary from the hierarchy of previous RDD operations.

2.4.2 . Clustering performance

To evaluate the clustering performance of our algorithm, we compare our approach with two distributed algorithms for the DPMM inference: M-R[2][59] and SubC[3] [46], and two parallelized clustering algorithms: Kmeans and GMM (for both methods, we have used the Spark Mllib implementation). These two parametric methods require the number of clusters. To ensure a fair comparison of the clustering quality with our approach, we set the number of clusters in these methods equal to the one inferred by the distributed collapsed Gibbs sampler.

It is important to note that, to the best of our knowledge, SubC and M-R are the most recent distributed inference approaches for DPMMs that are closest to our model in terms of assumptions and structure, and for which publicly available, working open-source implementations exist. Moreover, the method from [114] does not apply to the chosen datasets because it assumes a Dirichlet Process Gaussian mixture model with known variances.

All the executions are performed on the Neowise machine by distributing the data evenly on 32 cores. In this experiment, we use 8 different datasets described in table Table 2.1. Due to the smaller size of the EngyTime dataset, we distributed the data only to two workers. The image datasets are encoded using a variational auto-encoder (the architecture and implementation details are provided in the appendix), and each image is encoded into an 8-dimensional vector. We emphasize that all the datasets used do not contain any missing values. To evaluate the clustering performance, we compute the three clustering metrics: Adjusted Rand Index, Normalized Mutual Information, and clustering accuracy.

Table 2.2 reports the mean and the standard deviation of the clustering metrics, ARI, NMI, and ACC, achieved by each method on each dataset over 10 trials. The results show that our distributed approach outperforms other methods or achieves the second-best score in almost all the datasets for the three clustering metrics. However, we note that our method is less effective on certain datasets. Several factors may contribute to this. First, the cluster assignments are sampled from an approximation of the posterior distribution, which can result in noisy partitions. These partitions could be aggregated

[2]The code source is taken from: https://github.com/wangruohui/distributed-dpmm

[3]The code source is taken from: https://github.com/BGU-CS-VIL/DPMMSubClusters.jl/tree/master

Dataset	n	d	K	Description
EngyTime	4096	2	2	Synthetic dataset generated from two Gaussian mixtures.
Synthetic 10K	10000	2	6	Synthetic dataset generated from Gaussian components of dimension 2.
Mnist	70000	8	10	Handwritten digits [44].
Fashion mnist	70000	8	10	Zalando's article images [168].
Letter	103600	8	26	Handwritten letters [38].
Balanced	131600	8	47	Both of handwritten digits and letters [38].
Digits	280000	8	10	Handwritten digits [38].
UrbanGB	360177	3	469	Coordinates (longitude and latitude) of road accidents [47].

Table 2.1: The description of the datasets used to evaluate the clustering performance of our distributed inference approach. n denotes the size, d the dimension, K the true number of clusters.

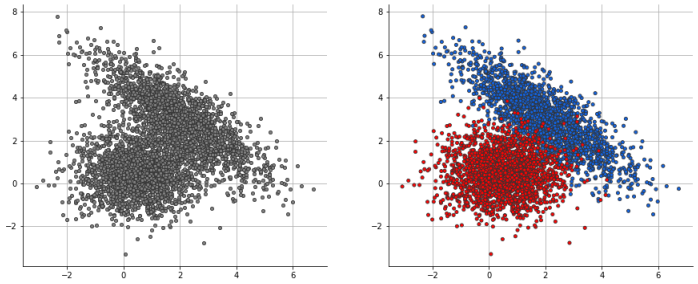

Figure 2.2: Unlabeled data (left) and labeled data (right), after $M = 100$ iterations of DisCGS on EngyTime dataset, ARI $= 0.99$, NMI $= 0.99$, and ACC $= 0.99$. The number of estimated clusters is 2.

by applying a consensus partition estimation. Second, inadequate feature extraction techniques may lead to inadequate information representation. Third, the current initialization strategy may not provide an optimal starting point for inference. Incorporating a more sophisticated initialization method, such as K-means or GMM, could potentially enhance clustering quality. Additionally, tuning the number of iterations performed at the worker or master level before sharing information between them may further improve both clustering performance and reduce communication overhead. Moreover, it is important to note that in this experiment, we only focus on comparing the clustering performance of the different methods; we do not compare the execution times of the three methods because the other approaches proposed an inference algorithm that differs from the collapsed Gibbs sampler.

Figures 2.2 and 2.3 illustrate the estimated clusters obtained by our distributed approach on the EngyTime and synthetic 10K datasets, respectively. These figures respectively represent the best partition achieved over the ten trials. The purpose of using these datasets is to evaluate our algorithm's performance when confronted with complex datasets that exhibit overlapping clusters. The results demonstrate that our method performs well even when dealing with such challenging datasets. Specifically, on the EngyTime dataset,

Dataset		DisCGS (ours)	M-R [59]	SubC [46]	GMM	Kmeans
EngyTime	ARI	$\mathbf{0.94} \mp 0.07$	0.47 ∓ 0.04	$\underline{0.87} \mp 0.00$	0.73 ∓ 0.13	0.54 ∓ 0.20
	NMI	$\mathbf{0.92} \mp 0.07$	0.42 ∓ 0.02	$\underline{0.79} \mp 0.00$	0.67 ∓ 0.09	0.56 ∓ 0.11
	ACC	$\underline{0.96} \mp 0.04$	0.75 ∓ 0.04	$\mathbf{0.97} \mp 0.00$	0.85 ∓ 0.11	0.68 ∓ 0.20
Synthetic 10K	ARI	$\mathbf{0.80} \mp 0.06$	0.10 ∓ 0.01	0.38 ∓ 0.07	$\underline{0.80} \mp 0.07$	0.75 ∓ 0.03
	NMI	$\underline{0.86} \mp 0.04$	0.12 ∓ 0.01	0.61 ∓ 0.08	$\mathbf{0.88} \mp 0.04$	0.85 ∓ 0.01
	ACC	$\mathbf{0.88} \mp 0.06$	0.29 ∓ 0.01	0.38 ∓ 0.08	$\underline{0.85} \mp 0.08$	0.76 ∓ 0.05
Mnist	ARI	$\mathbf{0.72} \mp 0.01$	0.20 ∓ 0.07	$\underline{0.66} \mp 0.04$	0.39 ∓ 0.02	0.26 ∓ 0.01
	NMI	$\underline{0.74} \mp 0.00$	0.38 ∓ 0.08	$\mathbf{0.79} \mp 0.01$	0.69 ∓ 0.01	0.62 ∓ 0.00
	ACC	$\mathbf{0.79} \mp 0.01$	0.30 ∓ 0.06	$\underline{0.71} \mp 0.03$	0.38 ∓ 0.02	0.23 ∓ 0.01
Fashion-Mnist	ARI	$\mathbf{0.45} \mp 0.02$	0.35 ∓ 0.02	0.40 ∓ 0.02	$\underline{0.41} \mp 0.02$	0.37 ∓ 0.02
	NMI	$\mathbf{0.60} \mp 0.01$	0.54 ∓ 0.01	$\mathbf{0.60} \mp 0.01$	$\underline{0.59} \mp 0.02$	0.57 ∓ 0.01
	ACC	$\mathbf{0.55} \mp 0.02$	0.45 ∓ 0.03	0.48 ∓ 0.01	$\underline{0.52} \mp 0.04$	0.48 ∓ 0.01
Letter	ARI	$\mathbf{0.30} \mp 0.01$	0.07 ∓ 0.03	$\underline{0.23} \mp 0.05$	$\mathbf{0.30} \mp 0.01$	0.20 ∓ 0.00
	NMI	$\underline{0.56} \mp 0.01$	0.23 ∓ 0.05	0.47 ∓ 0.06	$\mathbf{0.61} \mp 0.00$	0.52 ∓ 0.00
	ACC	$\mathbf{0.41} \mp 0.01$	0.14 ∓ 0.04	0.31 ∓ 0.06	$\underline{0.33} \mp 0.01$	0.24 ∓ 0.01
Balanced	ARI	$\mathbf{0.35} \mp 0.00$	0.02 ∓ 0.01	0.05 ∓ 0.02	$\underline{0.35} \mp 0.01$	0.22 ∓ 0.00
	NMI	$\underline{0.59} \mp 0.00$	0.17 ∓ 0.06	0.31 ∓ 0.04	$\mathbf{0.63} \mp 0.00$	0.54 ∓ 0.00
	ACC	$\mathbf{0.46} \mp 0.01$	0.07 ∓ 0.02	0.10 ∓ 0.02	$\underline{0.44} \mp 0.02$	0.30 ∓ 0.01
Digits	ARI	$\underline{0.55} \mp 0.01$	0.28 ∓ 0.14	$\mathbf{0.74} \mp 0.05$	0.46 ∓ 0.02	0.36 ∓ 0.01
	NMI	$\underline{0.71} \mp 0.01$	0.51 ∓ 0.14	$\mathbf{0.79} \mp 0.02$	$\underline{0.71} \mp 0.01$	0.63 ∓ 0.00
	ACC	$\underline{0.58} \mp 0.00$	0.38 ∓ 0.09	$\mathbf{0.81} \mp 0.05$	0.44 ∓ 0.03	0.34 ∓ 0.01
UrbanGB	ARI	$\underline{0.63} \mp 0.01$	0.12 ∓ 0.05	0.09 ∓ 0.00	$\mathbf{0.67} \mp 0.05$	0.49 ∓ 0.06
	NMI	0.68 ∓ 0.01	0.21 ∓ 0.07	0.24 ∓ 0.00	$\underline{0.77} \mp 0.01$	$\mathbf{0.81} \mp 0.01$
	ACC	0.45 ∓ 0.01	0.29 ∓ 0.02	0.29 ∓ 0.00	$\underline{0.53} \mp 0.02$	$\mathbf{0.54} \mp 0.02$

Table 2.2: The mean and the standard deviation of the clustering metrics, over 10 runs on different datasets. The best result within each row is marked in bold, and the runner-up is underlined.

our method attained an ARI, NMI, and ACC score of 0.99. It accurately inferred the correct number of clusters, which is two. On the synthetic 10K dataset, our method achieved an ARI, NMI, and ACC score of 0.85, 0.89, and 0.91, respectively. It estimated a total of seven clusters, while the exact number of clusters is six.

Figure 2.4 illustrates a sample of the ten first partitions estimated by our approach on the Mnist dataset. It also represents the best partition achieved over the ten trials. The results show that the partitions group together the same digits. For instance, the first cluster groups the digit four (4). The second cluster contains mostly the digit eight (8) and so on. Moreover, our method obtained an ARI, NMI, and ACC score of 0.73, 0.74, and 0.80, respectively, on the Mnist dataset. It is important to note that our approach inferred a total of 57 clusters for this dataset. This is because our method not only grouped digits of the same type together but also took into account variations in handwriting styles. Consequently, our method estimated a larger number of clusters than

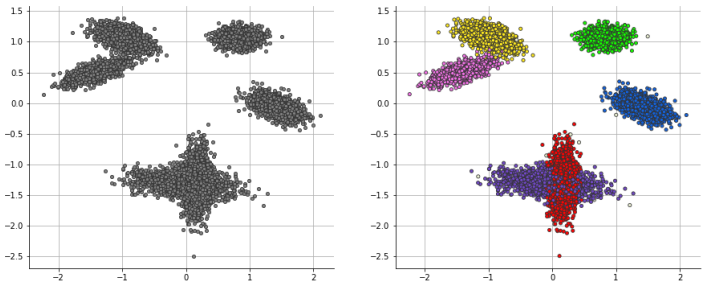

Figure 2.3: Unlabeled (left) and labeled data (right), after 100 iterations of DisCGS on a synthetic dataset with overlapped clusters, ARI $= 0.85$, NMI $= 0.89$, and ACC $= 0.91$. The number of estimated clusters is 7.

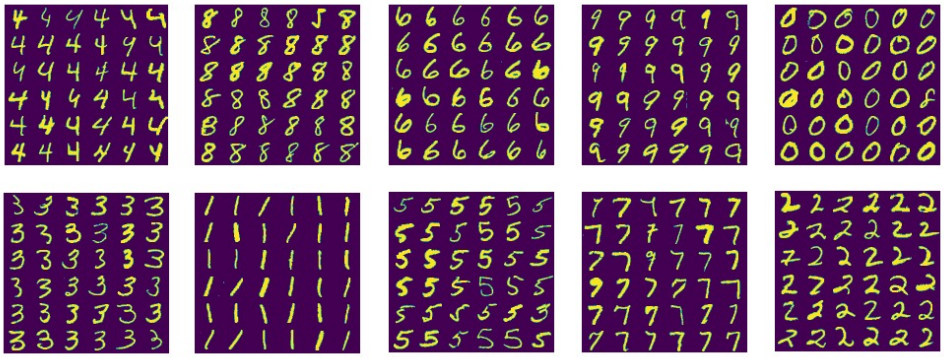

Figure 2.4: A sample of the first ten clusters inferred by the DisCGS on Mnist dataset ARI $= 0.73$, NMI $= 0.74$, and ACC $= 0.80$. The number of estimated clusters is 57.

the ground truth, as depicted in Figure 2.5. However, as stated in [118], in many cases of clustering tasks, determining the appropriate number of clusters is not straightforward. There is often no well-defined or "true" value for the parameter K, and this value may not be unique.

2.4.3 . Convergence

In this experiment, we examine the convergence of both likelihood and ARI score of three methods: our distributed collapsed Gibbs sampler approach (DisCGS), collapsed Gibbs sampler (CGS), and the distributed MCMC inference (SubC) from [46]. These evaluations are performed on six datasets: Synthetic 100K, Mnist, Fashion-mnist, Mnist-letter, and Balanced. It's important to note that only the centralized collapsed Gibbs sampler and SubC share the same model assumption as our distributed method, resulting in a comparable likelihood.

Figure 2.6 illustrates the evolution of the log-likelihood and ARI score at each iteration. We observe that our algorithm converges almost at a similar

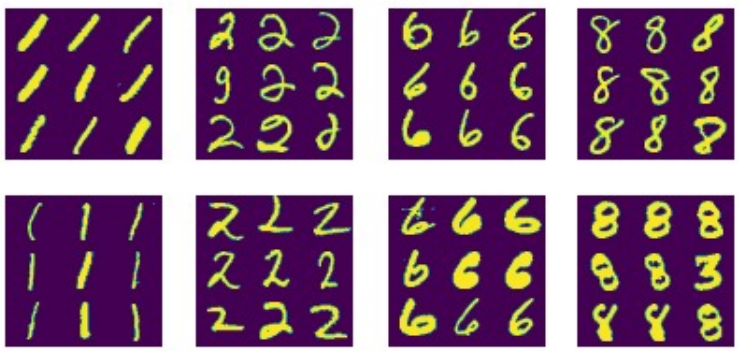

Figure 2.5: Each line illustrates two different clusters that grouped the same digits with different handwriting digits.

rate as the centralized CGS algorithm and is much faster than SubC, which takes more iterations to converge. This is because our algorithm is able to discover new local clusters inside each worker. Whereas, in SubC, the number of clusters is fixed when performing the restricted Gibbs sampler in each worker, and new components are only discovered at the master level during the split step. Thus, more iterations are required to generate enough components to model the data. This phenomenon is also observed in [164]. Overall, our algorithm converges really fast and maintains a stable ARI score over the iterations. Whereas the CGS and SubC may downgrade their ARI score after some iterations, as can be observed for the Fashion-Mnist dataset.

2.4.4 . Comparison of the distributed and centralized collapsed Gibbs sampler

In this experience, we compare the execution time and clustering performance of the distributed collapsed Gibbs sampler and the centralized collapsed Gibbs sampler from [121], we execute both algorithms on synthetic datasets of different sizes (from $n = 20$K to $n = 100$K) generated from $K = 10$ Gaussian components of dimension 2. The centralized version is too slow; running over 100K observations would take too much time. We use 32 cores for this experiment.

Figure 2.7 illustrates, using a logarithmic scale, the execution time for $M = 100$ iterations of both the distributed and centralized inference methods. The results show that our distributed algorithm significantly outperforms the centralized approach. For instance, when considering 100K data points, the centralized algorithm takes approximately 12 hours to complete 100 iterations, whereas our approach achieves the same number of iterations in just 3 minutes, reducing the execution time by a factor of 200. Table 2.3 presents the clustering metrics (ARI, NMI, and ACC) obtained by each algorithm on each dataset. The results indicate that our approach consistently achieves high

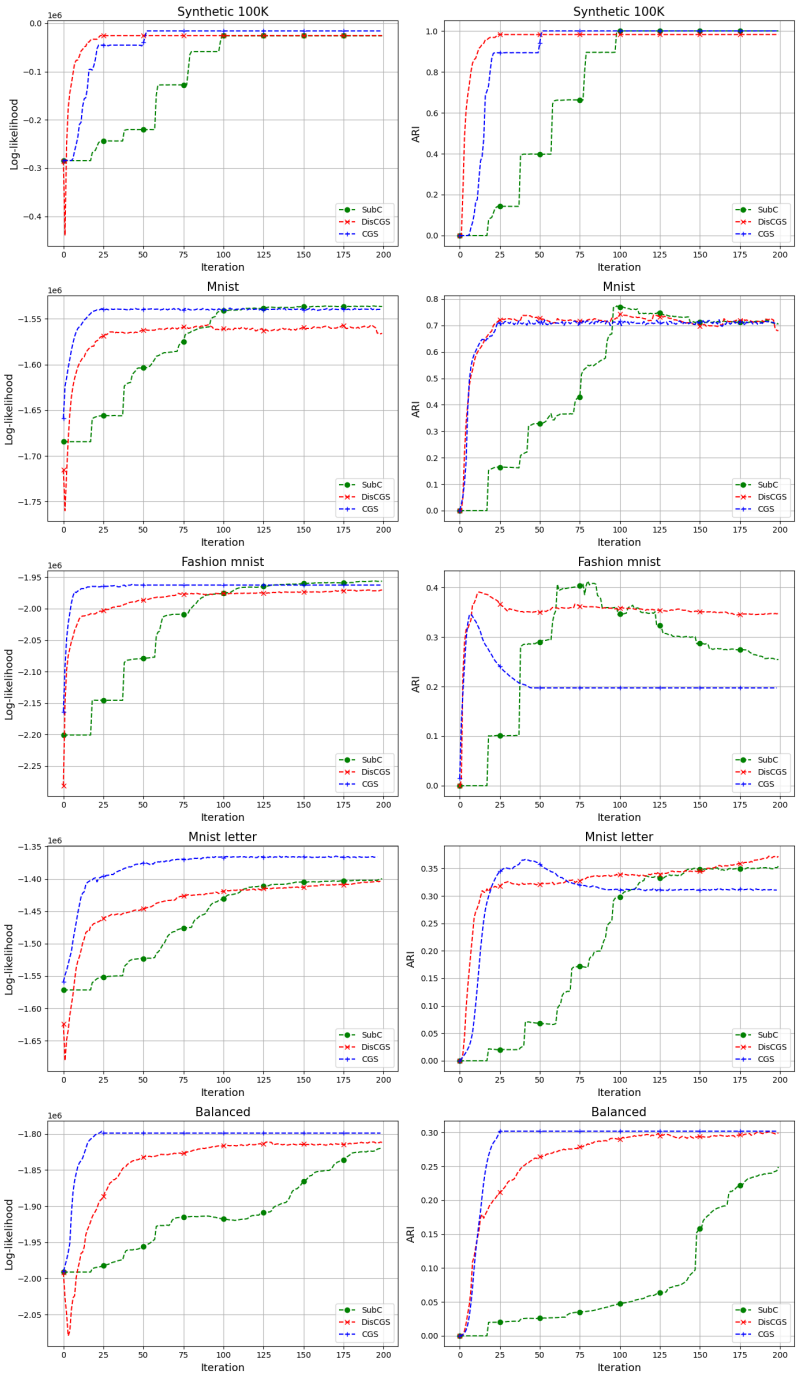

Figure 2.6: Log-likelihood and ARI score every iteration.

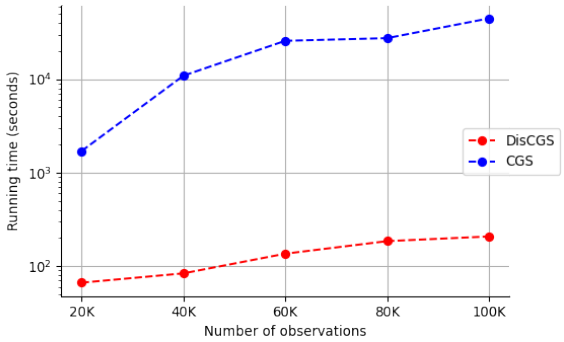

Figure 2.7: Running time (in logarithmic scale) for 100 iterations of DisCGS and CGS inference for DPMM on synthetic datasets of different sizes.

scores and outperforms the centralized algorithm in almost all cases. We observe that using the dataset of size 40K, the centralized algorithm obtained slightly higher ARI and NMI scores than the distributed algorithm. This is not surprising since both approaches sample the memberships from an approximation of the posterior distribution, resulting in noisy inferred partitions. These samples can be aggregated after a given number of burn-in iterations with a consensus partition estimation. Overall, the findings confirm that the distributed inference does not compromise the clustering performance while considerably reducing the execution time.

Dataset size	ARI		NMI		ACC	
	Dis.	Cen.	Dis.	Cen.	Dis.	Cen.
20K	**0.99**	0.89	**0.99**	0.96	**0.99**	0.89
40K	0.96	**0.99**	0.97	**0.99**	**0.99**	0.97
60K	**0.91**	0.89	0.92	**0.96**	**0.92**	0.89
80K	**0.94**	0.89	**0.96**	**0.96**	**0.91**	0.89
100K	**0.91**	0.89	**0.94**	0.89	**0.91**	0.89

Table 2.3: Clustering metrics obtained by the distributed (Dis.) and centralized (Cen.) inference on synthetic datasets of different sizes.

2.4.5 . Distributed algorithm scale-up

In this experiment, we use $n = 10^6$ data points generated from $K = 10$ components of two-dimensional Gaussian. We run our distributed inference several times by increasing the number of cores from 8 to 48.

Figure 2.8 represents the running time as a function of the number of cores. We observe that the running time decreases when the number of cores increases, showing that our algorithm scales efficiently with the number of

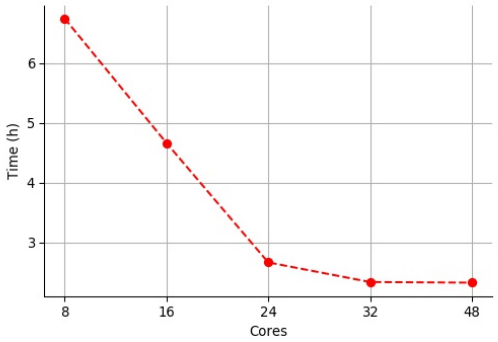

Figure 2.8: Running time (hours) as a function of the number of cores of DisCGS on 10^6 data points.

workers. Figure 2.9 shows the clusters (labels) inferred by our approach on 10^6 data points distributed across 32 cores. As depicted, our approach successfully identifies meaningful and coherent clusters, resulting in high ARI, NMI, and ACC scores of approximately 0.98. Similar results were observed when distributing the data on different numbers of cores. Additionally, it has inferred 14 clusters while the ground truth is 10. However, only 10 of them are significant clusters, and the 4 others are only "outliers". Moreover, the clustering performance and the number of clusters depend on the auto-encoder. Also, the number of clusters might be influenced by the concentration parameter α [118].

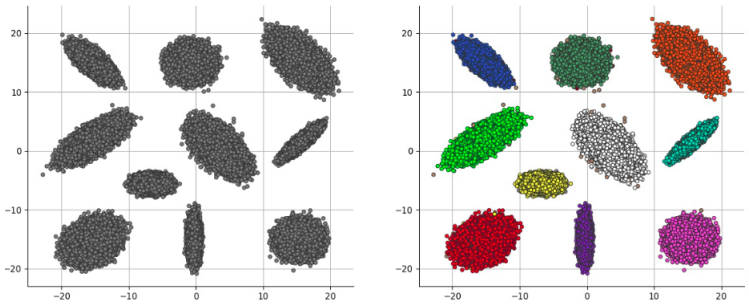

Figure 2.9: Unlabeled data (left) and labeled data (right), after $M = 100$ iterations of DisCGS on 10^6 data points, using 32 cores, ARI $= 0.98$, NMI $= 0.98$, and ACC $= 0.98$. The number of inferred clusters is 14.

2.5 . Conclusion

In this chapter, we presented DisCGS, a novel distributed inference method for Dirichlet Process Mixture Models. DPMMs are a powerful tool for clus-

tering, particularly when the number of clusters is unknown. However, their inference process becomes computationally expensive as the number of observations is large.

To address this challenge, we developed a distributed approach that approximates the collapsed Gibbs sampler using sufficient statistics. This allows for efficient computation of the posterior distribution, estimation of the global model, and clustering structure, without sharing raw data. We validated our method through extensive experiments on both synthetic and real-world datasets, showing that it drastically reduces inference time while maintaining high clustering accuracy.

In Chapter 4, we extend this framework to the exponential family of component distributions, relaxing the assumption that clusters follow a multivariate Gaussian distribution. This generalization enables its application to other data types, such as discrete data. Furthermore, the algorithm serves as a foundational component for distributing the inference process in Bayesian non-parametric latent block models, as discussed in Chapter 5.

Finally, the proposed methodology is designed to operate in distributed environments with independent and heterogeneous machines, making it particularly suitable for horizontal federated learning scenarios, where workers correspond to different components of an edge network. A natural extension of this work would be to adapt and implement the method on a fully federated environment.

3 - Distributed Inference for DPMMs with Multinomial Components and Generalization to the Exponential Family

In this chapter, we introduce a distributed inference method for Multinomial Dirichlet Process Mixture Models designed for discrete data, with a focus on text clustering. We then demonstrate how the methodology generalizes to the exponential family of distributions. The work presented in this chapter has been published in [90].

3.1 . Introduction

Several works involving Dirichlet Process Mixture Models naturally choose Gaussian distributions for the component models due to their favorable computational properties and convenience. However, this choice is not always the most suitable. For instance, [93, 99] proposed Dirichlet process mixtures of beta distributions. In [93], the model was applied to density and intensity estimation, whereas in [99], it was used for image categorization and object detection.

Moreover, when dealing with discrete data (e.g., count data), Gaussian distributions are clearly inappropriate. Natural alternatives in such cases include the Multinomial and Poisson distributions. For example, [178, 110] introduced Dirichlet–Multinomial and Gamma–Poisson mixture models, respectively, for short text clustering. These examples highlight the need to go beyond Gaussian distributions.

In this context, we extend our distributed inference method, originally designed for continuous observations with Multivariate Gaussian clusters, to discrete observations with Multinomial clusters. For clarity, we present the model in the context of text clustering. Furthermore, we generalize the approach to cases where the distribution associated with each cluster belongs to the exponential family. We demonstrate the effectiveness of this method on real-world datasets, showing that it reduces inference time significantly without compromising clustering quality.

The remainder of this chapter is organized as follows. In Section 3.2, we recall the formulation of the Dirichlet process Multinomial mixture models in the context of text clustering. In Section 3.3, we present our distributed inference algorithm, including the collapsed Gibbs sampler at the worker and master levels. In Section 3.4, we describe its generalization to the exponential family. Section 3.5 reports empirical results, and Section 3.6 offers concluding

remarks.

3.2 . Model definition

Consider a corpus $\mathbf{x} = (x_1, \ldots, x_n)$ consisting of n documents and a vocabulary of size V. For each document x_i in the corpus, let z_i denote its cluster assignment. For a given cluster k, let n_k be the number of documents assigned to cluster k. For a word w in the vocabulary, let $n_{k,w}$ denote the total number of times w appears in cluster k; hence, $N_k = \sum_{w=1}^{V} n_{k,w}$ is the total number of word occurrences in cluster k. Similarly, let $\ell_{x_i,w}$ be the number of times w appears in document x_i, and define the total length of document x_i as $\ell_{x_i} = \sum_{w=1}^{V} \ell_{x_i,w} = \sum_{w \in x_i} \ell_{x_i,w}$. Therefore, the DPMM assumes the following generative process:

$$
\begin{aligned}
&\pi \sim SB(\alpha), \\
&z_i \overset{\text{i.i.d.}}{\sim} \text{Mult}\left(\pi\right), && \forall i \in \{1, \cdots, n\}, \\
&\theta_k \overset{\text{i.i.d.}}{\sim} \text{Dir}(\boldsymbol{\gamma}), && \forall k \in \{1, 2, \cdots\}, \\
&x_i \mid \{z_i = k, \theta_k\} \overset{\text{i.i.d.}}{\sim} \text{p}\left(x_i \mid \theta_k\right), && \forall i \in \{1, \cdots, n\}.
\end{aligned}
$$

Under this model definition, to generate a document x_i, we first sample its cluster (or "topic") label according to an infinite set of mixture weights defined by the stick-breaking process. Then, we draw the corresponding cluster parameters θ_k from the conjugate prior, which is the Dirichlet distribution with hyperparameter $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_V)$, where $\gamma_w > 0$ for all $w \in \{1, \ldots, V\}$. Finally, the document $x_i \mid \{z_i = k, \theta_k\}$ is generated according to the k-th component of the mixture distribution $\text{p}(x_i \mid \theta_k)$.

For simplicity, and following common practice in the literature [177, 179], we set $\gamma_1 = \cdots = \gamma_V = \gamma$. We also adopt the Naive Bayes assumption: for all $i \in \{1, \ldots, n\}$, given $z_i = k$, the words in document x_i are generated independently, and the probability of a word is assumed to be independent of its position within the document. Under this assumption, the conditional distribution $\text{p}(x_i \mid \theta_k)$ can be defined as follows:

$$
\text{p}(x_i \mid \theta_k) = \prod_{w \in x_i} \text{Mult}(w \mid \theta_k),
$$

This means that each mixture component distribution is a multinomial distribution over words. In addition, for all $k \geq 1$, we have $\theta_k = (\theta_{k,w})_{w=1}^{V} \in [0, 1]^V$ and

$$
\sum_{w=1}^{V} \theta_{k,w} = 1.
$$

In the text clustering problem, given a corpus with an observable finite number of clusters (topics), the goal is to estimate the membership $\mathbf{z}$. In this case,

the prior Equation (1.8) and posterior predictive Equation (1.8) distributions are respectively computed as follows:

$$\mathrm{p}(x_i \mid G_0) = \frac{\prod_{w \in x_i} \prod_{s=1}^{\ell_{x_i,w}} (\gamma + s - 1)}{\prod_{s=1}^{\ell_{x_i}} (V\gamma + s - 1)}, \tag{3.1}$$

and

$$\mathrm{p}(x_i \mid z_i = k, \mathbf{x}_k, G_0) = \frac{\prod_{w \in x_i} \prod_{s=1}^{\ell_{x_i,w}} (\gamma + n_{k,w} + s - 1)}{\prod_{s=1}^{\ell_{x_i}} (V\gamma + N_k + s - 1)}, \tag{3.2}$$

Here, $n_{k,w}$ denotes the total number of times word w appears in the k-th cluster and $N_k = \sum_{w=1}^{V} n_{k,w}$ is the total number of word occurrences in the same local cluster. The derivation of Equations (3.1) and (3.2) are provided in [179].

3.3 . Distributed inference

In this section, we present the theoretical foundations of the proposed distributed inference algorithm in the setting where observations are discrete and component distributions follow a Multinomial model. We focus on the text clustering problem, where the objective is to estimate the cluster memberships $\mathbf{z}$ for a corpus of documents associated with a finite but unknown number of topics. Once the data have been evenly distributed among the workers, our distributed approach performs the following two main steps:

3.3.1 . Worker level

We denote by $\mathbf{x}^e = (x_1^e, \cdots, x_{n^e}^e)$ the set of documents assigned to the e-th worker, where n^e is the total number of documents assigned to that worker. Similarly, let $\mathbf{z}^e = (z_1^e, \cdots, z_{n^e}^e)$ denote the corresponding local membership assignments. As in the multivariate Gaussian case, documents are assigned to local clusters one at a time by sampling their label z_i^e from the conditional distribution $\mathrm{p}(z_i^e \mid \mathbf{z}_{-i}^e, \mathbf{x}^e, \Omega)$. The only difference lies in the computation of the prior and posterior predictive distributions. Under the assumption of conjugacy, these are respectively computed as follows:

$$\mathrm{p}(x_i^e \mid G_0) = \frac{\prod_{w \in x_i^e} \prod_{s=1}^{\ell_{x_i^e,w}} (\gamma + s - 1)}{\prod_{s=1}^{\ell_{x_i^e}} (V^e\gamma + s - 1)}, \tag{3.3}$$

and

$$\mathrm{p}(x_i^e \mid z_i^e = k, \mathbf{x}_k^e, G_0) = \frac{\prod_{w \in x_i^e} \prod_{s=1}^{\ell_{x_i^e,w}} (\gamma + n_{k,w}^e + s - 1)}{\prod_{s=1}^{\ell_{x_i^e}} (V^e\gamma + N_k^e + s - 1)}, \tag{3.4}$$

Here, $n_{k,w}^e$ denote the total number of times word w appears in the k-th cluster of the e-th worker, and $N_k^e = \sum_{w=1}^{V^e} n_{k,w}^j$ is the total number of word occurrences in the same local cluster. The derivation of Equations (3.3) and (3.4)

is provided in [179]. Since the component distribution is a multinomial over words, the sufficient statistics for cluster k in worker e are given by the word count vector $\boldsymbol{n}_k^e = (n_{k,w}^e)_{w \in V^e}$. The sufficient statistics and the sizes of each cluster are then sent to the master.

3.3.2 . Master level

The goal at this stage is to estimate the global membership vector $\mathbf{z} = (z_1, \ldots, z_n)$ and update the prior hyperparameters for each global cluster. To this end, sets of documents assigned to the same local cluster on a worker are grouped and reassigned to a global cluster at the master level. Consequently, all documents mapped to the same global cluster will share a common membership label. For each set of documents $\mathbf{x}_h^e$ belonging to the local cluster h of worker e, we sample the global membership z_h^e from the posterior distribution $\mathrm{p}(z_h^e \mid \mathbf{z}_{-h}^{-e}, \mathbf{x}, \Omega) \propto$

$$\begin{cases} \dfrac{n_k}{n-1+\alpha} \mathrm{p}(\mathbf{x}_h^e \mid z_h^e = k, \mathbf{x}_k, G_0), & \text{existing cluster } k, \quad (3.5) \\ \dfrac{\alpha}{n-1+\alpha} \mathrm{p}(\mathbf{x}_h^e \mid G_0), & \text{new cluster.} \quad (3.6) \end{cases}$$

In practice, the joint posterior predictive and the joint prior predictive distributions in Equations (3.5) and (3.6) are computed by only using sufficient statistics, i.e., it is not necessary to have access to the content of each cluster. In fact, we have:

$$\mathrm{p}\left(\mathbf{x}_h^e \mid G_0\right) = \frac{\prod_{w \in \mathbf{x}_h^e} \prod_{s=1}^{n_{h,w}^e} (\gamma + s - 1)}{\prod_{s=1}^{N_h^e} (V\gamma + s - 1)}, \tag{3.7}$$

and

$$\mathrm{p}(\mathbf{x}_h^e \mid z_h^e = k, \mathbf{x}_k, G_0) = \frac{\prod_{w \in \mathbf{x}_h^e} \prod_{s=1}^{n_{h,w}^e} (\gamma + n_{k,w} + s - 1)}{\prod_{s=1}^{N_h^e} (V\gamma + N_k + s - 1)}. \tag{3.8}$$

Here, $n_{k,w}$ denotes the total number of times word w appears in global cluster k, and $N_k = \sum_{w=1}^{V} n_{k,w}$ is the total number of word occurrences in that cluster. These quantities represent the aggregated sufficient statistics obtained by merging the local clusters. The derivation of Equations (3.7) and (3.8) is provided in Section 6.7.3.

3.4 . Generalization to the exponential family case

3.4.1 . Model definition

In this section, we generalize the results above to the exponential family. In this context, the DPMM is defined as follows:

$$\pi \sim SB(\alpha),$$

$$
\begin{aligned}
z_i &\overset{\text{i.i.d.}}{\sim} \mathrm{Cat}\left(\pi\right), && \forall i \in \{1, \cdots, n\}, \\
\theta_k &\overset{\text{i.i.d.}}{\sim} G_0, && \forall k \in \{1, 2, \cdots\}, \\
x_i \mid \{z_i = k, \theta_k\} &\overset{\text{i.i.d.}}{\sim} f(x_i, \theta_k), && \forall i \in \{1, \cdots, n\}.
\end{aligned}
$$

with $f(\cdot, \theta_k)$ the component distribution associated to cluster k. In the following, we assume that f belongs to the exponential family. This means that f can be written as follows:

$$
f(x, \theta) = \exp\left(\eta(\theta)^T S(x) - \phi(x) - \psi(\theta)\right), \quad \forall x \in \mathbb{R}^d, \forall \theta \in \Theta,
$$

where η, S, ϕ and ψ are known functions. In this assumption, $\eta(\theta)$ is the natural parameter vector, $S(x)$ is said to be the sufficient statistics for the natural parameter vector $\eta(\theta)$, ψ is the log partition function that ensures the density to integrate to 1, and ϕ is the base measure of distribution. Under this assumption, a conjugate prior G_0 with a closed form will always exist, and we assume that it can be written as:

$$
g(\theta, \Lambda_0, \tau_0) = \exp\left(\eta(\theta)^T \Lambda_0 - \tau_0 \psi(\theta) - \xi(\Lambda_0, \tau_0)\right).
$$

Here, Λ_0 and τ_0 are the prior hyperparameters of the base distribution G_0. They might be interpreted as follows: Λ_0 represents our prior on $S(x)$ and has the same form as the natural parameter. τ_0 is a scalar that represents our prior on the number of observations n.

3.4.2 . Distributed inference

Worker level

Under these assumptions, the posterior and prior predictive distributions in worker e have a closed form, and they are given by:

$$
\mathrm{p}(x_i^e | z_i^e = k, \mathbf{x}_k^e, G_0) = \exp\left(\xi(\Lambda_k^e + S(x_i^e), \tau_k^e + 1) - \xi(\Lambda_k^e, \tau_k^e) - \phi(x_i^e)\right),
$$

and

$$
\mathrm{p}(x_i^e | G_0) = \exp\left(\xi(\Lambda_0 + S(x_i^e), \tau_0 + 1) - \xi(\Lambda_0, \tau_0) - \phi(x_i^e)\right),
$$

where Λ_k^e and τ_k^e are the posterior hyper-parameters of cluster k in worker e, and are given by

$$
\tau_k^e = \tau_0 + n_k^e,
$$

and

$$
\Lambda_k^e = \Lambda_0 + \sum_{x \in \mathbf{x}_k^e} S(x).
$$

Master level

At the master level, the joint posterior and prior predictive distributions (Equation (2.7)) have a closed form and they are given by:

$$\mathrm{p}(\mathbf{x}_h^e \mid z_h^e = k, \mathbf{x}_k, G_0) = \exp\left(\xi(\Lambda_k + \sum_{x \in \mathbf{x}_h^e} S(x), \tau_k + 1) - \xi(\Lambda_k, \tau_k) - \sum_{x \in \mathbf{x}_h^e} \phi(x)\right)$$

and

$$\mathrm{p}(\mathbf{x}_h^e \mid G_0) = \exp\left(\xi(\Lambda_0 + \sum_{x \in \mathbf{x}_h^e} S(x), \tau_0 + 1) - \xi(\Lambda_0, \tau_0) - \sum_{x \in \mathbf{x}_h^e} \phi(x)\right)$$

where Λ_k and τ_k are the posterior hyper-parameters of global cluster k. They are obtained as follows:

$$\Lambda_k = \sum_{(e,h)|z_h^e = k} \sum_{x \in x_h^e} S(x)$$

and

$$\tau_k = \tau_0 + \sum_{(e,h)|z_h^e = k} n_h^e$$

In the posterior, prior, joint posterior, and joint prior predictive distributions depend on ϕ, but since this term appears in both prior and posterior, it is not necessary to compute this term. One can factorize and implicitly compute this term when computing the normalizing constant. This means that the joint prior and posterior predictive distributions depend on data only through sufficient statistics.

3.5 . Experiments

To evaluate our approach on discrete data in the context of text clustering, we conduct three types of experiments on real-world datasets. First, we compare our distributed algorithm with several state-of-the-art clustering methods in terms of clustering performance. Second, we assess both the execution time and clustering quality of our distributed approach against the centralized collapsed Gibbs sampler from [179] on datasets of different sizes. Finally, we examine the scalability of our distributed algorithm by increasing the number of processing nodes while keeping the number of documents fixed. For this purpose, we run our distributed method multiple times on a corpus of 300,000 documents, varying the number of CPU cores.

3.5.1 . Implementation settings

In the following experiments, we initialize the model with a one-cluster partition. For the centralized approach, the concentration parameter α is set to $0.1 \times n$, where n is the total number of observations, following the configuration in [179]. For the distributed approach, we adopt a hierarchical setting for α: at the worker level, for each worker e, we set the local concentration parameter $\alpha = 0.1 \times n^e$, where n^e is the number of observations handled by worker e. At the master level, the parameter is set to $\alpha = 0.1 \times \frac{n_M}{W}$, where n_M is the number of aggregated observations at the master and W is the number of workers. This setup is based on the idea that with more workers, there will be more clusters to merge at the master level. So, we set the parameters to encourage stronger merging of clusters over the creation of new ones, in order to prevent overestimating the number of clusters. The hyperprior parameter γ is set to 0.02 for the centralized approach, following [179]. In the distributed setting, α is set to either 0.02 or 0.05 at the worker level, while at the master level, γ is scaled as $\gamma \times W$, reflecting the same intuition used for setting the concentration parameter, encouraging cluster merging as the number of workers increases. All the used datasets are preprocessed by transforming text to lowercase, eliminating stop words, and applying stemming to standardize word forms as in [179].

All distributed experiments were executed on the Neowise machine (1× AMD EPYC 7642 CPU, 48 cores) hosted by the Grid'5000 cluster infrastructure[1]. For fair comparison, the centralized approach was run on the same machine using a single CPU core.

3.5.2 . Clustering performance

In this experiment, we evaluate the clustering performance of our algorithm by comparing it against several established text clustering methods, including GSDPMM [179], GSDMM [177], K-means, and LDA [21], using four different datasets summarized in Table 3.1. For the parametric baselines (K-means and LDA), multiple values for the number of clusters are tested. The baseline results are taken from [179]. Following the same experimental setup, we fix the number of iterations to 10 to ensure a fair comparison, as it is sufficient for convergence. All experiments are run on the Neowise machine, with data evenly distributed across two cores.

The results are reported in Table 3.2. Our proposed distributed method for text clustering, DisCGS, demonstrates strong and consistent performance across all evaluated datasets. DisCGS outperforms or closely matches state-of-the-art baselines, including GSDPMM, K-means, LDA, and GSDMM, in terms of Normalized Mutual Information (NMI). Notably, on the TS dataset, DisCGS achieves the highest average NMI, outperforming all other methods and con-

[1] https://www.grid5000.fr/w/Grid5000:Home

Dataset	n	V	K	$\bar{\ell}_x$	Description
Tweet	2472	5098	89	8.56	Tweet relevance to queries manually annotated as part of the TREC Microblog tracks. [2].
T	11109	8111	152	6.23	Titles of news articles reporting on distinct real-world events [177].
S	11109	18478	152	22.20	Snippets of news articles focused on individual real-world events.
TS	11109	19672	152	28.43	Combined titles and snippets from news articles related to specific events.

Table 3.1: Description of the text datasets used to evaluate the clustering performance of our distributed inference approach. n denotes the number of documents, V the vocabulary size, K the true number of clusters, and $\bar{\ell}_x$ the average document length.

Dataset	K	Clustering Methods				
		DisCGS	GSDPMM	K-means	LDA	GSDMM
Tweet	50	$\underline{0.849} \pm 0.004$	$\mathbf{0.875} \pm 0.005$	0.696 ± 0.008	0.775 ± 0.012	0.844 ± 0.006
	89	0.849 ± 0.004	$\mathbf{0.875} \pm 0.005$	0.725 ± 0.007	0.797 ± 0.011	$\underline{0.862} \pm 0.008$
	150	0.849 ± 0.004	$\mathbf{0.875} \pm 0.005$	0.742 ± 0.006	0.811 ± 0.012	$\underline{0.871} \pm 0.004$
T	100	$\underline{0.847} \pm 0.005$	$\mathbf{0.873} \pm 0.002$	0.687 ± 0.005	0.769 ± 0.012	0.830 ± 0.004
	152	0.847 ± 0.005	$\mathbf{0.873} \pm 0.002$	0.721 ± 0.009	0.784 ± 0.015	$\underline{0.852} \pm 0.009$
	200	0.847 ± 0.005	$\mathbf{0.873} \pm 0.002$	0.730 ± 0.008	0.806 ± 0.013	$\underline{0.868} \pm 0.006$
S	100	$\underline{0.868} \pm 0.004$	$\mathbf{0.891} \pm 0.004$	0.739 ± 0.006	0.848 ± 0.005	0.854 ± 0.004
	152	$\underline{0.868} \pm 0.004$	$\mathbf{0.891} \pm 0.004$	0.756 ± 0.006	0.856 ± 0.002	0.867 ± 0.008
	200	$\underline{0.868} \pm 0.004$	$\mathbf{0.891} \pm 0.004$	0.768 ± 0.007	0.862 ± 0.004	0.885 ± 0.005
TS	100	$\mathbf{0.915 \pm 0.004}$	$\underline{0.912} \pm 0.003$	0.803 ± 0.009	0.867 ± 0.004	0.879 ± 0.009
	152	$\mathbf{0.915 \pm 0.004}$	$\underline{0.912} \pm 0.003$	0.837 ± 0.004	0.881 ± 0.004	0.898 ± 0.004
	200	$\mathbf{0.915 \pm 0.004}$	$\underline{0.912} \pm 0.003$	0.841 ± 0.005	0.904 ± 0.005	0.910 ± 0.003

Table 3.2: The mean and the standard deviation of the NMI over 20 runs on different datasets. The best result within each row is marked in bold, and the runner-up is underlined.

firming its robustness. On the S dataset, DisCGS consistently ranks second, closely trailing GSDPMM.

It is important to emphasize that DisCGS is designed as a distributed approximation of the centralized GSDPMM, and is therefore not expected to consistently outperform it. Rather, the objective is to achieve comparable clustering quality while enabling scalability in distributed environments. The results in Table 3.2 confirm the effectiveness of our approach in preserving clustering performance while benefiting from parallel computation. Moreover, the low variance in NMI scores across runs highlights the robustness and stability of DisCGS.

3.5.3 . Comparison of the distributed and centralized collapsed Gibbs sampler

In this experiment, we compare the execution time and clustering performance of the Distributed Collapsed Gibbs Sampler (DisCGS) with the Centralized Collapsed Gibbs Sampler (GSDPMM) proposed in [179]. Both methods

n	V	K	$\bar{\ell}_x$	Cores
20K	24755	14	7.42	4
40K	48562	14	8.52	4
60K	71834	14	9.48	8
80K	94805	14	10.47	8
100K	116154	14	11.37	8

Table 3.3: Statistics of the text datasets extracted from the DBpedia dataset used to compare the distributed inference approach with the centralized one. The number of cores used in the distributed setting is also indicated for each dataset.

are evaluated on multiple subsets of the DBpedia dataset, each varying in size. DBpedia[3] contains Wikipedia abstracts categorized according to a predefined ontology, where each document represents an entity with an associated label. Table 3.3 summarizes the statistics of these subsets. To ensure consistency, the number of iterations is fixed at 10 for both algorithms.

Figure 3.1 and Table 3.4 compare the clustering performance and computational efficiency of our distributed collapsed Gibbs sampler (DisCGS) with the centralized GSDPMM on DBpedia subsets ranging from 20K to 100K documents. As shown in Figure 3.1, DisCGS significantly reduces computational time compared to GSDPMM. This speed-up becomes more pronounced as the dataset size increases. For example, on the 100K dataset, DisCGS completes 10 iterations in approximately 85 seconds, while GSDPMM requires over 410 seconds, achieving nearly a 5-times improvement. In terms of clus-

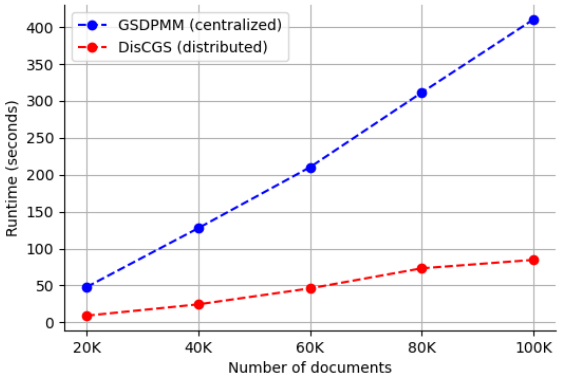

Figure 3.1: Running time of the distributed (DisCGS) and centralized (GSDPMM) inference on DBpedia sub-datasets of increasing sizes.

[3] https://www.dbpedia.org/

tering quality, as shown in Table 3.4, overall, DisCGS achieves competitive clustering quality, often outperforming the centralized method in terms of Adjusted Rand Index, while maintaining comparable Normalized Mutual Information scores. Specifically, DisCGS consistently achieves the best ARI across all dataset sizes, indicating its strength in recovering true partitions even under distributed approximations. For NMI, DisCGS performs as well as, or in some cases slightly below, the centralized approach. These results confirm that the distributed inference not only maintains strong clustering quality but, in some settings, can even surpass the centralized approach.

Dataset size	ARI		NMI	
	Dis.	Cen.	Dis.	Cen.
20K	**0.460**	0.282	**0.606**	0.578
40K	**0.355**	0.259	0.544	**0.590**
60K	**0.519**	0.299	**0.616**	0.613
80K	**0.405**	0.284	**0.612**	0.609
100K	**0.403**	0.345	0.621	**0.659**

Table 3.4: Clustering metrics ARI and NMI obtained by the distributed (Dis.) and centralized (Cen.) inference on DBpedia sub-datasets of increasing sizes.

These findings validate the core objective of DisCGS to efficiently approximate centralized inference without sacrificing clustering accuracy while enabling scalable inference in distributed environments.

3.5.4 . Distributed algorithm scale-up

In this experiment, we extract 300,000 documents from the DBpedia dataset. The resulting vocabulary size is 334,556, and the average document length is 19.89 words. We evaluate the scalability of our distributed inference algorithm by running it multiple times while progressively increasing the number of cores from 2 to 10.

Figure 3.2 illustrates the effect of increasing the number of CPU cores on the runtime of our distributed collapsed Gibbs sampler applied to the DBpedia dataset with 300,000 documents. As the number of cores increases from 2 to 10, the runtime decreases significantly, demonstrating the strong scalability and parallel efficiency of DisCGS. Additionally, clustering quality metrics such as NMI and ARI remain stable (results reported in Appendix 6.9), indicating that parallelization does not compromise inference accuracy. These findings confirm the suitability of DisCGS for large-scale clustering in distributed computing environments.

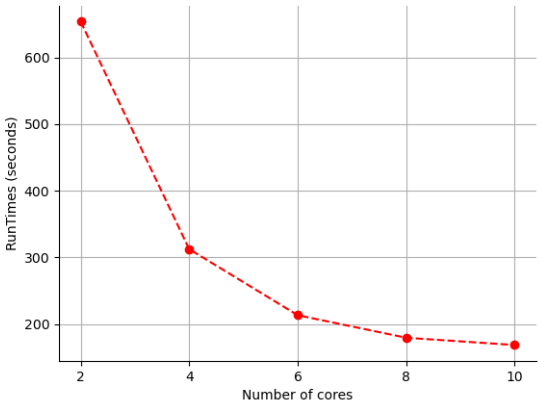

Figure 3.2: Running time (seconds) of DisCGS on a DBpedia dataset with 300,000 documents as a function of the number of cores.

3.6 . Conclusion

While many works on DPMMs assume multivariate Gaussian clusters for their computational convenience, this assumption is not always appropriate, particularly when dealing with discrete data. In this chapter, we extended our distributed inference framework for DPMMs, originally designed for multivariate Gaussian components, to the case of Multinomial components, with a focus on text clustering. Furthermore, we generalized the proposed methodology to handle any component distribution from the exponential family, broadening its applicability to diverse data types and problem domains. Experiments on real-world datasets demonstrate that the proposed approach reduces the inference time compared to the centralized method, while preserving high clustering accuracy.

4 - Distributed Inference for Bayesian Non-Parametric Latent Block Models

In this chapter, we introduce a distributed inference method for the Bayesian Non-Parametric Latent Block Model (DisNPLBM) using a Master/Worker architecture. This approach extends the distributed inference framework originally developed for Dirichlet Process Mixture Models in Chapter 3 to the setting of co-clustering with non-parametric latent block models. In this chapter, we assume that the block component distributions are multivariate Gaussian. The proposed method performs simultaneous clustering of both rows and columns (i.e., co-clustering), where each block is modeled using multivariate Gaussian distributions. The work presented in this chapter has been published in [88].

4.1 . Introduction

Co-clustering aims to simultaneously infer row and column partitions of a data matrix. It often outperforms conventional clustering algorithms, which only focus on inferring a row partition, especially when the data exhibits a dual structure between observations and variables.

Bayesian Non-Parametric Latent Block Models make independent priors on the row and column cluster proportions and a prior on the block component distributions, allowing the number of co-clusters to be automatically inferred during the sampling process. To perform inference in NPLBM, the collapsed Gibbs sampler introduced in [121] is commonly used. This Markov Chain Monte Carlo algorithm iteratively samples the row partition conditioned on the column partition, and vice versa, based on marginal posterior probabilities. However, as stated in 1.6, MCMC inference approaches suffer from slow convergence when applied to large datasets.

To address this in a co-clustering setting when the number of observations is large, we propose a distributed inference algorithm for NPLBM. Our method follows a Master/Worker architecture where the rows are partitioned and distributed evenly across the workers, which do not communicate between them but interact only with the master. Each worker performs the inference of the local row partition given a shared global column partition. It then sends the sufficient statistics of its local row clusters to the master. The master uses these statistics to infer the global row partition and the column partition, which enables the estimation of the global co-clustering structure. The main challenge addressed in our method is how to estimate the global row and column partitions without direct access to the full data, using only

the aggregated sufficient statistics.

Our experimental results demonstrate the effectiveness of the proposed approach in terms of clustering accuracy and computational efficiency. Additionally, we showcase its application to a real-world gene expression dataset, highlighting the practical utility of DisNPLBM in high-dimensional biological data analysis.

The remainder of this chapter is organized as follows. In Section 4.2, we recall the definition of NPLBMs with multivariate Gaussian component distributions. In Section 4.3, we present the distributed inference algorithm, detailing the inference of local row clusters at the worker level and the estimation of global row and column partitions at the master level. Section 4.4 reports empirical results, and Section 4.5 provides concluding remarks.

4.2 . Model definition

Let $X = (x_{i,j})_{n,p} \in \mathbb{R}^{n \times p \times d}$ be the observed dataset. Here, n denotes the number of rows, p represents the number of columns, and d is the dimension of the observation space. Let $\mathbf{z} = (z_i)_{i=1}^{n}$ and $\mathbf{w} = (w_j)_{j=1}^{p}$ be the corresponding row membership and column membership vectors, respectively. In this work, we assume the following Non-Parametric Latent Bloc Model:

$$\begin{gathered}
\pi \sim \mathrm{SB}(\alpha),\ \rho \sim \mathrm{SB}(\beta), \\
\theta_{z_i,w_j} \sim \mathrm{NIW}(\lambda_0),\ z_i\,|\pi \sim \mathrm{Mult}(\pi),\ w_j|\,\rho \sim \mathrm{Mult}(\rho), \\
x_{i,j} \mid \{z_i, w_j, \theta_{z_i,w_j}\} \sim \mathcal{N}\left(\theta_{z_i,w_j}\right).
\end{gathered}$$

Since we assume that the block component distribution is multivariate Gaussian. Thus, each block parameter $\theta_{k,l} = (\mu_{k,l}, \Sigma_{k,l})$, where $\mu_{k,l} \in \mathbb{R}^d$ is the mean vector and $\Sigma_{k,l} \in \mathbb{R}^{d \times d}$ is a positive semi-definite covariance matrix. The base distribution G_0 is chosen as the Normal-Inverse-Wishart (NIW) prior [117], which is conjugate to the multivariate Gaussian. It is parameterized by the hyperparameters $(\mu_0, \kappa_0, \Psi_0, \nu_0)$.

In the co-clustering problem, given the observed dataset X, the prior G_0, and the concentration parameters α and β, the objective is to estimate the row and column partitions, denoted by $\mathbf{z}$ and $\mathbf{w}$. In the following section, we present the distributed inference procedure for NPLBMs in the case where the number of observations n becomes large.

4.3 . Distributed Inference

The main objective of our method is to make the inference process scalable when the number of observations becomes very large, while preserving high clustering accuracy. To achieve this, we adopt a Master/Worker archi-

tecture in which the rows of the data matrix are evenly distributed across the workers. Each iteration of the algorithm alternates between two levels:

- **Worker level:** Given the current global column partition, each worker performs a local row clustering using the collapsed Gibbs sampler, thereby inferring a local NPLBM. This step estimates the row memberships locally and computes the sufficient statistics and sizes for each local row and block cluster. These statistics are then sent to the master.
- **Master level:** The master aggregates the information received from all workers and estimates the global co-clustering structure. First, the global row partition is inferred. This step also updates the posterior hyperparameters of the block distributions. Next, the column partition is updated, allowing the master to infer the global NPLBM. The updated global row and column partitions, along with relevant hyperparameters, are then communicated back to the workers. It is important to emphasize that, at this level, row and column partitions are estimated without direct access to the data, only through the aggregated sufficient statistics received from the workers.

In the following, we detail the inference performed at each stage of the algorithm.

4.3.1 . Worker level

Let E be the number of workers, n^e be the number of rows in worker e, $X^e = (x^e_{i,j})_{n^e \times p} \in \mathbb{R}^{n^e \times p \times d}$ the local dataset in worker e, each cell $x^e_{i,j}$ is a d-dimensional vector. Let $\mathbf{z}^e = (z^e_i)_{n^e}$ be the local row partition (i.e. $z^e_i = k$ means that the i-th row of e-th worker belongs to the k-th local row cluster). At this level, each local row membership z^e_i is updated given the global column membership and other local row memberships $\mathbf{z}^e_{-i} = \{z^e_r | r \neq i\}$ by sampling from $\mathrm{p}(z^e_i | \mathbf{z}^e_{-i}, \mathbf{w}, X^e, G_0, \alpha) \propto$:

$$\begin{cases} \dfrac{n^e_k}{n-1+\alpha} \mathrm{p}\left(x^e_{i,\cdot} \mid \mathbf{w}, \mathbf{x}^e_{k,\cdot}, G_0\right) & \text{existing row cluster } k, \quad (4.1) \\ \dfrac{\alpha}{n-1+\alpha} \mathrm{p}\left(x^e_{i,\cdot} \mid \mathbf{w}, G_0\right), & \text{new row cluster,} \quad (4.2) \end{cases}$$

where n^e_k is the size of local row cluster k in worker e, $x^e_{i,\cdot}$ is the i-th row of worker e, and $\mathbf{x}^e_{k,\cdot} = \{x^e_{i,\cdot} | z^e_i = k\}$ the content of local row cluster k in the worker e. Thanks to the conjugacy assumption, the joint posterior and prior predictive distributions required in Equations (4.1) and (4.2) can be computed in closed form. In fact, we have

$$\mathrm{p}(x^e_{i,\cdot} \mid \mathbf{w}, \mathbf{x}^e_{k,\cdot}, G_0) = \prod_{l=1}^{L} \mathrm{p}(\mathbf{x}^e_{i,l} \mid \lambda^e_{k,l}),$$

$$= \prod_{l=1}^{L} \pi^{\frac{-n_{i,l}^e d}{2}} \frac{(\kappa_{k,l}^e)^{d/2}}{(\kappa_{i,l}^e)^{d/2}} \cdot \frac{\Gamma_d\left(\nu_{i,l}^e/2\right)}{\Gamma_d\left(\nu_{k,l}^e/2\right)} \cdot \frac{\det\left(\Psi_{k,l}^e\right)^{\nu_{k,l}^e/2}}{\det\left(\Psi_{i,l}^e\right)^{\nu_{i,l}^e/2}}, \quad (4.3)$$

where, L denotes the number of inferred column clusters, and $\mathbf{x}_{i,l}^e = \{x_{i,j}^e : w_j = l\}$ represents the subset of elements in row i of worker e that belong to column cluster l. Moreover, $\lambda_{k,l}^e = (\mu_{k,l}^e, \kappa_{k,l}^e, \Psi_{k,l}^e, \nu_{k,l}^e)$ corresponds to the hyperparameters of the posterior component distribution associated with the local block (k,l) of worker e, which are updated as follows

$$\mu_{k,l}^e = \frac{\kappa_0\mu_0 + n_{k,l}^e \bar{\mathbf{x}}_{k,l}^e}{\kappa_{k,l}^e}, \quad \kappa_{k,l}^e = \kappa_0 + n_{k,l}^e, \quad \nu_{k,l}^e = \nu_0 + n_{k,l}^e,$$

$$\Psi_{k,l}^e = \Psi_0 + C + \frac{\kappa_0 n_{k,l}^e}{\kappa_{k,l}^e}\left(\mu_0 - \bar{\mathbf{x}}_{k,l}^e\right)\left(\mu_0 - \bar{\mathbf{x}}_{k,l}^e\right)^T, C = \sum_{x \in \mathbf{x}_{k,l}^e}\left(x - \bar{\mathbf{x}}_{k,l}^e\right)\left(x - \bar{\mathbf{x}}_{k,l}^e\right)^T,$$

where, $\mathbf{x}_{k,l}^e$ denotes the content of the local block (k,l), and $n_{k,l}^e$ its cardinality. The joint prior predictive distribution in Equation (4.1) is given by

$$\begin{aligned} \mathrm{p}(\mathbf{x}_{i,\cdot}^e \mid \mathbf{w}, G_0) &= \prod_{l=1}^{L} \mathrm{p}(\mathbf{x}_{i,l}^e \mid G_0), \\ &= \prod_{l=1}^{L} \pi^{\frac{-dn_{i,l}^e}{2}} \frac{\kappa_0^{d/2}}{(\kappa_{i,l}^e)^{d/2}} \cdot \frac{\Gamma_d\left(\nu_{i,l}^e/2\right)}{\Gamma_d\left(\nu_0/2\right)} \cdot \frac{\det(\Psi_0)^{\nu_0/2}}{\det\left(\Psi_{i,l}^e\right)^{\nu_{i,l}^e/2}}, \end{aligned} \quad (4.4)$$

with $n_{i,l}^e$ the size of $\mathbf{x}_{i,l}^e$ and where the hyperparameters values are given by:

$$\mu_{i,l}^e = \frac{\kappa_0\mu_0 + n_{i,l}^e \bar{\mathbf{x}}_{i,l}^e}{\kappa_{i,l}^e}, \quad \kappa_{i,l}^e = \kappa_0 + n_{i,l}^e, \quad \nu_{i,l}^e = \nu_0 + n_{i,l}^e,$$

$$\Psi_{i,l}^e = \Psi_0 + C + \frac{\kappa_0 n_{i,l}^e}{\kappa_{i,l}^e}\left(\mu_0 - \bar{\mathbf{x}}_{i,l}\right)\left(\mu_0 - \bar{\mathbf{x}}_{i,l}^e\right)^T, C = \sum_{x \in \mathbf{x}_{i,l}^e}\left(x - \bar{\mathbf{x}}_{i,l}^e\right)\left(x - \bar{\mathbf{x}}_{i,l}^e\right)^T.$$

After having updated the local row partition, for a given row cluster k in worker e, for each column j, we compute the following sufficient statistics:

$$T_{k,j}^e = \frac{1}{n_k^e} \sum_{i=1, z_i^e = k}^{n^e} x_{i,j}^e \in \mathbb{R}^d, \quad (4.5)$$

$$S_{k,j}^e = \sum_{i=1, z_i^e = k}^{n^e} (x_{i,j}^e - T_{k,j}^e)(x_{i,j}^e - T_{k,j}^e)^T \in \mathbb{R}^{d \times d}. \quad (4.6)$$

Let K^e denote the number of row clusters inferred by worker e. We denote by $\mathcal{S}^e$ the set of sufficient statistics associated with these clusters, defined as

$$\mathcal{S}^e = \left\{(T_{k,j}^e, S_{k,j}^e) \mid (k,j) \in \{1,\ldots,K^e\} \times \{1,\ldots,p\}\right\},$$

Finally, the sufficient statistics and sizes of each cluster are sent to the master. The DisNPLBM inference process at the worker level is described in Algorithm 7, which represents the Map function.

Algorithm 7 Map function: local inference of the row partition

Input: Dataset X^e,
    Column partition $\mathbf{w}$,
    Concentration parameter α,
    Prior G_0.
For $i \leftarrow 1$ **to** n^e
    Remove $x^e_{i,\cdot}$ from its local row cluster.
    Compute $\mathrm{p}(z^e_i | \mathbf{z}^e_{-i}, \mathbf{w}, X^e, G_0, \alpha)$ ▷ *Using to Equations (4.1) and (4.2)*
    Sample z^e_i ▷ *Draw new local row membership*
    Add $x^e_{i,\cdot}$ to its new local row cluster.
For $k \leftarrow 1$ **to** K^e
    For $j \leftarrow 1$ **to** p
        Compute $T^e_{k,j}$ and $S^e_{k,j}$ ▷ *Using Equations (4.5) and (4.6)*
Output: Updated row partition $\mathbf{z}^e$, sufficient statistics $\mathcal{S}^e$, sizes of each cluster.

4.3.2 . Master level

At this level, the objective is to estimate the global row and column partition given sufficient statistics, local cluster sizes, and the prior. In the following, we detail these two steps:

Global row partition estimation

The global row membership $\mathbf{z}$ is estimated by clustering the local row clusters. Instead of assigning the rows sequentially and individually to a global row cluster, we assign a batch of rows that already share the same local row cluster to a global row cluster. Hence, the rows assigned to the same global row cluster will share the same global row membership. Since the workers operate asynchronously, the results are aggregated using the Reduce function without waiting for all workers to finish their tasks.

Let $\mathcal{S}^{e_1}, S^{e_2}, K^{e_1}$ and K^{e_2} be the sets of sufficient statistics and the number of local row clusters returned by two workers e_1 and e_2 respectively. The goal is to aggregate the local row clusters $\{\mathbf{x}^{e_1}_{1,\cdot}, \ldots, \mathbf{x}^{e_1}_{K^{e_1},\cdot}\}$ and $\{\mathbf{x}^{e_2}_{1,\cdot}, \ldots, \mathbf{x}^{e_2}_{K^{e_2},\cdot}\}$ into a global structure by performing an additional clustering step. To perform such clustering, we proceed as follows: we first set the initial cluster partition equal to the local partition inferred in cluster e_1. Then, for each $h \in$

$\{1, \ldots, K^{e_2}\}$, we sample $z_h^{e_2}$, the membership of $\mathbf{x}_{h,\cdot}^{e_2}$ from $\mathrm{p}(z_h^{e_2} \mid \mathbf{z}_{-h}^{e_2}, X, G_0, \alpha) \propto$

$$\begin{cases} \dfrac{n_k}{n-1+\alpha}\mathrm{p}(\mathbf{x}_{h,\cdot}^{e_2} \mid z_h^{e_2}=k, \mathbf{X}_{k,\cdot}, G_0), & \text{existing row cluster } k, \quad (4.7) \\ \dfrac{\alpha}{n-1+\alpha}\mathrm{p}(\mathbf{x}_{h,\cdot}^{e_2} \mid G_0) & \text{new row cluster,} \quad (4.8) \end{cases}$$

where n_k is the size of global row cluster k, $\mathbf{X}_{k,\cdot}$ the content of global row cluster k, and $\mathbf{z}_{-h}^{e_2} = \{z_{h'}^{e_2} | h' \neq h\}$. The joint posterior and the joint prior predictive distributions needed in Equations (4.7) and (4.8) respectively are computed analytically by only using sufficient statistics, i.e., without having access to the content of local and global clusters:

$$\mathrm{p}\left(\mathbf{x}_{h,\cdot}^{e} \mid G_0\right) = \pi^{-n_h^e \frac{d}{2}} \cdot \frac{\kappa_0^{d/2}}{\left(\kappa_h^e\right)^{d/2}} \cdot \frac{\Gamma_d\left(\nu_h^e/2\right)}{\Gamma_d\left(\nu_0/2\right)} \cdot \frac{\det(\Psi_0)^{\nu_0/2}}{\det\left(\Psi_h^e\right)^{\nu_h^e/2}}$$

where the hyper-parameters $(\mu_h^e, \kappa_h^e, \Psi_h^e, \nu_h^e)$ are obtained using the sufficient statistics:

$$\mu_h^e = \frac{\kappa_0\mu_0 + n_h^e T_h^e}{\kappa_h^e}, \quad \kappa_h^e = \kappa_0 + n_h^e, \quad \nu_h^e = \nu_0 + n_h^e,$$

$$\Psi_h^e = \Psi_0 + S_h^e + \frac{\kappa_0 n_h^e}{\kappa_h^e}\left(\mu_0 - T_h^e\right)\left(\mu_0 - T_h^e\right)^T,$$

where $T_h^e = \frac{1}{p}\sum_{j=1}^{p} T_{h,j}$ and $S_h^e = \frac{1}{p}\sum_{j=1}^{p} S_{h,j}^e$. Moreover, we have

$$\mathrm{p}(\mathbf{x}_{h,\cdot}^{e} \mid z_h^e = k, \mathbf{X}_{k,\cdot}, G_0) = \pi^{\frac{-dn_h^e}{2}} \cdot \frac{\kappa_k^{d/2}}{\left(\kappa_h^e\right)^{d/2}} \cdot \frac{\Gamma_d\left(\nu_h^e/2\right)}{\Gamma_d\left(\nu_k/2\right)} \cdot \frac{|\Psi_k|^{\nu_k/2}}{\left|\Psi_h^e\right|^{\nu_h^e/2}}$$

where the posterior distribution parameters $(\mu_k, \kappa_k, \Psi_k, \nu_k)$ associated to the global cluster k are updated from the prior as follows:

$$\mu_k = \frac{\kappa_0\mu_0 + n_k T_k}{\kappa_k}, \quad \kappa_k = \kappa_0 + n_k, \quad \nu_k = \nu_0 + n_k,$$

$$\Psi_k = \Psi_0 + S_k + \frac{\kappa_0 n_k}{\kappa_k}\left(\mu_0 - T_k\right)\left(\mu_0 - T_k\right)^T,$$

with T_k and S_k the aggregated sufficient statistics when local clusters are assigned to the same global cluster. They are given by:

$$T_k = \frac{1}{n_k}\sum_{e,h|z_h^e=k} n_h^e \cdot T_h^e, \quad (4.9)$$

and

$$S_k = \sum_{e,h|z_h^e=k} S_h^e + \sum_{e,h|z_h^e=k}\left(n_e^h \cdot T_h^e \cdot T_h^{eT}\right) - n_k \cdot T_k \cdot T_k^T. \quad (4.10)$$

This step consists of joining workers' local row clusters in a streaming way. The recursive joining process stops when the global row partition is estimated. If $K^{(e_1,e_2)}$ is the number of inferred global row clusters, then the process stops when $\sum_{k=1}^{K^{(e_1,e_2)}} n_k = n$. The procedure is detailed in the algorithm 8.

Algorithm 8 Reduce Function: global row-partition estimation

Input: Sufficient statistics $\mathcal{S}^{e_1}$, $\mathcal{S}^{e_2}$
  Concentration parameter α,
  Prior G_0.
Initialize global membership $\mathbf{z}$ according to $\mathbf{z}^{e_1}$
For each $h \in K^{e_2}$ **do**:
  Compute $\mathrm{p}(z_h^{e_2} \mid \mathbf{z}_{-h}^{e_2}, X, G_0, \alpha)$ ▷ *Using Equations (4.7) and (4.7).*
  Sample $z_h^{e_2}$ ▷ Draw new global row membership
  Add $\mathbf{x}_h^{e_2}$ to its new global row cluster.
  Update the membership vector $\mathbf{z}$.
For $k \leftarrow 1$ **to** $K^{(e_1,e_2)}$**do**:
  Compute S_k and T_k ▷ *Using Equations (4.9) and (4.10)*
Output: Updated row partition $\mathbf{z}$, aggregated sufficient statistics and clusters sizes.

Column memberships estimation

Given the sufficient statistics $\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^E$, the global row partition $\mathbf{z}$, the prior G_0, and the concentration parameter β, the objective is to update the column partition $\mathbf{w} = (w_j)_p$, each w_j is drawn according to $\mathrm{p}(w_j|\mathbf{w}_{-j}, \mathbf{z}, X, G_0, \beta) \propto$

$$\begin{cases} p_k \mathrm{p}(x_{\cdot,j} \mid \mathbf{z}, \mathbf{w}_{-j}, X_{-j}, G_0, \beta), & \text{existing column cluster } l, \quad (4.11) \\ \beta \mathrm{p}(x_{\cdot,j} \mid \mathbf{z}, G_0) & \text{new column cluster,} \quad (4.12) \end{cases}$$

with $x_{\cdot,j}$ the j-th column and X_{-j} the dataset without column j. Similarly, the joint posterior predictive and the joint prior predictive distributions needed in Equations (4.11) and (4.12) are computed analytically without having access to the columns, but only by using sufficient statistics. In fact, we have:

$$\mathrm{p}\left(x_{\cdot,j} \mid \mathbf{z}, G_0\right) = \prod_{k=1}^{K} \mathrm{p}\left(\mathbf{x}_{k,j} | G_0\right)$$

with K the global number of inferred row clusters, and $\mathbf{x}_{k,j}$ the element of column j that belong to the row cluster k. We have:

$$\mathrm{p}\left(\mathbf{x}_{k,j} | G_0\right) = \pi^{-p_{k,j} \times \frac{d}{2}} \cdot \frac{\kappa_0^{d/2}}{\kappa_{k,j}^{d/2}} \cdot \frac{\Gamma_d(\nu_{k,j}/2)}{\Gamma_d(\nu_0/2)} \cdot \frac{|\Psi_0|^{\nu_0/2}}{|\Psi_{k,j}|^{\nu_{k,j}/2}}$$

with $p_{k,j}$ the cardinal of $\mathbf{x}_{k,j}$. The updated hyper-parameters are obtained with:

$$\mu_{k,j} = \frac{\kappa_0 \mu_0 + p_{k,j} T_{k,j}}{\kappa_{k,j}}, \quad \kappa_{k,j} = \kappa_0 + p_{k,j}, \quad \nu_{k,j} = \nu_0 + p_{k,j},$$
$$\Psi_{k,j} = \Psi_0 + S_{k,j} + \frac{\kappa_0 p_{k,j}}{\kappa_{k,j}} \left(\mu_0 - T_{k,j}\right) \left(\mu_0 - T_{k,j}\right)^T.$$

Moreover, the posterior predictive distribution is computed as follows:

$$\mathrm{p}(x_{.,j} \mid \mathbf{z}, \mathbf{w}_{-j}, X_{-j}, G_0, \beta) = \prod_{k=1}^{K} \mathrm{p}\left(\mathbf{x}_{k,j} | G_{k,l}\right)$$

with $G_{k,l}$ the posterior distribution associated with block (k, l) (i.e., row cluster k and column cluster l). We have:

$$\mathrm{p}\left(\mathbf{x}_{k,j} | G_{k,l}\right) = \pi^{-p_{k,j} \times \frac{d}{2}} \cdot \frac{\kappa_{k,l}^{d/2}}{\kappa_{k,j}^{d/2}} \cdot \frac{\Gamma_d(\nu_{k,j}/2)}{\Gamma_d(\nu_{k,l}/2)} \cdot \frac{\det(\Psi_{k,l})^{\nu_{k,l}/2}}{\det(\Psi_{k,j})^{\nu_{k,j}/2}}$$

with $(\mu_{k,l}, \kappa_{k,l}, \Psi_{k,l}, \nu_{k,l})$ the block posterior distribution parameters given by:

$$\mu_{k,l} = \frac{\kappa_0 \mu_0 + p_{k,l} T_{k,l}}{\kappa_{k,l}}, \quad \kappa_{k,l} = \kappa_0 + p_{k,l}, \quad \nu_{k,l} = \nu_0 + p_{k,l},$$
$$\Psi_{k,l} = \Psi_0 + S_{k,l} + \frac{\kappa_0 p_{k,l}}{\kappa_{k,l}} \left(\mu_0 - T_{k,l}\right) \left(\mu_0 - T_{k,l}\right)^T .$$

With $T_{k,l}$ and $S_{k,l}$, the aggregated sufficient statistics are obtained when local clusters are assigned to the same global block (k, l), and they are computed as follows:

$$T_{k,l} = \frac{1}{p_{k,l}} \sum_{e,h|\, z_h^e = k, w = l} p_{h,l}^e \cdot T_{h,l}^e$$

$$S_{k,l} = \sum_{e,h|\, z_h^e = k, w = l} S_{h,l}^e + \sum_{e,h|\, z_h^e = k, w = l} \left(p_{h,l}^e \cdot T_{h,l}^e \cdot {T_{h,l}^e}^T \right) - p_{k,l} \cdot T_{k,l} \cdot T_{k,l}^T$$

where $p_{k,l}$ is the number of cells in the global cluster (k, l). The column partition update is detailed in algorithm 9.

Algorithm 9 Column partition estimation

Input: Sufficient statistics,
    Global row partition,
    Concentration parameter β,
    prior G_0.
For $j \leftarrow 1$ **to** p **do**:
    Remove $x_{.,j}$ from its column cluster.
    Compute $\mathrm{p}(w_j | \mathbf{w}_{-j}, \mathbf{z}, X, G_0, \beta)$ ▷ *Using Equations (4.11) and (4.12).*
    Sample w_j ▷ *Draw new column membership*
    Add $x_{.,j}$ to its new column cluster.
Output: Column-partition $\mathbf{w}$.

4.4 . Experiments

To evaluate our approach, we conducted several experiments. Firstly, we compare our distributed algorithm with other state-of-the-art co-clustering and clustering algorithms in terms of row clustering performance on synthetic and real-world datasets. Secondly, we compare the execution time and clustering performance of our distributed algorithm, DisNPLBM, and the centralized NPLBM [63] on synthetic datasets with different row sizes. Lastly, we investigate the scalability of DisNPLBM by increasing the number of nodes while keeping the number of rows fixed. The clustering performance is evaluated using the clustering metrics Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI).

4.4.1 . Experiment settings

In the following experiments, we use an uninformative prior NIW as in [63]. Therefore, we set the NIW hyper-parameters as follows: μ_0, and the matrix precision Ψ_0 are respectively set to be empirical mean vector and covariance matrix of all data. κ_0 and ν_0 are set to their lowest values, which are 1 and $d+1$, respectively, where d is the dimension of the observation space. The initial partition consists of a single cluster, and the algorithms run for 100 iterations.

The distributed algorithm is executed on the Neowise machine (1 CPU AMD EPYC 7642, 48 cores/CPU) and Gros machines (1 CPU Intel Xeon Gold 5220, 18 cores/CPU), both hosted by Grid5000[1]. For enhanced portability and deployment flexibility, DisNPLBM is containerized using the Docker image bitnami/spark 3.3.0. We employ Kubernetes for orchestrating Docker images and deploy the Kubernetes cluster on Grid5000 using Terraform[2].

4.4.2 . Clustering performance

We first evaluate the row clustering performance of our algorithm on both synthetic and real-world datasets; we compare its results with two co-clustering algorithms, NPLBM [63] and LBM [64], and two clustering algorithms, K-means and Gaussian mixture model (GMM). We applied the algorithms to 4 datasets: *Synthetic* dataset of size 150×150, generated from 10×3 Gaussian components. *Wine* dataset [4] represents a chemical analysis of three types of wines grown in the same region. The dataset consists of 178 observations, 12 features, and 3 clusters. We also apply the algorithms to two bioinformatics datasets *Chowdary* (104 samples, 182 genes, and 2 clusters) [37] and *Nutt* (22 samples, 1152 genes, and 2 clusters) [125]. Each sample's gene expression level is measured using the Affymetrix technology leading to strictly positive data ranging from 0 to 16000. we apply the Box-Cox transformation [23], to

[1] https://www.grid5000.fr/

[2] https://www.terraform.io/

Dataset		DisNPLBM	NPLBM	LBM	GMM	K-means
Synthetic	ARI	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	0.42 ± 0.03	0.38 ± 0.05	0.39 ± 0.01
	NMI	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	0.78 ± 0.02	0.70 ± 0.02	0.71 ± 0.01
Wine	ARI	0.52 ± 0.03	$\mathbf{0.56 \pm 0.04}$	0.56 ± 0.07	0.51 ± 0.07	0.50 ± 0.04
	NMI	0.59 ± 0.02	$\mathbf{0.65 \pm 0.03}$	0.64 ± 0.03	0.64 ± 0.03	0.64 ± 0.02
Chowdary	ARI	$\mathbf{0.78 \pm 0.01}$	0.07 ± 0.01	0.65 ± 0.00	0.74 ± 0.01	0.75 ± 0.01
	NMI	$\mathbf{0.68 \pm 0.02}$	0.11 ± 0.01	0.58 ± 0.01	0.63 ± 0.01	0.64 ± 0.01
Nutt	ARI	$\mathbf{0.58 \pm 0.01}$	0.56 ± 0.02	0.54 ± 0.04	0.08 ± 0.00	0.11 ± 0.04
	NMI	$\mathbf{0.74 \pm 0.01}$	0.74 ± 0.01	0.68 ± 0.02	0.28 ± 0.02	0.30 ± 0.00

Table 4.1: The mean and the standard deviation of ARI and NMI over 10 runs on different datasets. The best result within each row is marked as bold.

make the data Gaussian-like. Since the number of Genes is much greater than the number of samples we distribute the columns across the workers to achieve scalability, this is legitimate since the row and column clustering are symmetric in our case.

Table 4.1 presents the mean and standard deviation of ARI and NMI across 10 launches for each method on each dataset. Our method outperformed other approaches in the Bioinformatics datasets. Additionally, it has estimated the true clustering structure in the synthetic dataset. While NPLBM and LBM slightly outperform our method on the *Wine* dataset, our approach still yields satisfying results, surpassing traditional methods like GMM and K-means. It's crucial to note that this experiment focuses on comparing clustering performance, without considering execution times due to different inference algorithms. Figure 4.1 illustrates the Heatmaps of Chowdary data before DisNPLBM and reordered data after DisNPLBM. In the recorded data, there is a visible checkerboard pattern distinguishing co-clusters. Co-clustering simultaneously clusters samples and genes, revealing groups of highly correlated genes with distinct correlation structures among different sets of individuals, such as between disease and healthy individuals or different types of disease. This may allow to identify which genes are responsible for some diseases.

4.4.3 . Comparison of the distributed and centralized approaches

We compare the execution times and clustering performance of the distributed and the centralized NPLBM. We execute both algorithms on synthetic datasets of sizes $n \times p \times d$, where $n \in \{20\text{K}, \dots, 100\text{K}\}$, $p = 90$ and $d = 1$ generated from $K \times L$ Gaussian components, with $K = 10$ and $L = 3$ (i.e., $K = 10$ row clusters and $L = 3$ column clusters). We stop at $n = 100$K because the centralized version is too slow; running over 100K observations would take too much time. The distributed algorithm is executed on the Neowise machine in local mode using 24 cores. The centralized algorithm is executed on

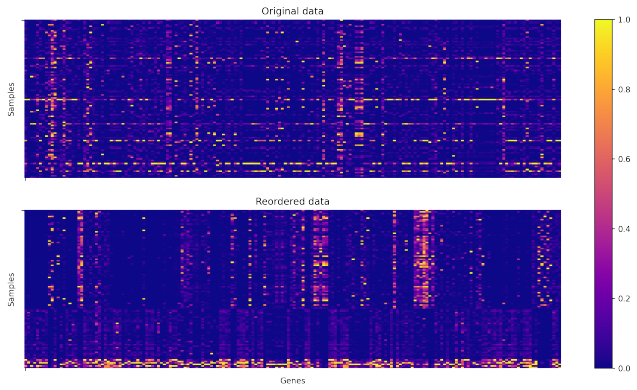

Figure 4.1: Heatmaps of Chowdary data. The first row represents the original data. The second row represents the reordered data after DisNPLBM.

the same machine using one core.

n	ARI		NMI		$\hat{K} \times \hat{L}$		Running time (s)	
	Dis.	Cen.	Dis.	Cen.	Dis.	Cen.	Dis.	Cen.
20K	1.0	1.0	1.0	1.0	30	30	400.21	2265.69
40K	1.0	1.0	1.0	1.0	30	30	693.02	6452.78
60K	1.0	1.0	1.0	1.0	30	30	1122.80	10511.01
80K	1.0	1.0	1.0	1.0	30	30	1373.04	19965.01
100K	1.0	1.0	1.0	1.0	30	30	1572.90	41897.12

Table 4.2: ARI, NMI, number of inferred block clusters ($\hat{K} \times \hat{L}$), and the running time in seconds achieved by the distributed (Dis.) and centralized (Cen.) algorithms.

Table 4.2 reports the clustering metrics ARI, NMI, number of inferred block clusters ($\hat{K} \times \hat{L}$), and the running times obtained by the centralized and distributed inference algorithms on datasets with different row sizes. The results show that our approach considerably reduces the execution time. For example, it is reduced by a factor of 26 for a dataset with 100K rows. On the other hand, we remark that both the distributed and centralized methods performed very well in terms of clustering with values of 1 indicating perfect clustering. Moreover, both methods inferred the true number of clusters. Overall, the distributed approach runs much faster than the centralized method without compromising the clustering performance which makes it more efficient in terms of computational time.

4.4.4 . Distributed Algorithm Scalability

We now investigate the scalability of our approach by increasing the number of cores up to 64 in a distributed computing environment. We employ a dataset with $n = 500$K rows, $p = 20$ columns, and $d = 1$ (representing the observation space dimension). The dataset is generated from $K \times L$ Gaus-

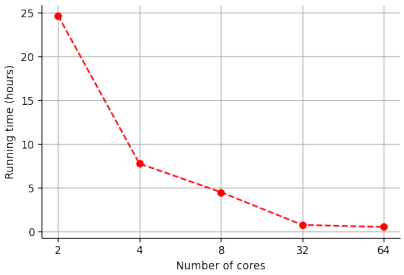

Figure 4.2: Running time (hours) as a function of the number of cores of the distributed approach.

Cores	ARI	NMI	$\hat{K} \times \hat{L}$	Running time (s)
2	1.0	1.0	30	88943.45
4	1.0	1.0	30	27964.73
8	0.99	0.99	30	16202.15
32	0.98	0.99	33	2715.80
64	0.98	0.99	33	1861.85

Table 4.3: ARI, NMI, number of inferred clusters, and the running time in seconds achieved by the distributed approach when distributing on different numbers of cores.

sian components, where $K = 10$ is the number of row clusters and $L = 3$ is the number of column clusters. To conduct this evaluation, we deploy a Kubernetes cluster using up to 6 Gros Machines. Table 4.3 presents clustering metrics ARI and NMI, the number of inferred block clusters, and running time as the number of cores increases. The running time significantly decreases with an increasing number of cores, with the execution time reduced by a factor of 48 when using 64 cores compared to two cores. This demonstrates the efficient scalability of our algorithm with the number of workers. It's worth noting a slight overestimation of the number of clusters with more cores. Additionally, there is a slight decrease in ARI and NMI scores. Nevertheless, our approach still achieves very high clustering metrics and accurately estimates the number of clusters.

4.5 . Conclusion

This chapter presents a novel distributed MCMC inference for NPLBM. NPLBM has the advantage of estimating the number of row and column clusters. However, the inference process becomes too slow when dealing with large datasets. Our proposed method achieves high scalability without compromising the clustering performance. This method can also serve as a basis

component to distribute to multiple co-clustering [62].

5 - Average sensitivity analysis of the EM algorithm of Gaussian Mixture Models

This research was conducted during a five-month international mobility at the National Institute of Informatics (NII)[1], Japan, from October 27, 2024, to March 29, 2025. In this chapter, we study the stability of the Expectation-Maximization algorithm, one of the most widely used inference methods for Gaussian Mixture Models, from the perspective of average sensitivity, which measures how the algorithm's output changes when a random data point is removed. We first show mathematically that the EM algorithm is stable when the mixture proportions are fixed, known, and equal. We then extend this result to the more general case where the mixture proportions are unknown and must be inferred.

5.1 . Introduction

Gaussian Mixture Models play a central role in clustering and are widely applied within data mining. Their parameter estimation is often performed using the Expectation–Maximization algorithm [43], which combines an intuitive iterative structure and robust empirical performance. Importantly, EM is frequently applied even when the data does not strictly follow a Gaussian mixture distribution. Since clustering outcomes often guide decision-making, it is crucial to ensure their stability under missing data or small input perturbations.

Figure 5.1 demonstrates that the EM algorithm may behave unstably in certain pathological scenarios. For example, the estimated means $\hat{\mu}_1$ (red) and $\hat{\mu}_2$ (blue) obtained from the full dataset differ considerably from $\tilde{\mu}_1$ (orange) and $\tilde{\mu}_2$ (light blue) after deleting a subset of points. On the other hand, prior work has established that if the data is generated by a GMM and the correct number of components is used, EM reliably recovers the underlying parameters [40, 173, 98]. This result indicates that EM is robust to random deletions of observations. Hence, the central question of this work is to investigate whether the widely used EM algorithm remains stable under random deletions of data points, even when the data are not generated by a GMM.

Our approach to answering this question is based on the notion of average sensitivity [116, 159], which serves as a rigorous metric for stability. In essence, it measures the expected deviation in the EM-estimated GMM parameters when a single, randomly selected data point is removed. This per-

[1] https://www.nii.ac.jp/en/about/international/mouresearch/internship2024-1/

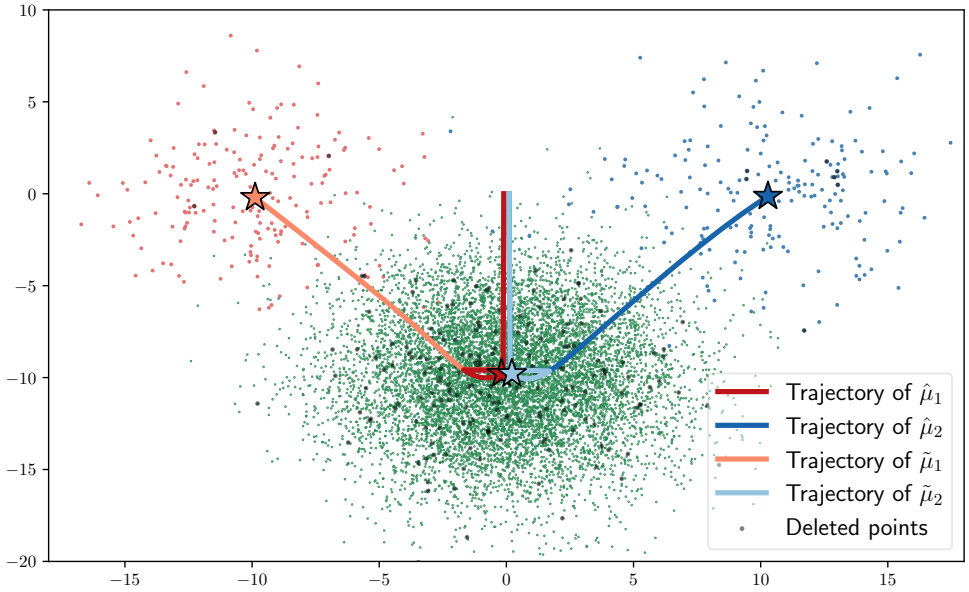

Figure 5.1: The trajectories of the estimated means $\hat{\mu}_1, \hat{\mu}_2$ (resp., $\tilde{\mu}_1, \tilde{\mu}_2$) by the EM algorithm, both before (resp., after) randomly deleting 200 points from a dataset of $n = 10,000$ observations. Stars indicate the final positions of the estimated means. The data is generated from the mixtures of three Gaussians (red, green, and blue) with mixture proportions $\boldsymbol{\pi} = (0.02, 0.96, 0.02)$, means $(\mu_1, \mu_2, \mu_3) = ((-10.0, 0.0), (0.0, -10.0), (10.0, 0.0))$, and variance $\sigma^2 = 10.0$. The EM algorithm is initialized with $K = 2$ components, with initial means set as follows: $\hat{\mu}_1 = \tilde{\mu}_1 = (-0.1, 0.0)$ and $\hat{\mu}_2 = \tilde{\mu}_2 = (0.1, 0.0)$. The initial mixture proportions are $\hat{\boldsymbol{\pi}} = \tilde{\boldsymbol{\pi}} = (0.5, 0.5)$.

spective allows us to evaluate the robustness of EM under typical small perturbations. With this in place, our contributions are threefold.

Our first contribution is a theoretical analysis proving that the EM algorithm exhibits low average sensitivity. Considering the setting with uniform mixture proportions and cluster covariance $\sigma^2 I_d$ for some $\sigma > 0$, we establish that the average sensitivity is $O(K/n)$ when the EM variance parameter is fixed to a sufficiently large constant, where K denotes the number of clusters and n the number of observations. This bound is substantially tighter than the trivial $O(1)$ estimate and, more significantly, independent of the number of iterations m. Such independence is striking, as average sensitivity is generally expected to increase exponentially in m. Indeed, prior work has only focused on one-shot (non-iterative) algorithms to circumvent this difficulty [159, 95].

Second, we extend our analysis to the more general case where the mixture proportions are unknown and must be estimated. In this setting, we establish that the average sensitivity is $O(c^m/n)$, where c is a constant depending on the EM variance parameter. Although this bound grows exponentially

with m, it remains non-trivial since it is smaller than $O(1)$ when $m = o(\log n)$.

Finally, we validate our theoretical findings through comprehensive experiments on both synthetic and real-world datasets. Our results not only deepen the understanding of the EM algorithm's stability but also reassure practitioners that clustering outcomes produced by EM remain robust under slight input perturbations. Moreover, our analysis reveals an important trade-off: increasing σ^2 improves stability but may degrade clustering quality. This trade-off should be carefully considered when tuning the algorithm in practice.

5.2 . Preliminaries and assumptions

Let n, d, and K be positive integers, and define $[n] \coloneqq \{1, \dots, n\}$ and $[K] \coloneqq \{1, \dots, K\}$. For a set S and an element $i \in S$, let $S - i$ denote the set $S \setminus i$. Let $\mathcal{D}(\theta)$ be a probability distribution over $\mathbb{R}^d$ parametrized by θ. For a vector $x \in \mathbb{R}^d$, we write $\mathcal{D}(x \mid \theta)$ to denote its probability density at x. For a vector $\mu \in \mathbb{R}^d$ and a positive semi-definite matrix $\Sigma \in \mathbb{R}^{d \times d}$, we let $\mathcal{N}(\mu, \Sigma)$ denote the Gaussian distribution with mean μ and covariance matrix Σ. In this work, we make the following assumption:

Assumption 1. *We assume that the clusters are spherical and share the same known and diagonal covariance matrix $\sigma^2 I_d$, where $\sigma > 0$ and I_d denotes the $d \times d$ identity matrix.*

It is important to emphasize that this assumption is widely adopted in the literature on the theoretical analysis of the EM algorithm [83, 172, 14], because analyzing the behavior of the EM algorithm under the general setting—where covariance matrices are unknown and distinct—is a highly challenging problem, which significantly complicates the derivation of rigorous theoretical guarantees. However, this generalization constitutes an interesting direction for future research.

5.2.1 . Gaussian Mixture Models

In this section, we review spherical Gaussian mixture models. Let $(\mu_1, \dots, \mu_K)$ be a sequence of vectors in $\mathbb{R}^d$ and $\boldsymbol{\pi} = (\pi_1, \dots, \pi_K)$ be a probability vector (i.e., it verifies $0 \leq \pi_k \leq 1$ for every $k \in [K]$ and $\sum_{k \in [K]} \pi_k = 1$). Then the Gaussian mixture model generates a sequence $\mathbf{x} = (x_1, \dots, x_n)$ of vectors in $\mathbb{R}^d$ as follows:

$$
\begin{aligned}
z_i &\overset{\text{i.i.d.}}{\sim} \mathrm{Cat}\left(\boldsymbol{\pi}\right), && \forall i \in [n], \\
x_i \mid z_i = k &\overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(\mu_k, \sigma^2 I_d\right), && \forall i \in [n].
\end{aligned}
$$

Each observation x_i is sampled first by sampling its membership z_i from a categorical distribution $\mathrm{Cat}(\boldsymbol{\pi})$ parameterized by the mixture weights $\boldsymbol{\pi}$. Then,

given the membership and its corresponding component parameter μ_k, the observation x_i is sampled from the normal distribution $\mathcal{N}(\mu_k, \sigma^2 I_d)$. The distribution $\mathcal{N}(\mu_k, \sigma^2 I_d)$ is called a *component distribution*. According to this definition, the probability density function $p(x)$ of the Gaussian mixture model parameterized by $(\mu_1, \ldots, \mu_K)$ and $\boldsymbol{\pi}$ is given by:

$$p(x) = \sum_{k \in [K]} \pi_k \mathcal{N}(x \mid \mu_k, \sigma^2 I_d).$$

The quantity π_k can be seen as a prior probability of $z_i = k$ (i.e., $\pi_k = \Pr[z_i = k]$). The corresponding posterior probability $r_{ik} = \Pr[z_i = k \mid x_i]$ is called *responsibility* and is given by the Bayes rule:

$$r_{ik} = \frac{\pi_k \mathcal{N}(x_i \mid \mu_k, \sigma^2 I_d)}{\sum_{\ell \in [K]} \pi_\ell \mathcal{N}(x_i \mid \mu_\ell, \sigma^2 I_d)}.$$

We denote by $\theta = (\boldsymbol{\pi}; \mu_1, \ldots, \mu_K)$ the set of all parameters.

5.2.2 . Expectation Maximization Algorithm

The goal of the expectation maximization algorithm is that, assuming that the observed data $\mathbf{x} = (x_1, \ldots, x_n)$ is generated from the Gaussian mixture model, estimates the unknown parameter $\theta = (\boldsymbol{\pi}; \mu_1, \ldots, \mu_K)$. Then, we can recover cluster structure in $\mathbf{x}$ by viewing $\mu_1, \ldots, \mu_K$ as the centroids of the clusters.

First, we observe that the data log-likelihood is given by:

$$\ell(\theta) = \sum_{i \in [n]} \log \left(\sum_{k \in [K]} \pi_k \mathcal{N}(x_i \mid \mu_k, \sigma^2 I_d) \right).$$

Maximization of the log-likelihood of the data is a complex problem due to the presence of the summation over k inside the $\log$ function. The EM algorithm is an alternative to finding the maximum likelihood estimator. Given initial parameters $\hat{\theta}^{(0)} = (\hat{\boldsymbol{\pi}}^{(0)}; \hat{\mu}_1^{(0)}, \ldots, \hat{\mu}_K^{(0)})$, the EM algorithm alternates between two steps at each iteration $m \geq 1$:

1. **E step:** For each pair $(i, k) \in [n] \times [K]$, compute the responsibilities $\hat{r}_{ik}^{(m)}$ as follows:

$$\hat{r}_{ik}^{(m)} = \frac{\hat{\pi}_k^{(m-1)} \mathcal{N}\left(x_i \mid \hat{\mu}_k^{(m-1)}, \sigma^2 I_d\right)}{\sum_{\ell \in [K]} \hat{\pi}_\ell^{(m-1)} \mathcal{N}\left(x_i \mid \hat{\mu}_\ell^{(m-1)}, \sigma^2 I_d\right)}.$$

2. **M step:** For each $k \in [K]$, compute the parameters $\hat{\pi}_k^{(m)}$ and $\hat{\mu}_k^{(m)}$ associated with each component k as follows:

$$\hat{\pi}_k^{(m)} = \frac{\hat{r}_k^{(m)}}{n},$$

$$\hat{\mu}_k^{(m)} = \frac{1}{\hat{r}_k^{(m)}} \sum_{i \in [n]} \hat{r}_{ik}^{(m)} x_i,$$

where $\hat{r}_k := \sum_{i \in [n]} \hat{r}_{ik}$.

The EM algorithm monotonically increases the log-likelihood of the data until it reaches a local maximum.

5.2.3 . Average Sensitivity

Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a data consisting of n vectors. For $j \in [n]$, let $\mathbf{x}_{-j}$ denote the data obtained from $\mathbf{x}$ by deleting x_j. The EM algorithm with m iterations run on $\mathbf{x}_{-j}$ will output the parameters $\tilde{\theta}^{(m)} = \{\tilde{\boldsymbol{\pi}}^{(m)}; \tilde{\mu}_1^{(m)}, \ldots, \tilde{\mu}_K^{(m)}\}$. It is important to emphasize that $\tilde{\theta}^{(m)}$ and $\tilde{r}_{ik}^{(m)}$ depend on j (i.e., $\tilde{\theta}^{(m)} = \tilde{\theta}^{(m)}(j)$ and $\tilde{r}_{ik}^{(m)} = \tilde{r}_{ik}^{(m)}(j)$). The notation of this dependency will be omitted if the deleted observation x_j is clear from the context. Then, we define the *average sensitivity* for the parameters of the EM algorithm as:

$$\frac{1}{n} \sum_{j \in [n]} \sum_{k \in [K]} D\left(\hat{\theta}_k^{(m)}, \tilde{\theta}_k^{(m)}(j)\right),$$

where

$$\begin{aligned}
\hat{\theta}_k^{(m)} &:= \left(\hat{\pi}_k^{(m)}; \hat{\mu}_k^{(m)}\right) \\
\tilde{\theta}_k^{(m)} &:= \left(\tilde{\pi}_k^{(m)}; \tilde{\mu}_k^{(m)}\right) \\
D\left(\hat{\theta}_k^{(m)}, \tilde{\theta}_k^{(m)}(j)\right) &:= \left\| \hat{\pi}_k^{(m)} \cdot \hat{\mu}_k^{(m)} - \tilde{\pi}_k^{(m)}(j) \cdot \tilde{\mu}_k^{(m)}(j) \right\|.
\end{aligned}$$

In the simplest case where $\hat{\pi}_k^{(m)} = \tilde{\pi}_k^{(m)}(j) = 1/K$ for every $k \in [K]$, the distance $D\left(\hat{\theta}_k^{(m)}, \tilde{\theta}_k^{(m)}(j)\right)$ is equal to $\|\hat{\mu}_k^{(m)} - \tilde{\mu}_k^{(m)}(j)\|/K$. Hence, we simply consider the stability of the estimated mean vector when deleting an observation x_j.

In the general case, we give more weight to clusters with larger proportions and ignore perturbations of cluster centers when the corresponding clusters are small. We note that by the triangle inequality

$$D\left(\hat{\theta}_k^{(m)}, \tilde{\theta}_k^{(m)}(j)\right) \leq \left|\hat{\pi}_k^{(m)} - \tilde{\pi}_k^{(m)}\right| \cdot \left\|\hat{\mu}_k^{(m)}\right\| + \tilde{\pi}_k^{(m)}(j) \cdot \left\|\hat{\mu}_k^{(m)} - \tilde{\mu}_k^{(m)}(j)\right\|.$$

Hence, when $\hat{\pi}_k^{(m)}$ and $\tilde{\pi}_k^{(m)}$ are close, and $\hat{\mu}_k^{(m)}$ and $\tilde{\mu}_k^{(m)}(j)$ are close, then the average sensitivity is small.

In practice, random initialization is commonly used across different runs of the EM algorithm. However, in our setting, allowing different initializations for $\hat{\theta}_k^{(0)}$ and $\tilde{\theta}_k^{(0)}$ implies shifting the cluster centers, leading to a label-switching problem. This makes the analysis of average sensitivity infeasible, as the clustering outputs may only match up to an unknown permutation. To address

this and make the average sensitivity analysis tractable, we introduce the following assumption:

Assumption 2. *We assume that $\hat{\theta}^{(0)}$ and $\tilde{\theta}^{(0)}$ are arbitrary but fixed and equal (i.e., $\hat{\theta}^{(0)} = \tilde{\theta}^{(0)}$) across different runs.*

This assumption implies that for all $k \in [K]$ and $j \in [n]$, we have $\hat{\pi}_k^{(0)} = \tilde{\pi}_k^{(0)}(j)$ and $\hat{\mu}_k^{(0)} = \tilde{\mu}_k^{(0)}(j)$.

In this work, we analyze the average sensitivity of the EM algorithm for GMMs without assuming that the observed data are i.i.d. samples from the mixture distribution. Instead, we consider a non-probabilistic setting in which the data points are assumed to lie within a ball of radius $R > 0$. It is important to note that in this case, R does not depend on n and thus can be treated as a universal constant. Furthermore, because the data can be rescaled without loss of generality, we assume that all observed points lie within the unit ball. This leads to the following assumption:

Assumption 3. *We assume that the data points $x_1, \ldots, x_n$ lie within the unit ball, i.e., for all $i \in [n]$, $|x_i| \leq 1$.*

5.3 . Uniform Mixture Proportions

In this section, we analyze the average sensitivity of the EM algorithm for the case where the mixture proportions are uniform, i.e., $\pi_1 = \cdots = \pi_K = 1/K$, and known. Hence, the goal of the EM algorithm is to estimate the means $\mu_1, \ldots, \mu_K \in \mathbb{R}^d$. The goal of this section is to show the following:

Theorem 5.3.1. *Suppose that the mixture weight $\boldsymbol{\pi} = \mathbf{1}/K$ is known and that Assumptions 2 and 3 hold. Then, the average sensitivity of the EM algorithm with m iterations is at most*

$$\frac{2K}{n} \cdot \frac{(1-(2c)^m)}{1-2c},$$

where $c = \exp\bigl(4/\sigma^2\bigr) - 1$.

As $\hat{\mu}_k$ and $\tilde{\mu}_k$ lie within the unit ball, the trivial upper bound on the average sensitivity is 2. When σ^2 is sufficiently large such that $c < 1/2$ ($\sigma > 2/\sqrt{\log(3/2)}$ suffices), the average sensitivity bound is at most $2K/(1-2c)n$. This is significantly smaller than the trivial upper bound of 2, and remarkably, it is independent of the number of iterations m.

Conversely, when σ^2 is small such that $c > 1/2$, the average sensitivity bound is at most $2K \cdot (2c)^m/n$, which grows exponentially with m. However, this bound is smaller than the trivial bound of 2 whenever $m = o(\log_c(n/K))$.[2]

[2]When $c \to 1/2$, the bound of Theorem 5.3.1 converges to $2Km/n$.

Before proving Theorem 5.3.1, we first recall how the EM algorithm updates relevant parameters for the current setting. Given the initial parameters $(\hat{\mu}_1^{(0)}, \ldots, \hat{\mu}_K^{(0)})$, at each iteration m, the EM algorithm, run on the dataset $\mathbf{x} = \{x_1, \ldots, x_n\}$, performs the following two steps:

1. **E step:** For each pair $(i, k) \in [n] \times [K]$ set:

$$\begin{aligned}\hat{r}_{ik}^{(m)} &= \frac{\mathcal{N}(x_i \mid \hat{\mu}_k^{(m-1)}, \sigma^2 I_d)}{\sum_{\ell \in [K]} \mathcal{N}(x_i \mid \hat{\mu}_\ell^{(m-1)}, \sigma^2 I_d)} \\ &= \frac{\exp\left(-\frac{1}{2\sigma^2}\|x_i - \hat{\mu}_k^{(m-1)}\|^2\right)}{\sum_{\ell \in [K]} \exp\left(-\frac{1}{2\sigma^2}\|x_i - \hat{\mu}_\ell^{(m-1)}\|^2\right)}\end{aligned}$$

2. **M step:** For each $k \in [K]$, update the parameter $\hat{\mu}_k^{(m-1)}$ associated with the component k as follows:

$$\hat{\mu}_k^{(m)} = \frac{1}{\hat{r}_k^{(m)}} \sum_{i \in [n]} \hat{r}_{ik}^{(m)} x_i.$$

Given $j \in [n]$, the EM algorithm performs similar steps when it is run on $\mathbf{x}_{-j}$ and outputs $(\tilde{\mu}_1^{(m)}, \ldots, \tilde{\mu}_K^{(m)})$ and $(\tilde{r}_{ik}^{(m)})_{i,k}$ at the m-th iteration. For readability, we omit the iteration superscript indices if they are clear from the context.

Our proof is by induction on m. Let $\Delta\mu_k^{(m)} := \|\hat{\mu}_k^{(m)} - \tilde{\mu}_k^{(m)}\|$ and $\Delta r_{ik}^{(m)} := |\hat{r}_{ik}^{(m)} - \tilde{r}_{ik}^{(m)}|$. We first bound $\Delta\mu_k^{(m)}$ using $\Delta r_{ik}^{(m)}$.

Lemma 4. *For any $k \in [K]$ and a positive integer m, we have*

$$\Delta\mu_k^{(m)} \leq 2\left(\frac{\hat{r}_{jk}^{(m)}}{\hat{r}_k^{(m)}} + \max_{i \in [n]-j} \frac{\Delta r_{ik}^{(m)}}{\hat{r}_{ik}^{(m)}}\right).$$

Proof. We omit the superscript (m) for readability. Note that

$$\begin{aligned}\Delta\mu_k &= \left\| \frac{1}{\hat{r}_k} \sum_{i \in [n]} \hat{r}_{ik} x_i - \frac{1}{\tilde{r}_k} \sum_{i \in [n]-j} \tilde{r}_{ik} x_i \right\| \\ &= \frac{1}{\hat{r}_k \tilde{r}_k} \left\| \sum_{i \in [n]} \tilde{r}_k \hat{r}_{ik} x_i - \sum_{i \in [n]-j} \hat{r}_k \tilde{r}_{ik} x_i \right\| \\ &= \frac{1}{\hat{r}_k \tilde{r}_k} \left\| \tilde{r}_k \hat{r}_{jk} x_j + \sum_{i \in [n]-j} \left(\tilde{r}_k \hat{r}_{ik} - \hat{r}_k \tilde{r}_{ik}\right) x_i \right\| \\ &\leq \frac{1}{\hat{r}_k \tilde{r}_k} \|\tilde{r}_k \hat{r}_{jk} x_j\| + \frac{1}{\hat{r}_k \tilde{r}_k} \left\| \sum_{i \in [n]-j} \left(\tilde{r}_k \hat{r}_{ik} - \hat{r}_k \tilde{r}_{ik}\right) x_i \right\|\end{aligned}$$

$$
\begin{aligned}
&\leq \frac{\hat{r}_{jk}}{\hat{r}_k} \|x_j\| + \frac{1}{\hat{r}_k \tilde{r}_k} \left\| \sum_{i \in [n]-j} (\tilde{r}_k \hat{r}_{ik} - \hat{r}_k \tilde{r}_{ik})\, x_i \right\| \\
&\leq \frac{\hat{r}_{jk}}{\hat{r}_k} \|x_j\| + \frac{1}{\hat{r}_k \tilde{r}_k} \sum_{i \in [n]-j} |\tilde{r}_k \hat{r}_{ik} - \hat{r}_k \tilde{r}_{ik}| \, \|x_i\| \\
&\leq \frac{\hat{r}_{jk}}{\hat{r}_k} + \frac{1}{\hat{r}_k \tilde{r}_k} \sum_{i \in [n]-j} |\tilde{r}_k \hat{r}_{ik} - \hat{r}_k \tilde{r}_{ik}| \qquad \text{(by } \|x_i\| \leq 1 \text{ for all } i \in [n]\text{)} \\
&= \frac{\hat{r}_{jk}}{\hat{r}_k} + \frac{1}{\hat{r}_k \tilde{r}_k} \sum_{i \in [n]-j} |(\tilde{r}_k (\hat{r}_{ik} - \tilde{r}_{ik}) + (\tilde{r}_k - \hat{r}_k) \tilde{r}_{ik})| \\
&\leq \frac{\hat{r}_{jk}}{\hat{r}_k} + \frac{1}{\hat{r}_k \tilde{r}_k} \sum_{i \in [n]-j} \tilde{r}_k \, |\hat{r}_{ik} - \tilde{r}_{ik}| + \frac{|\tilde{r}_k - \hat{r}_k|}{\hat{r}_k \tilde{r}_k} \sum_{i \in [n]-j} \tilde{r}_{ik} \\
&= \frac{\hat{r}_{jk}}{\hat{r}_k} + \frac{1}{\hat{r}_k} \sum_{i \in [n]-j} |\hat{r}_{ik} - \tilde{r}_{ik}| + \left| \frac{\hat{r}_k - \tilde{r}_k}{\hat{r}_k} \right|. \qquad (5.1)
\end{aligned}
$$

Using the fact that $\hat{r}_k \geq \sum_{i \in [n]-j} \hat{r}_{ik}$ and for any $u_1, \ldots, u_n \geq 0$ and $v_1, \ldots, v_n > 0$, we have

$$
\frac{\sum_{i \in [n]} u_i}{\sum_{i \in [n]} v_i} \leq \max_{i \in [n]} \frac{u_i}{v_i},
$$

we obtain

$$
\frac{1}{\hat{r}_k} \sum_{i \in [n]-j} |\hat{r}_{ik} - \tilde{r}_{ik}| \leq \frac{\sum_{i \in [n]-j} |\hat{r}_{ik} - \tilde{r}_{ik}|}{\sum_{i \in [n]-j} \hat{r}_{ik}} \leq \max_{i \in [n]-j} \frac{|\hat{r}_{ik} - \tilde{r}_{ik}|}{\hat{r}_{ik}} \qquad (5.2)
$$

and

$$
\frac{|\hat{r}_k - \tilde{r}_k|}{\hat{r}_k} \leq \frac{\hat{r}_{jk}}{\hat{r}_k} + \frac{\sum_{i \in [n]-j} |\hat{r}_{ik} - \tilde{r}_{ik}|}{\sum_{i \in [n]-j} \hat{r}_{ik}} \leq \frac{\hat{r}_{jk}}{\hat{r}_k} + \max_{i \in [n]-j} \frac{|\hat{r}_{ik} - \tilde{r}_{ik}|}{\hat{r}_{ik}}. \qquad (5.3)
$$

Thus, from Inequalities 1 to 3, we have:

$$
\Delta\mu_k \leq 2 \left(\frac{\hat{r}_{jk}}{\hat{r}_k} + \max_{i \in [n]-j} \frac{|\hat{r}_{ik} - \tilde{r}_{ik}|}{\hat{r}_{ik}} \right). \qquad \square
$$

Let $\Delta\mu_{\max}^{(m)} := \max_{k \in [K]} \Delta\mu_k^{(m)}$. Next, we bound $\Delta r_{ik}^{(m)}$ by using $\Delta\mu_{\max}^{(m-1)}$.

Lemma 5. *For any $k \in [K]$, a positive integer m, and $i \in [n]$, we have*

$$
\frac{\Delta r_{ik}^{(m)}}{\hat{r}_{ik}^{(m)}} \leq \left(\exp\left(\frac{4}{\sigma^2} \right) - 1 \right) \Delta\mu_{\max}^{(m-1)}.
$$

Proof. In this proof, we drop the superscript (m) from $\hat{r}_{ik}^{(m)}$ and $\tilde{r}_{ik}^{(m)}$ and $(m-1)$ from $\hat{\mu}_k^{(m-1)}$, $\tilde{\mu}_k^{(m-1)}$, and $\Delta\mu_{\max}^{(m-1)}$ for readability. For $i \in [n] - j$, let

$$
\hat{\eta}_{ik} := \frac{1}{2\sigma^2} \|x_i - \hat{\mu}_k\|^2,
$$

and

$$\tilde{\eta}_{ik} := \frac{1}{2\sigma^2}\|x_i - \tilde{\mu}_k\|^2.$$

Then, we have

$$\begin{aligned}
\Delta\eta_{ik} &:= \hat{\eta}_{ik} - \tilde{\eta}_{ik} \\
&= \frac{1}{2\sigma^2}\left(\|x_i - \hat{\mu}_k\| + \|x_i - \tilde{\mu}_k\|\right) \cdot \left(\|x_i - \hat{\mu}_k\| - \|x_i - \tilde{\mu}_k\|\right) \\
&\leq \frac{1}{2\sigma^2}(2\|x_i\| + \|\tilde{\mu}_k + \hat{\mu}_k\|) \cdot \|\hat{\mu}_k - \tilde{\mu}_k\| \\
&\leq \frac{2}{\sigma^2}\Delta\mu_k \qquad (5.4) \\
&\leq \frac{2}{\sigma^2}\Delta\mu_{\max}. \qquad (5.5)
\end{aligned}$$

Similarly noting that $-\Delta\eta_{ik} \leq \frac{2}{\sigma^2}\Delta\mu_{\max}$. We obtain

$$-\frac{2}{\sigma^2}\Delta\mu_{\max} \leq \Delta\eta_{ik} \leq \frac{2}{\sigma^2}\Delta\mu_{\max} \qquad (5.6)$$

Then, noting that $\forall \ell \in [K]$ (thus particularly for $\ell = k$), we have

$$\exp(-\tilde{\eta}_{il}) = \exp(-\hat{\eta}_{il} + \Delta\eta_{il}).$$

From Equation (5.6), we have

$$\begin{aligned}
&\exp\left(-\frac{4}{\sigma^2}\Delta\mu_{\max}\right)\frac{\exp\left(-\hat{\eta}_{ik}\right)}{\sum_{\ell\in[K]}\exp(-\hat{\eta}_{il})} \\
&\leq \frac{\exp(-\tilde{\eta}_{ik})}{\sum_{\ell\in[K]}\exp(-\tilde{\eta}_{il})} \leq \exp\left(\frac{4}{\sigma^2}\Delta\mu_{\max}\right)\frac{\exp(-\hat{\eta}_{ik})}{\sum_{\ell\in[K]}\exp(-\hat{\eta}_{il})}
\end{aligned}$$

Therefore

$$\exp\left(-\frac{4\Delta\mu_{\max}}{\sigma^2}\right)\hat{r}_{ik} \leq \tilde{r}_{ik} \leq \exp\left(\frac{4\Delta\mu_{\max}}{\sigma^2}\right)\hat{r}_{ik}$$

Hence, since for each pair $(i,k) \in [n] \times [K]$, we have $\hat{r}_{ik} > 0$, we have

$$\exp\left(-\frac{4\Delta\mu_{\max}}{\sigma^2}\right) \leq \frac{\tilde{r}_{ik}}{\hat{r}_{ik}} \leq \exp\left(\frac{4\Delta\mu_{\max}}{\sigma^2}\right) \qquad (5.7)$$

On one hand, we have

$$\frac{\tilde{r}_{ik}}{\hat{r}_{ik}} - 1 \leq \exp\left(\frac{4\Delta\mu_{\max}}{\sigma^2}\right) - 1 \qquad (5.8)$$

On the other hand, we have

$$1 - \frac{\tilde{r}_{ik}}{\hat{r}_{ik}} \leq 1 - \exp\left(-\frac{4\Delta\mu_{\max}}{\sigma^2}\right) \qquad (5.9)$$

Therefore

$$\begin{aligned}\frac{\Delta r_{ik}}{\hat{r}_{ik}} &= \left|1 - \frac{\tilde{r}_{ik}}{\hat{r}_{ik}}\right| \\ &\leq \max\left\{1 - \exp\left(-\frac{4\Delta\mu_{\max}}{\sigma^2}\right), \exp\left(\frac{4\Delta\mu_{\max}}{\sigma^2}\right) - 1\right\} \\ &\leq \max\left\{\frac{4\Delta\mu_{\max}}{\sigma^2}, \exp\left(\frac{4\Delta\mu_{\max}}{\sigma^2}\right) - 1\right\} \\ &\qquad\qquad (1 - \exp(-x) \leq x \text{ for any } x \in \mathbb{R}) \\ &\leq \exp\left(\frac{4\Delta\mu_{\max}}{\sigma^2}\right) - 1.\end{aligned}$$

We obtain the claim by using the inequality $\exp(Rx) - 1 \leq (\exp(R) - 1)x$ whenever $x \in [0, 1]$. □

Proof of Theorem 5.3.1. Recall $c = \exp\bigl(4/\sigma^2\bigr) - 1$. Fix $k \in [K]$. Combining Lemmas 4 and 5, we obtain

$$\Delta\mu_{\max}^{(m)} \leq 2 \max_{k\in[K]} \frac{\hat{r}_{jk}^{(m)}}{\hat{r}_k^{(m)}} + 2c\Delta\mu_{\max}^{(m-1)}.$$

Now, we bound $\Delta\mu_{\max}^{(m)}$ by an inductive argument. First, we have

$$\begin{aligned}\Delta\mu_{\max}^{(m)} &\leq 2 \max_{k\in[K]} \frac{\hat{r}_{jk}^{(m)}}{\hat{r}_k^{(m)}} + 2c\Delta\mu_{\max}^{(m-1)} \\ &\vdots \\ &\leq 2 \sum_{t\in[m]} (2c)^{m-t} \max_{k\in[K]} \frac{\hat{r}_{jk}^{(t)}}{\hat{r}_k^{(t)}} + (2c)^m \Delta\mu_{\max}^{(0)}.\end{aligned}$$

By assumption, we have $\hat{\mu}_k^{(0)} = \tilde{\mu}_k^{(0)}$, and hence we have

$$\begin{aligned}\Delta\mu_{\max}^{(m)} &\leq 2 \sum_{t\in[m]} (2c)^{m-t} \max_{k\in[K]} \frac{\hat{r}_{jk}^{(t)}}{\hat{r}_k^{(t)}} \\ &\leq 2 \sum_{t\in[m]} (2c)^{m-t} \sum_{k\in[K]} \frac{\hat{r}_{jk}^{(t)}}{\hat{r}_k^{(t)}}. \qquad (5.10)\end{aligned}$$

We remind that the m-th iterate $\tilde{\mu}_k^{(m)}$ of the EM algorithm on $\mathbf{x}_{-j}$ can be viewed as a function of j (and $\Delta\mu_{\max}^{(m)}$ as well). Then from Equation (5.10), the average sensitivity can be bounded as

$$\frac{1}{Kn} \sum_{j\in[n]} \sum_{k\in[K]} \Delta\mu_k^{(m)}(j) \leq \frac{1}{n} \sum_{j\in[n]} \max_{k\in[K]} \Delta\mu_k^{(m)}(j),$$

$$
\begin{aligned}
&= \frac{1}{n} \sum_{j \in [n]} \Delta\mu_{\max}^{(m)}(j), \\
&\leq \frac{2}{n} \sum_{j \in [n]} \sum_{t \in [m]} \sum_{k \in [K]} (2c)^{m-t} \frac{\hat{r}_{jk}^{(t)}}{\hat{r}_k^{(t)}}, \\
&\leq \frac{2}{n} \sum_{k \in [K]} \sum_{t \in [m]} (2c)^{m-t} \sum_{j \in [n]} \frac{\hat{r}_{jk}^{(t)}}{\hat{r}_k^{(t)}}, \\
&\leq \frac{KC}{n},
\end{aligned}
$$

where in the last inequality, we have used the fact that

$$
\sum_{j \in [n]} \frac{\hat{r}_{jk}^{(t)}}{\hat{r}_k^{(t)}} = 1,
$$

and

$$
C := 2 \sum_{t \in [m]} (2c)^{m-t} = \frac{2(1 - (2c)^m)}{1 - 2c}.
$$

Hence, the claim follows. □

5.4 . Unknown Mixture Proportions

In this section, we analyze the average sensitivity of the EM algorithm for the case where the mixture proportions are unknown. Hence, the goal of the EM algorithm is to estimate $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$ and the means $\mu_1, \ldots, \mu_K \in \mathbb{R}^d$. The goal of this section is to show the following:

Theorem 5.4.1. *Suppose that Assumptions 2 and 3 hold. Then, the average sensitivity of the EM algorithm with m iterations is at most*

$$
\frac{2}{n} \frac{c^m - 1}{c - 1},
$$

where

$$
c = \left(\exp\left(\frac{2}{\sigma^2} \right) + \exp\left(\frac{4}{\sigma^2} \right) \right) \cdot \left(1 + \frac{4}{\sigma^2} \right) > 2.
$$

Although the bound is exponential in m, it is significantly smaller than the trivial upper bound of 2 (when $m = o(\log_c n)$). Before proving Theorem 5.4.1, for notational convenience, we let

$$
\begin{aligned}
D_k^{(m)} &:= D\left(\hat{\theta}_k^{(m)}, \tilde{\theta}_k^{(m)} \right), \\
&= \left\| \hat{\pi}_k^{(m)} \cdot \hat{\mu}_k^{(m)} - \tilde{\pi}_k^{(m)} \cdot \tilde{\mu}_k^{(m)} \right\|,
\end{aligned}
$$

and

$$\Delta\pi_k^{(m)} := \left|\hat{\pi}_k^{(m)} - \tilde{\pi}_k^{(m)}\right|.$$

For $i \in [n] - j$ and $k \in [K]$, let

$$\Delta r_{ik}^{(m)} := |\hat{r}_{ik}^{(m)} - \tilde{r}_{ik}^{(m)}|.$$

Refer to Section 5.2.2 for how the parameters are updated through the EM algorithm.

5.4.1 . Recursive Bound on Average Sensitivity

First, we bound $D_k^{(m)}$ using $\Delta r_{ik}^{(m)}$.

Lemma 6. *For any $k \in [K]$ and a positive integer m, we have*

$$D_k^{(m)} = \frac{\hat{r}_{jk}^{(m)}}{n} + \frac{\tilde{\pi}_k^{(m)}}{n} + \frac{1}{n}\sum_{i\in[n]-j} \Delta r_{ik}^{(m)}.$$

Proof. We omit the superscript (m) for readability. We have

$$\begin{aligned}
D_k^{(m)} &= \left\| \frac{\hat{r}_k}{n} \cdot \frac{1}{\hat{r}_k} \sum_{i\in[n]} \hat{r}_{ik} x_i - \frac{\tilde{r}_k}{n-1} \cdot \frac{1}{\tilde{r}_k} \sum_{i\in[n]-j} \tilde{r}_{ik} x_i \right\| \\
&= \left\| \frac{1}{n} \sum_{i\in[n]} \hat{r}_{ik} x_i - \frac{1}{n-1} \sum_{i\in[n]-j} \tilde{r}_{ik} x_i \right\| \\
&\le \frac{\hat{r}_{jk}}{n} + \left\| \frac{1}{n} \sum_{i\in[n]-j} \hat{r}_{ik} x_i - \frac{1}{n-1} \sum_{i\in[n]-j} \tilde{r}_{ik} x_i \right\| \\
&= \frac{\hat{r}_{jk}}{n} + \left\| \sum_{i\in[n]-j} \left(\frac{\hat{r}_{ik}}{n} - \frac{\tilde{r}_{ik}}{n-1} \right) x_i \right\| \\
&\le \frac{\hat{r}_{jk}}{n} + \frac{\sum_{i\in[n]-j} \tilde{r}_{ik}}{n(n-1)} + \frac{1}{n} \left\| \sum_{i\in[n]-j} \left(\hat{r}_{ik} - \tilde{r}_{ik}\right) x_i \right\| \\
&= \frac{\hat{r}_{jk}}{n} + \frac{\tilde{\pi}_k}{n} + \frac{1}{n} \left\| \sum_{i\in[n]-j} \left(\hat{r}_{ik} - \tilde{r}_{ik}\right) x_i \right\| \\
&\le \frac{\hat{r}_{jk}}{n} + \frac{\tilde{\pi}_k}{n} + \frac{1}{n} \sum_{i\in[n]-j} \left\| \left(\hat{r}_{ik} - \tilde{r}_{ik}\right) x_i \right\| \\
&\le \frac{\hat{r}_{jk}}{n} + \frac{\tilde{\pi}_k}{n} + \frac{1}{n} \sum_{i\in[n]-j} \Delta r_{ik}, && \text{(by } \|x_i\| \le 1\text{)}
\end{aligned}$$

and the claim follows. □

5.4.2 . Recursive bound on Mixture Proportions

Next, we bound $\Delta\pi_k^{(m)}$ using $\Delta r_{ik}^{(m)}$.

Lemma 7. *We have*

$$\Delta\pi_k^{(m)} \leq \frac{\hat{r}_{jk}^{(m)}}{n} + \frac{\tilde{\pi}_k^{(m)}}{n} + \frac{1}{n}\sum_{i\in[n]-j} \Delta r_{ik}^{(m)}.$$

Proof. We omit the superscript (m) for readability. We have

$$\begin{aligned}
\Delta\pi_k &= \left|\frac{\hat{r}_k}{n} - \frac{\tilde{r}_k}{n-1}\right| \\
&= \left|\frac{1}{n}\sum_{i\in[n]} \hat{r}_{ik} - \frac{1}{n-1}\sum_{i\in[n]-j} \tilde{r}_{ik}\right| \\
&\leq \frac{\hat{r}_{jk}}{n} + \left|\sum_{i\in[n]-j}\left(\frac{\hat{r}_{ik}}{n} - \frac{\tilde{r}_{ik}}{n-1}\right)\right| \\
&\leq \frac{\hat{r}_{jk}}{n} + \frac{\sum_{i\in[n]-j}\tilde{r}_{ik}}{n(n-1)} + \frac{1}{n}\left|\sum_{i\in[n]-j} \hat{r}_{ik} - \tilde{r}_{ik}\right| \\
&\leq \frac{\hat{r}_{jk}}{n} + \frac{\tilde{\pi}_k}{n} + \frac{1}{n}\sum_{i\in[n]-j} \Delta r_{ik}.
\end{aligned}$$

□

5.4.3 . Recursive Bound on Responsibilities

For $i \in [n]-j$, $k \in [K]$, and a positive integer m, let

$$\hat{\eta}_{ik}^{(m)} := \frac{1}{2\sigma^2}\|x_i - \hat{\mu_k}^{(m-1)}\|^2,$$

and

$$\tilde{\eta}_{ik}^{(m)} := \frac{1}{2\sigma^2}\|x_i - \tilde{\mu}_k^{(m)}\|^2.$$

Also, let

$$\hat{\phi}_{ik}^{(m)} := \hat{\pi}_k^{(m)} \exp\Big(-\hat{\eta}_{ik}^{(m)}\Big),$$

and

$$\tilde{\phi}_{ik}^{(m)} := \tilde{\pi}_k^{(m)} \exp\Big(-\tilde{\eta}_{ik}^{(m)}\Big),$$

The goal of this section is to bound $\Delta r_{ik}^{(m)}$ using $D_k^{(m-1)}$ and $\Delta\pi_k^{(m-1)}$. We start with the following lemma.

Lemma 8. *For any $i \in [n]-j$, $k \in [K]$, and a positive integer m, we have*

$$\left|\hat{\phi}_{ik}^{(m)} - \tilde{\phi}_{ik}^{(m)}\right| \leq \left(1 + \frac{2}{\sigma^2}\right)\Delta\pi_k^{(m)} + \frac{2}{\sigma^2}D_k^{(m)}.$$

Proof. We omit the superscript (m) for readability. First, we note that

$$-\frac{2}{\sigma^2}\left\|\hat{\mu}_k-\tilde{\mu}_k\right\| \leq \hat{\eta}_{ik}-\tilde{\eta}_{ik} \leq \frac{2}{\sigma^2}\left\|\hat{\mu}_k-\tilde{\mu}_k\right\|$$

Hence

$$|\hat{\eta}_{ik}-\tilde{\eta}_{ik}| \leq \frac{2}{\sigma^2}\left\|\hat{\mu}_k-\tilde{\mu}_k\right\| \tag{5.11}$$

Moreover, we have

$$\begin{aligned}
\left|\hat{\phi}_{ik}-\tilde{\phi}_{ik}\right| &= |\hat{\pi}_k \exp(-\hat{\eta}_{ik}) - \tilde{\pi}_k \exp(-\hat{\eta}_{ik}) + \tilde{\pi}_k \exp(-\hat{\eta}_{ik}) - \tilde{\pi}_k \exp(-\tilde{\eta}_{ik})| \\
&\leq \Delta\pi_k + \tilde{\pi}_k |\exp(-\hat{\eta}_{ik}) - \exp(-\tilde{\eta}_{ik})| \\
&\leq \Delta\pi_k + \tilde{\pi}_k |\hat{\eta}_{ik}-\tilde{\eta}_{ik}| \\
&\qquad\qquad \text{(by Lipschitz continuity of } \exp(-x) \text{ for } x \geq 0) \\
&\leq \Delta\pi_k + \frac{2\tilde{\pi}_k}{\sigma^2}\|\hat{\mu}_k-\tilde{\mu}_k\| \qquad \text{(using 5.11 and } \|x_i\|, \|\hat{\boldsymbol{\mu}}_k\|, \|\tilde{\mu}\| \leq 1) \\
&= \Delta\pi_k + \frac{2}{\sigma^2}\|\tilde{\pi}_k\hat{\mu}_k - \tilde{\pi}_k\tilde{\mu}_k\| \\
&= \Delta\pi_k + \frac{2}{\sigma^2}\|\tilde{\pi}_k\hat{\mu}_k - \hat{\pi}_k\hat{\mu}_k + \hat{\pi}_k\hat{\boldsymbol{\mu}}_k - \tilde{\pi}_k\tilde{\mu}_k\| \\
&\leq \Delta\pi_k + \frac{2}{\sigma^2}\Delta\pi_k \cdot \|\hat{\mu}_k\| + \frac{2}{\sigma^2}\|\hat{\pi}_k\hat{\mu}_k - \tilde{\pi}_k\tilde{\mu}_k\| \\
&\leq \Delta\pi_k + \frac{2}{\sigma^2}\Delta\pi_k + \frac{2}{\sigma^2}\|\hat{\pi}_k\hat{\mu}_k - \tilde{\pi}_k\tilde{\mu}_k\| \qquad \text{(by } \|\mu_k\| \leq 1) \\
&\leq \left(1+\frac{2}{\sigma^2}\right)\Delta\pi_k + \frac{2}{\sigma^2}D_k. \qquad \square
\end{aligned}$$

Lemma 9. *For any $i \in [n]-j$ and a positive integer m, we have*

$$\sum_{k\in[K]} \Delta r_{ik}^{(m)} \leq \alpha \sum_{k\in[K]} \Delta\pi_k^{(m-1)} + \beta \sum_{k\in[K]} D_k^{(m-1)},$$

where

$$\begin{aligned}
\alpha &= \left(\exp\left(\frac{2}{\sigma^2}\right) + \exp\left(\frac{4}{\sigma^2}\right)\right) \cdot \left(1+\frac{2}{\sigma^2}\right), \\
\beta &= \left(\exp\left(\frac{2}{\sigma^2}\right) + \exp\left(\frac{4}{\sigma^2}\right)\right) \cdot \frac{2}{\sigma^2}.
\end{aligned}$$

Proof. In this proof, we omit the superscript (m) from $\hat{r}_{ik}^{(m)}$ and $\tilde{r}_{ik}^{(m)}$ and the superscript $(m-1)$ from other notations for readability.

Let $\hat{S} := \sum_{\ell\in[K]} \hat{\phi}_{i\ell}$ and $\tilde{S} := \sum_{\ell\in[K]} \tilde{\phi}_{i\ell}$. Because $\|x_i - \hat{\mu}_\ell\| \leq 2$ and $\|x_i - \tilde{\mu}_\ell\| \leq 2$ for every $i \in [n]$ and $\ell \in [K]$, we have $\hat{\pi}_\ell \exp\left(-2/\sigma^2\right) \leq \hat{\phi}_{i\ell} \leq \hat{\pi}_\ell$ and $\tilde{\pi}_\ell \exp\left(-2/\sigma^2\right) \leq \tilde{\phi}_{i\ell} \leq \tilde{\pi}_\ell$ for every $i \in [n]$ and $\ell \in [K]$. Then since $\sum_{\ell\in[K]} \hat{\pi}_\ell = \sum_{\ell\in[K]} \tilde{\pi}_\ell = 1$, we have

$$\exp\left(-\frac{2}{\sigma^2}\right) \leq \hat{S}, \tilde{S} \leq 1. \tag{5.12}$$

Next, we have

$$\begin{aligned}\Delta r_{ik} = \left|\frac{\hat{\phi}_{ik}}{\hat{S}} - \frac{\tilde{\phi}_{ik}}{\tilde{S}}\right| &\leq \left|\frac{\hat{\phi}_{ik} - \tilde{\phi}_{ik}}{\hat{S}}\right| + \tilde{\phi}_{ik}\left|\frac{1}{\hat{S}} - \frac{1}{\tilde{S}}\right| \\ &= \frac{|\hat{\phi}_{ik} - \tilde{\phi}_{ik}|}{\hat{S}} + \frac{\tilde{\phi}_{ik}}{\hat{S}\tilde{S}}\left|\hat{S} - \tilde{S}\right|.\end{aligned}$$

By Lemma 8, we have

$$\begin{aligned}|\hat{\phi}_{ik} - \tilde{\phi}_{ik}| &\leq \left(1 + \frac{2}{\sigma^2}\right)\Delta\pi_k + \frac{2}{\sigma^2}D_k, \\ |\hat{S} - \tilde{S}| &\leq \left(1 + \frac{2}{\sigma^2}\right)\sum_{\ell\in[K]}\Delta\pi_\ell + \frac{2}{\sigma^2}\sum_{\ell\in[K]}D_\ell.\end{aligned}$$

Combining all, we obtain

$$\begin{aligned}\sum_{k\in[K]}\Delta r_{ik} &\leq \sum_{k\in[K]}\frac{1}{\hat{S}}\left(\left(1 + \frac{2}{\sigma^2}\right)\Delta\pi_k + \frac{2}{\sigma^2}D_k\right) + \\ &\qquad \sum_{k\in[K]}\frac{\tilde{\phi}_{ik}}{\hat{S}\tilde{S}}\left(\left(1 + \frac{2}{\sigma^2}\right)\sum_{\ell\in[K]}\Delta\pi_\ell + \frac{2}{\sigma^2}\sum_{\ell\in[K]}D_\ell\right) \\ &\leq \exp\left(\frac{2}{\sigma^2}\right)\sum_{k\in[K]}\left(\left(1 + \frac{2}{\sigma^2}\right)\Delta\pi_k + \frac{2}{\sigma^2}D_k\right) + \\ &\qquad \exp\left(\frac{4}{\sigma^2}\right)\sum_{k\in[K]}\tilde{\pi}_k\left(\left(1 + \frac{2}{\sigma^2}\right)\sum_{\ell\in[K]}\Delta\pi_\ell + \frac{2}{\sigma^2}\sum_{\ell\in[K]}D_\ell\right). \\ &\qquad\qquad \text{(by (5.12) and } \tilde{\phi}_{ik} \leq \tilde{\pi}_k\text{)}\end{aligned}$$

Noting that $\sum_{k\in[K]}\tilde{\pi}_k = 1$, we have

$$\sum_{k\in[K]}\Delta r_{ik} \leq \alpha\sum_{k\in[K]}\Delta\pi_k + \beta\sum_{k\in[K]}D_k. \qquad \square$$

5.4.4 . Putting It Together

Proof of Theorem 5.4.1. By $\sum_{k\in[K]}\tilde{\pi}_k^{(m)} = 1$ and Lemmas 6 and 9, we have

$$\sum_{k\in[K]}D_k^{(m)} \leq \frac{\sum_{k\in[K]}\hat{r}_{jk}^{(m)}}{n} + \frac{1}{n} + \alpha\sum_{k\in[K]}\Delta\pi_k^{(m-1)} + \beta\sum_{k\in[K]}D_k^{(m-1)}.$$

By $\sum_{k\in[K]}\tilde{\pi}_k^{(m)} = 1$ and Lemmas 7 and 9, we have

$$\sum_{k\in[K]}\Delta\pi_k^{(m)} \leq \frac{\sum_{k\in[K]}\hat{r}_{jk}^{(m)}}{n} + \frac{1}{n} + \alpha\sum_{k\in[K]}\Delta\pi_k^{(m-1)} + \beta\sum_{k\in[K]}D_k^{(m-1)}.$$

Hence, we have

$$\max\left\{\sum_{k\in[K]}\Delta\pi_k^{(m)},\sum_{k\in[K]}D_k^{(m)}\right\}\leq\frac{\sum_{k\in[K]}\hat{r}_{jk}^{(m)}}{n}+\frac{1}{n}$$
$$+(\alpha+\beta)\max\left\{\sum_{k\in[K]}\Delta\pi_k^{(m-1)},\sum_{k\in[K]}D_k^{(m-1)}\right\}.$$

This inequality has the following form:

$$a_m\leq(\alpha+\beta)a_{m-1}+b_m,$$

where

$$a_m=\max\left\{\sum_{k\in[K]}\Delta\pi_k^{(m)},\sum_{k\in[K]}D_k^{(m)}\right\},$$
$$b_m=\frac{1}{n}\left(\sum_{k\in[K]}\hat{r}_{jk}^{(m)}+1\right).$$

Hence by induction, we have:

$$a_m\leq(\alpha+\beta)^m a_0+\sum_{t=1}^{m}(\alpha+\beta)^{m-t}b_t.$$

Recall that a_m and b_t are functions of the deleted point x_j, and thus we denote them by $a_m(j)$ and $b_t(j)$. From the assumption that we have the same initialization, we have

$$a_0(j)=\max\left\{\Delta\pi_k^{(0)},D_k^{(0)}\right\}=0.$$

On the other hand, we have

$$\frac{1}{n}\sum_{j\in[n]}b_t(j)=\frac{1}{n}\sum_{j\in[n]}\frac{1}{n}\left(\sum_{k\in[K]}\hat{r}_{jk}^{(t)}+1\right)\leq\frac{2}{n}.$$

Also $D_k^{(m)}$ is a function of the deleted point x_j, and we denote it by $D_k^{(m)}(j)$. Combining the inequalities above, the average sensitivity is

$$\begin{aligned}\frac{1}{n}\sum_{j\in[m]}D_k^{(m)}(j)&\leq\frac{1}{n}\sum_{j\in[m]}a_m(j)\\&\leq\frac{1}{n}\sum_{j\in[m]}(\alpha+\beta)^m a_0(j)+\frac{1}{n}\sum_{j\in[m]}\sum_{t=1}^{m}(\alpha+\beta)^{m-t}b_t(j)\\&\leq\frac{2}{n}\sum_{t=1}^{m}(\alpha+\beta)^{m-t}=\frac{2}{n}\cdot\frac{(\alpha+\beta)^m-1}{(\alpha+\beta)-1}.\end{aligned}$$

□

The reason we do not have a dependency on K in Theorem 5.4.1 is that, in the calculation of D_k, the terms $1/\hat{r}_k$ and $1/\tilde{r}_k$ are canceled out by $\hat{\pi}_k$ and $\tilde{\pi}_k$, respectively, in Lemma 6. In Lemma 5, when approximating $\tilde{r}_{ik}$ with $\hat{r}_{ik}$, we require error bounds for $\tilde{\eta}_{il}$ for $\ell \neq k$, which introduces an extra factor of K. An interesting open question is to eliminate the dependence on K in Theorem 5.3.1.

5.5 . Experiments

This section reports our experimental results, which validate our theoretical findings. After describing the datasets used in our experiments in Section 5.5.1. We analyze the behavior of average sensitivity as a function of the sample size n in Section 5.5.2. Then, we evaluate both average sensitivity and clustering quality with respect to the variance parameter σ^2 in Section 5.5.3. Finally, we analyze the effect of the number of clusters K on the average sensitivity in Section 5.5.4.

It is important to emphasize that the goal of the following experiments is not to show that our theoretical bounds are tight, but rather to illustrate the overall trend in average sensitivity.

5.5.1 . Datasets

In our experiments, we use both synthetic and real datasets, described below.

Synthetic: We generated two-dimensional data points from four Gaussian components ($K = 4$) with variance $\sigma^2 = 1.0$, means $\mu_1 = (-1.0, 0.0)$, $\mu_2 = (0.0, -1.0)$, $\mu_3 = (1.0, 0.0)$, and $\mu_4 = (0.0, 1.0)$, and mixture proportions $\boldsymbol{\pi} = (0.01, 0.01, 0.49, 0.49)$. We rescale the data to lie within the range $[-1, 1]^2$. For all the experiments, we generated $n = 10,000$ points except for the first experiment, where n varies.

MNIST [44]: This dataset consists of $n = 70,000$ images of handwritten digits, grouped into 10 clusters. For dimensionality reduction, we apply Principal Component Analysis (PCA), keeping the first $d = 25$ components, which preserve 70% of the explained variance. The data are rescaled to lie within $[-1, 1]^d$.

For all the following experiments, unless otherwise specified, we partition each dataset into chunks of 200 points, compute the average sensitivity after deleting each chunk (normalized by the chunk size), and then take the mean of these sensitivities.

5.5.2 . Effect of Sample Size

First, we analyze the average sensitivity of the EM algorithm as a function of n, where $n \in \{2^{11}, \ldots, 2^{17}\}$. The EM algorithm is set with $K = 2$, $\hat{\mu}_1^{(0)} =$

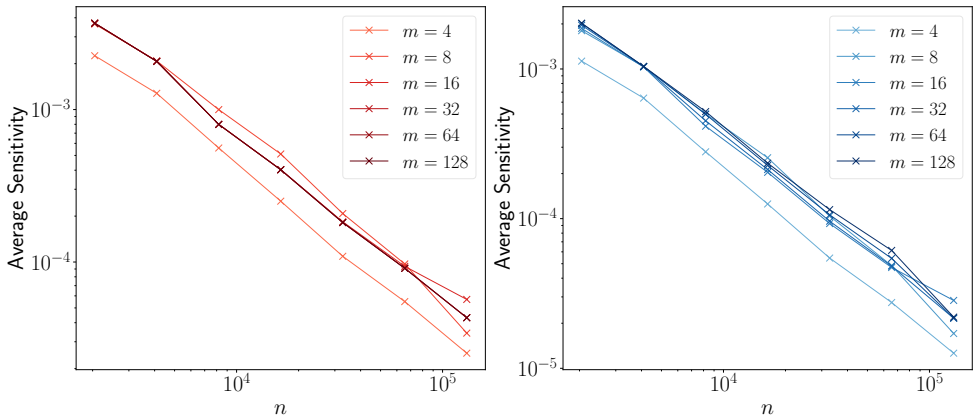

Figure 5.2: Average sensitivity v.s. n for the uniform mixture case (left) and the unknown mixture case (right) on the synthetic dataset.

$\tilde{\mu}_1^{(0)} = (-0.01, 0.0)$, $\hat{\mu}_2^{(0)} = \tilde{\mu}_2^{(0)} = (0.01, 0.0)$ with the variance parameter $\sigma^2 = 0.05$. For the unknown mixture case, we have set $\hat{\pi}^{(0)} = \tilde{\pi}^{(0)} = (0.5, 0.5)$. In this case, the number of chunks is 128 points (so that it divides 2^i).

Figure 5.2 presents the empirical average sensitivity of the EM algorithm with varying number of iterations m for the uniform and unknown mixture cases, respectively. Our theoretical bounds (Theorems 5.3.1 and 5.4.1) suggest that the average sensitivity is proportional to $1/n$, a trend that is corroborated by the empirical results. We note that this behavior cannot be attributed solely to the convergence of the empirical mean to the true mean, since the chosen value of K for the EM algorithm is different from the actual number of clusters and the convergence rate of the empirical mean is $O(1/\sqrt{n})$ according to the standard concentration bound.

5.5.3 . Effect of Variance Parameter

Mnist dataset

Next, we analyze the average sensitivity and clustering quality—measured using the Adjusted Rand Index (ARI), of the EM algorithm as a function of the variance parameter σ^2 on the MNIST dataset. The EM algorithm is configured with $K = 10$ clusters. For initialization, we randomly select one observation from each cluster and rescale the selected means by a factor of 5.0 to prevent trivial clustering and fast convergence. For the unknown mixture case, we set $\hat{\pi}_k^{(0)} = \tilde{\pi}_k^{(0)} = 0.1$ for all $k \in [K]$.

The results for the uniform mixture case (Figure 5.3) strongly support our theoretical bound (Theorem 5.3.1). Specifically, when the variance parameter σ^2 of the EM algorithm is high, the average sensitivity remains low regardless of the number of iterations m. In contrast, for low values of σ^2, the average sensitivity increases with m. However, when σ^2 is too large, clustering quality

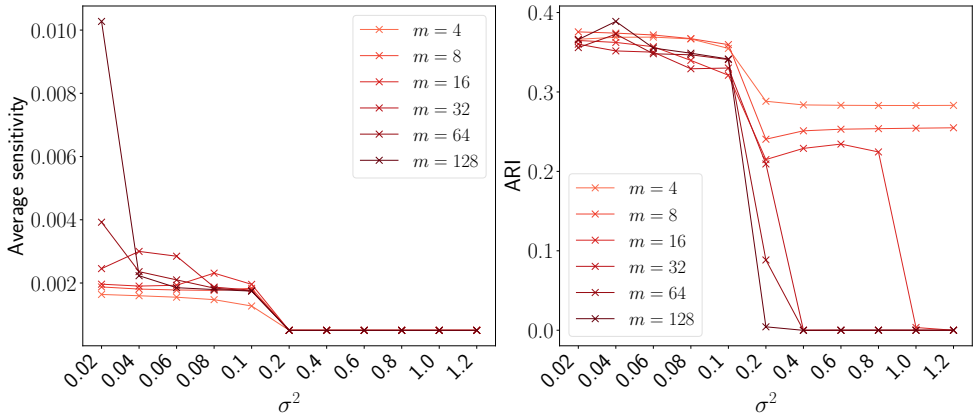

Figure 5.3: Average sensitivity (right) and clustering quality measured by ARI (left) as functions of σ^2 for the uniform mixture case on the MNIST dataset.

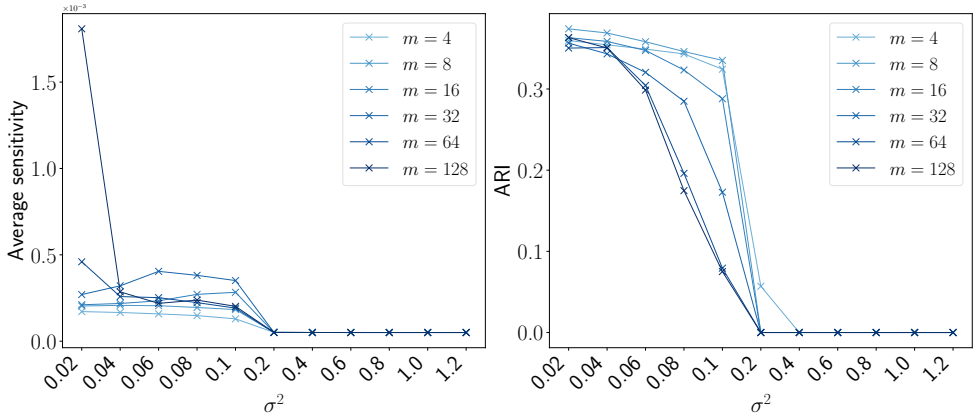

Figure 5.4: Average sensitivity (right) and clustering quality measured by ARI (left) as functions of σ^2 for the unknown mixture case on the MNIST dataset.

may degrade. As shown in Figure 5.3, there exists a range (e.g., $\sigma^2 \in [0.04, 0.1]$) where the average sensitivity is low and the ARI score remains high. This highlights a trade-off between stability and clustering quality when selecting the variance parameter. We also observe in the same figure that the ARI score is higher for small values of m, which is due to the slower convergence of the estimated means (centroids) when σ^2 is small. Similar results were also obtained on the synthetic dataset.

Synthetic dataset

We analyze the average sensitivity of the EM algorithm as a function of σ^2 on the synthetic dataset (Section 5.5.1). The EM algorithm is set with $K = 3$, $\hat{\boldsymbol{\mu}}_1^{(0)} = \tilde{\boldsymbol{\mu}}_1^{(0)} = (-0.1, 0.0)$, $\hat{\boldsymbol{\mu}}_2^{(0)} = \tilde{\boldsymbol{\mu}}_2^{(0)} = (0.1, 0.0)$, $\hat{\boldsymbol{\mu}}_3^{(0)} = \tilde{\boldsymbol{\mu}}_3^{(0)} = (0.0, 0.1)$, For

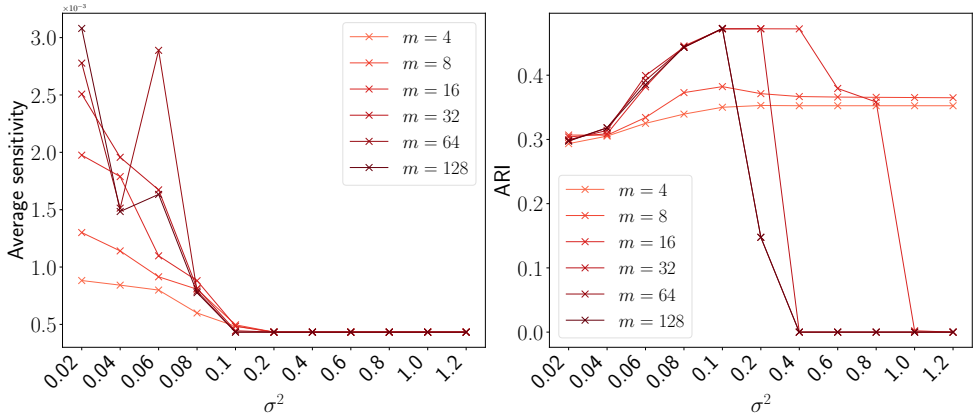

Figure 5.5: Average sensitivity (right) and clustering quality measured by ARI (left) as functions of σ^2 for the uniform mixture case on the synthetic dataset.

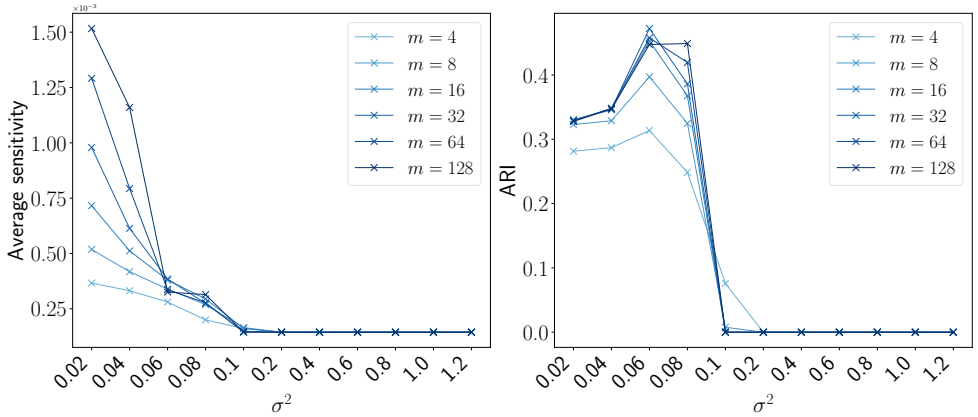

Figure 5.6: Average sensitivity (right) and clustering quality measured by ARI (left) as functions of σ^2 for the unknown mixture case on the synthetic dataset.

the unknown mixture case, we have set $\hat{\pi}^{(0)} = \tilde{\pi}^{(0)} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. See Figures 5.5 and 5.6 for the results. As observed, the trend is similar to that obtained for the MNIST dataset (Figures 5.3 and 5.4).

5.5.4 . Effect of the Number of Clusters

Mnist dataset

Next, we analyze the average sensitivity of the EM algorithm as a function of the number of clusters K on the MNIST dataset. The EM algorithm is set with $\sigma^2 = 0.01$ and for each $k \in [K]$, $\hat{\boldsymbol{\mu}}_k^{(0)} = \tilde{\boldsymbol{\mu}}_k^{(0)}$ are set by randomly sampling K points from the data and rescaling them by a factor of 5.0. For the unknown mixture case, we have set $\hat{\pi}_k^{(0)} = \tilde{\pi}_k^{(0)} = 1/K$ for every $k \in [K]$. Since the dimension d is large and the number of clusters K increases, computing

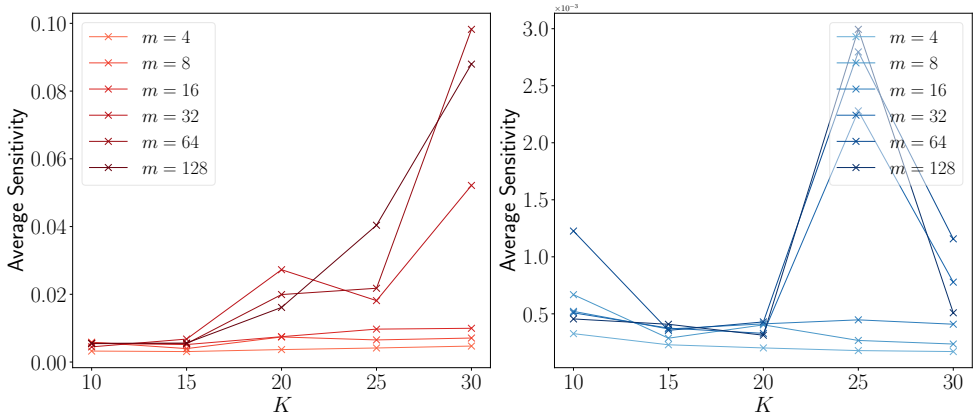

Figure 5.7: Average sensitivity v.s. K for the uniform mixture case (left) and for the unknown mixture case (right) on the MNIST dataset.

the average sensitivity becomes prohibitively expensive when the number of chunks is small. For computational efficiency, we fix the number of chunks to 500.

The results are reported in Figure 5.7. In the uniform mixture setting, the average sensitivity increases with K. In contrast, for the unknown mixture case, no clear increasing trend is observed as K increases. This is consistent with the theoretical bound in Theorem 5.4.1, which is independent of K. A similar trend is observed on the synthetic dataset. In particular, Figure 5.8 clearly shows that the sensitivity increases linearly with respect to K, strongly supporting our theoretical result in Theorem 5.3.1, which depends linearly on K.

Synthetic dataset

Next, we analyze the average sensitivity of the EM algorithm as a function of the number of clusters K on the synthetic dataset (Section 5.5.1). The EM algorithm is set with $\sigma^2 = 0.05$ and for each $k \in [K]$, $\hat{\boldsymbol{\mu}}_k^{(0)} = \tilde{\boldsymbol{\mu}}_k^{(0)}$ are set by randomly sampling K from the data and rescaling them by a factor of 5.0. For the unknown mixture case, we have set $\hat{\pi}_k^{(0)} = \tilde{\pi}_k^{(0)} = 1/K$ for every $k \in [K]$.

The results are presented in Figure 5.8. For the uniform mixture case, we observe a similar trend to that seen on the MNIST dataset (Figure 5.7), with the average sensitivity increasing linearly with K. This observation aligns closely with our theoretical result in Theorem 5.3.1. When the mixture proportions are unknown, the average sensitivity shows a slight decrease as K increases. However, the magnitude of this change is minimal. This corroborates our theoretical finding (Theorem 5.4.1).

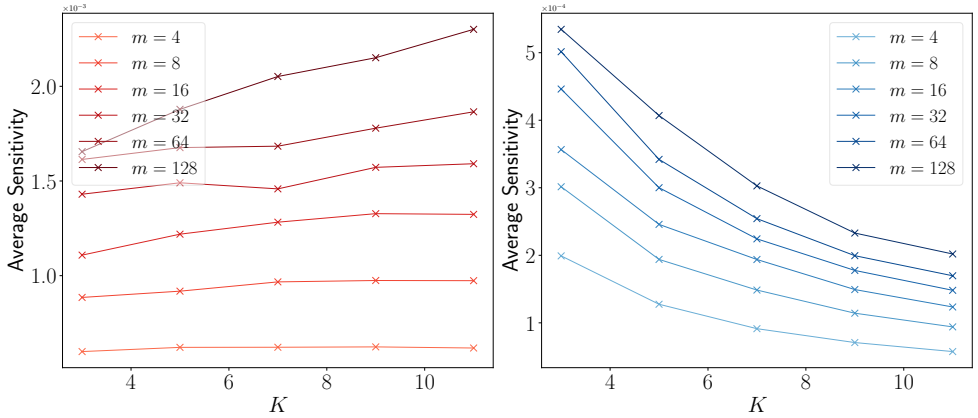

Figure 5.8: Average sensitivity v.s. K for the uniform mixture case (left) and for the unknown mixture case (right) on the synthetic dataset.

5.6 . Conclusion

In this chapter, we investigated the average sensitivity of the EM algorithm as applied to Gaussian mixture models. In the uniform mixture proportion scenario, we theoretically bounded the average sensitivity by $O(K/n)$ when the variance parameter σ^2 is sufficiently large. This bound is independent of the number of iterations m. In the unknown mixture proportions scenario, we showed that the average sensitivity is $O(c^m/n)$, where $c > 2$ is a decreasing function of σ^2. When m is small, which is often the case in practice, this bound is smaller than the trivial upper bound of $O(1)$. We further validated these theoretical findings with experiments. Although we observe that the EM algorithm becomes more stable as σ^2 increases, this improvement in stability may come at the expense of clustering quality, as illustrated in Section 5.5.3. Thus, these results naturally raise two important directions for future research. The first is to explore the trade-off between clustering quality and average sensitivity. The second is to investigate average sensitivity in scenarios where the covariance matrices are also unknown.

6 - Conclusion and perspectives

In this manuscript, we first proposed several distributed algorithms that approximate the collapsed Gibbs sampler in the context of clustering with Dirichlet Process Mixture Models and co-clustering using Non-Parametric Latent Block Models. We summarize our contributions as follows:

We first introduced DisCGS, a novel distributed inference algorithm for DPMMs based on sufficient statistics and a Master/Worker architecture. This method enables scalable and efficient clustering by avoiding the need to share raw data, making it well-suited for distributed environments and federated learning settings. Through extensive empirical evaluation, we demonstrated that DisCGS significantly reduces inference time while maintaining high clustering accuracy.

We then extended this framework to support non-Gaussian component distributions, particularly focusing on Multinomial component distributions in the context of text clustering. More generally, we developed a generic version of the distributed inference algorithm applicable to any component distribution that belongs to the exponential family, thereby broadening the applicability of our method to a wide range of data types.

Building on this foundation, we developed a distributed inference method for the NPLBM, which simultaneously estimates row and column partitions in large-scale data matrices. In this framework, local row partitions are inferred in parallel across workers, while the global co-clustering structure is reconstructed at the master using only sufficient statistics. This method allows scalable co-clustering without direct access to the full dataset.

In the last contribution, we conducted a theoretical analysis of the average sensitivity of the Expectation-Maximization algorithm for spherical Gaussian Mixture Models. We provided bounds on the average sensitivity, offering insights into its behavior. These results contribute to a better understanding of the robustness of EM-based inference in clustering tasks. We have also validated our theoretical results with experiments on synthetic and real data.

Finally, we suggest several research directions that naturally extend the contributions of this manuscript:

6.1 . Federated Dirichlet Process Mixture Models

The methodology introduced in Chapter 2 is well-suited for horizontal federated learning. A natural extension is to deploy our distributed inference algorithm in a federated setting, where data are partitioned across different clients (workers) without centralized access. Each client initializes with

a shared global model and runs collapsed Gibbs sampling locally on its private data to infer local cluster structures (tables). The sufficient statistics and associated counts from each local model are then transmitted to the central server (master), which aggregates them to update the global model and estimate the global clustering structure. These updates are broadcast back to the clients to synchronize their local models. This federated inference process alternates between local and global updates until convergence.

6.2 . Adaptive Prior on Concentration Parameters

The concentration parameter α influences the granularity of the inferred clustering: higher values of α generally lead to more inferred clusters. In our current work, we adopted fixed, empirically chosen values for α based on standard practices. However, this parameter could instead be inferred from the data. A promising extension is to place a Gamma prior on α and update it adaptively during inference at both the worker and master levels. This would enable the model to adjust the concentration parameter dynamically from the observed data, and is particularly relevant in federated environments with heterogeneous data.

6.3 . Distributed Multiple Co-Clustering

The methods developed in Chapters 2 and 4 may be extended to implement a distributed version of the Multiple Co-Clustering model proposed in [61]. This model combines two layers of Bayesian non-parametric structure: (1) a multiple clustering layer that groups variables sharing the same row partition, and (2) a co-clustering layer that assigns a separate NPLBM to each variable group. A distributed inference algorithm for this model would allow scalable co-clustering of very large and high-dimensional datasets.

6.4 . Inference with Non-Conjugate Priors via Metropolis-Hastings

The current distributed algorithm assumes the conjugacy of the prior to enable the computations in the collapsed Gibbs sampling. However, it is possible to extend this method to non-conjugate settings. In this case, the collapsed Gibbs sampler becomes inapplicable. An alternative is to incorporate general sampling methods like Metropolis-Hastings (MH) to perform inference. This would extend the flexibility of the algorithm to a broader class of models, including those with complex or non-standard priors, without sacrificing scalability.

6.5 . Average Sensitivity of EM with Unknown Full Covariance Matrices

In Chapter 5, we focused on a simplified scenario where clusters are assumed to be spherical with known diagonal covariance matrices (Assumption 1). While this assumption makes the analysis tractable, it limits the generality of the results. A more challenging and impactful extension involves analyzing the average sensitivity of the EM algorithm when each component has a full, unknown covariance matrix. Such an analysis would offer a more realistic understanding of EM behavior in practical GMM applications.

6.6 . Trade-Off Between Average Sensitivity and Clustering Quality

Our empirical results in Chapter 5 reveal that increasing the variance parameter σ^2 leads to greater stability of the EM algorithm (i.e., lower sensitivity), but this comes at the cost of reduced clustering accuracy, as shown in Section 5.5.3. This observation suggests a fundamental trade-off between stability and clustering quality. A deeper theoretical investigation into this trade-off could shed light on how to optimally balance robustness and performance in practical clustering scenarios.

Appendix

6.7 . Proofs

Before computing the expressions for the multivariate Gaussian case, we start by computing those of the exponential family case.

6.7.1 . Exponential family case

Prior predictive

$$\begin{aligned}
\mathrm{p}(x_i^j|\Lambda_0,\tau_0) &= \int_\Theta \mathrm{p}(x_i^j|\theta)\mathrm{p}(\theta|\Lambda_0,\tau_0)d\theta, && (6.1)\\
&= \int_\Theta f(x_i^j,\theta)g(\theta,\Lambda_0,\tau_0)d\theta, && (6.2)\\
&= \int_\Theta \exp\Big(\eta(\theta)^T\Lambda_1 - \phi(x_i^j) - \tau_1\psi(\theta) - \xi(\Lambda_0,\tau_0)\Big)\,d\theta && (6.3)\\
&= \exp\Big(-\phi(x_i^j) - \xi(\Lambda_0,\tau_0)\Big)\int_\Theta \exp\big(\eta(\theta)^T\Lambda_1 - \tau_1\psi(\theta)\big)\,d\theta && (6.4)
\end{aligned}$$

with $\Lambda_1 = \Lambda_0 + S(x_i^j)$ and $\tau_1 = \tau_0 + 1$ the integral part can be obtained as follows:

$$\begin{aligned}
p(\theta|\Lambda_1,\tau_1) &= g(\theta,\Lambda_1,\tau_1) && (6.5)\\
&= \exp\big(\eta(\theta)^T\Lambda_1 - \tau_1\psi(\theta) - \xi(\Lambda_1,\tau_1)\big) && (6.6)\\
&= \exp\left(-\xi(\Lambda_1,\tau_1)\right)\exp\big(\eta(\theta)^T\Lambda_1 - \tau_1\psi(\theta)\big) && (6.7)
\end{aligned}$$

using the fact that $\int_\Theta g(\theta,\Lambda_1,\tau_1)d\theta = 1$, we obtain:

$$\exp\left(-\xi(\Lambda_1,\tau_1)\right)\int_\Theta \exp\big(\eta(\theta)^T\Lambda_1 - \tau_1\psi(\theta)\big)\,d\theta = 1 \tag{6.8}$$

hence,

$$\int_\Theta \exp\big(\eta(\theta)^T\Lambda_1 - \tau_1\psi(\theta)\big)\,d\theta = \exp\left(\xi(\Lambda_1,\tau_1)\right) \tag{6.9}$$

therefore,

$$\mathrm{p}(x_i^j|\Lambda_0,\tau_0) = \exp\Big(\xi(\Lambda_0 + S(x_i^j),\tau_0+1) - \xi(\Lambda_0,\tau_0) - \phi(x_i^j)\Big). \tag{6.10}$$

□

Posterior predictive

$$\mathrm{p}(x_i^j|z_i^j = k, \mathbf{x}_k^j, \Lambda_0, \tau_0) = \int_\Theta \mathrm{p}(x_i^j|\theta)\mathrm{p}(\theta|\mathbf{x}_k^j, \Lambda_0, \tau_0)d\theta, \tag{6.11}$$

using the conjugacy property, we have:

$$\mathrm{p}(\theta|\mathbf{x}_k^j, \Lambda_0, \tau_0) = g(\theta, \Lambda_k^j, \tau_k^j), \tag{6.12}$$

with $\Lambda_k^j = \Lambda_0 + \sum_{x\in\mathbf{x}_k^j} x$ and $\tau_k^j = \tau_0 + n_k^j$ the updated hyper-parameters. Therefore:

$$\mathrm{p}(x_i^j|z_i^j = k, \mathbf{x}_k^j, \Lambda_0, \tau_0) = \int_\Theta f(x_i^j, \theta)g(\theta, \Lambda_k^j, \tau_k^j)d\theta, \tag{6.13}$$

$$= \int_\Theta \exp\left(\eta(\theta)^T\Lambda_k^j - \phi(x_i^j) - \tau_k^j\psi(\theta) - \xi(\Lambda_0, \tau_0)\right)d\theta \tag{6.14}$$

$$= \exp\left(-\phi(x_i^j) - \xi(\Lambda_0, \tau_0)\right)\int_\Theta \exp\left(\eta(\theta)^T\Lambda_k^j - \tau_k^j\psi(\theta)\right)d\theta \tag{6.15}$$

by computing the integral part similarly to the one in the equation 6.4, we obtain:

$$\int_\Theta \exp\left(\eta(\theta)^T\Lambda_k^j - \tau_k^j\psi(\theta)\right)d\theta = \exp\left(\xi(\Lambda_k^j, \tau_k^j)\right) \tag{6.16}$$

Hence

$$\mathrm{p}(x_i^j|z_i^j = k, \mathbf{x}_k^j, \Lambda_0, \tau_0) = \exp\left(\xi(\Lambda_k^j + S(x_i^j), \tau_k^j + 1) - \xi(\Lambda_k^j, \tau_k^j) - \phi(x_i^j)\right), \tag{6.17}$$

□

Joint prior predictive

We start by computing the expression of the prior predictive distribution:

$$\mathrm{p}(\mathbf{x}_h^j|\Lambda_0, \tau_0) = \int_\Theta \mathrm{p}(\mathbf{x}_h^j|\theta)\mathrm{p}(\theta|\Lambda_0, \tau_0)d\theta, \tag{6.18}$$

$$= \int_\Theta \prod_{x\in\mathbf{x}_h^j} \mathrm{p}(x|\theta)\mathrm{p}(\theta|\Lambda_0, \tau_0)d\theta, \tag{6.19}$$

$$= \int_\Theta \prod_{x\in\mathbf{x}_h^j} f(x, \theta)g(\theta, \Lambda_0, \tau_0)d\theta, \tag{6.20}$$

$$= \int_\Theta \exp\left(\eta(\theta)^T\Lambda_h^j - \sum_{x\in\mathbf{x}_h^j}\phi(x) - \tau_h^j\psi(\theta) - \xi(\Lambda_0, \tau_0)\right)d\theta, \tag{6.21}$$

$$= \exp\left(-\sum_{x\in\mathbf{x}_h^j}\phi(x) - \xi(\Lambda_0,\tau_0)\right)\int_\Theta \exp\left(\eta(\theta)^T\Lambda_h^j - \tau_h^j\psi(\theta)\right)d\theta, \tag{6.22}$$

$$= \exp\left(\xi(\Lambda_h^j,\tau_h^j) - \xi(\Lambda_0,\tau_0) - \sum_{x\in\mathbf{x}_h^j}\phi(x)\right). \tag{6.23}$$

□

Joint posterior predictive

$$\mathrm{p}(\mathbf{x}_h^j \mid z_h^j = k, \mathbf{x}_k, G_0) = \int_\Theta \mathrm{p}(\mathbf{x}_h^j|\theta)\mathrm{p}(\theta|\mathbf{x}_k,\Lambda_0,\tau_0)d\theta, \tag{6.24}$$

$$= \int_\Theta \prod_{x\in\mathbf{x}_h^j}\mathrm{p}(x|\theta)\mathrm{p}(\theta|\mathbf{x}_k,\Lambda_0,\tau_0)d\theta, \tag{6.25}$$

$$= \int_\Theta \prod_{x\in\mathbf{x}_h^j} f(x,\theta)g(\theta,\Lambda_k,\tau_k)d\theta, \tag{6.26}$$

$$= \exp\left(-\sum_{x\in\mathbf{x}_h^j}\phi(x) - \xi(\Lambda_k,\tau_k)\right)\int_\Theta \exp\left(\eta(\theta)^T\Lambda_{\tilde{k}} - \tau_{\tilde{k}}\psi(\theta)\right)d\theta, \tag{6.27}$$

$$= \exp\left(\xi(\Lambda_{\tilde{k}},\tau_{\tilde{k}}) - \xi(\Lambda_k,\tau_k) - \sum_{x\in\mathbf{x}_h^j}\phi(x)\right) \tag{6.28}$$

with $\Lambda_{\tilde{k}} = \Lambda_k + \sum_{x\in\mathbf{x}_h^j} S(x)$ and $\tau_{\tilde{k}} = n_h^j + \tau_k$, where Λ_k and τ_k are the posterior hyper-parameters of global cluster k. They are obtained as follows:

$$\Lambda_k = \sum_{(j,h)|z_h^j=k}\sum_{x\in\mathbf{x}_h^j} S(x),$$

and

$$\tau_k = \tau_0 + \sum_{(j,h)|z_h^j=k} n_h^j.$$

□

6.7.2 . Multivariate Gaussian case

In the multivariate Gaussian case, for the computational details of the prior and posterior distribution, please refer to [61]. In the following section, we provide proof for the joint posterior and joint prior distributions computed

at the master level. We recall that the density of the multivariate Gaussian distribution (i.e., $\theta = (\mu, \Sigma)$) is given by

$$f(x, \theta) = (2\pi)^{-d/2}|\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right). \tag{6.29}$$

The multivariate Gaussian distribution belongs to the exponential family of distributions, and its density can be written as [61]:

$$f(x, \theta) = \exp\left(\eta(\theta)^T S(x) - \phi(x) - \psi(\theta)\right), \tag{6.30}$$

where

$$\eta(\theta) = \begin{pmatrix} \Sigma^{-1}\mu \\ -\frac{1}{2}\text{vec}(\Sigma^{-1}) \end{pmatrix}, S(x) = \begin{pmatrix} x \\ \text{vec}(xx^T) \end{pmatrix}, \phi(x) = \frac{d}{2}\log(2\pi),$$

and $\psi(\theta) = \frac{1}{2}\left(\mu^T\Sigma^{-1}\mu + \log(|\Sigma|)\right)$. Here, $\text{vec}(\cdot)$ denotes the vectorization operator. On the other hand, the density function of the NIW prior can be written as [61]:

$$g(\theta, \Lambda_0, \tau_0) = \exp\left(\eta(\theta)^T\Lambda_0 - \tau_0\psi(\theta) - \xi(\Lambda_0, \tau_0)\right), \tag{6.31}$$

where $\Lambda_0 = \begin{pmatrix} \kappa_0\mu_0 \\ \text{vec}(\kappa_0\mu_0\mu_0^T + \Psi_0) \end{pmatrix}$, $\tau_0 = \kappa_0 + \nu_0 + d + 2$, and

$$\xi(\Lambda_0, \tau_0) = \log\left(\frac{2^{\frac{\nu_0 d}{2}}\pi^{d/2}\Gamma_d\left(\frac{\nu_0}{2}\right)}{\kappa_0^{d/2}|\Psi_0|^{\nu_0/2}}\right).$$

By injecting these terms into equation 6.23, the joint prior distribution is obtained as follows:

$$\text{p}(\mathbf{x}_h^j|\Lambda_0, \tau_0) = \exp\left(\xi(\Lambda_j^h, \tau_h^j) - \xi(\Lambda_0, \tau_0) - \sum_{x\in\mathbf{x}_h^j}\phi(x)\right) \tag{6.32}$$

$$= (2\pi)^{-\frac{n_h^j d}{2}} \cdot \frac{2^{\frac{\nu_h^j d}{2}}\pi^{d/2}\Gamma_d\left(\frac{\nu_h^j}{2}\right)}{(\kappa_h^j)^{d/2}|\Psi_h^j|^{\nu_h^j/2}} \cdot \frac{\kappa_0^{d/2}|\Psi_0|^{\nu_0/2}}{2^{\frac{\nu_0 d}{2}}\pi^{d/2}\Gamma_d\left(\frac{\nu_0}{2}\right)} \tag{6.33}$$

$$= \pi^{-\frac{n_h^j d}{2}} \cdot \frac{\kappa_0^{d/2}}{(\kappa_h^j)^{d/2}} \cdot \frac{\Gamma_d\left(\frac{\nu_h^j}{2}\right)}{\Gamma_d\left(\frac{\nu_0}{2}\right)} \cdot \frac{|\Psi_0|^{\nu_0/2}}{|\Psi_h^j|^{\nu_h^j/2}} \tag{6.34}$$

□

The joint posterior distribution can be obtained similarly by injecting the updated hyperparameters in equation 6.28

6.7.3 . Multinomial case

Consider a document $x_i \in \mathbf{x}^j$. For better readability, we omit the indices i and j, as they are clear from the context. Thus, we define $\boldsymbol{\ell}_x = (\ell_x^w)_{w \in V}$, where $\ell_x^w = 0$ for all $w \notin x$. We note that

$$\begin{aligned}
\mathrm{p}(x \mid \theta) &= \prod_{w \in x} \mathsf{Mult}(w \mid \theta), && (6.35)\\
&= \prod_{w \in x} \theta_w^{\ell_{x,w}}, && (6.36)\\
&= \exp\left(\sum_{w \in x} \ell_{x,w} \log\left(\theta_w\right) \right). && (6.37)
\end{aligned}$$

On the other hand, we let $\Delta(\boldsymbol{\gamma}) = \frac{\prod_w \Gamma(\gamma_w)}{\Gamma(\sum_w \gamma_w)}$. We have

$$\begin{aligned}
\mathrm{p}(\theta | G_0) &= \mathsf{Dir}(\theta \mid \boldsymbol{\gamma}), && (6.38)\\
&= \frac{1}{\Delta(\boldsymbol{\gamma})} \prod_{w \in V} \theta_w^{\gamma_w - 1}, && (6.39)\\
&= \exp\left(\sum_{w \in V} (\gamma_w - 1) \log\left(\theta_w\right) - \log(\Delta(\gamma)) \right). && (6.40)
\end{aligned}$$

Hence, the prior predictive distribution Equation (3.3) is obtained as follows

$$\begin{aligned}
\mathrm{p}(x \mid G_0) &= \int_\Theta \mathrm{p}(x|\theta)\mathrm{p}(\theta|G_0) d\theta, && (6.41)\\
&= \int_\Theta \prod_{w \in x} \mathsf{Mult}(w \mid \theta) \mathsf{Dir}(\theta \mid \boldsymbol{\gamma}) d\theta, && (6.42)\\
&= \int_\Theta \exp\left(\sum_{w \in V} (\ell_{x,w} + \gamma_w - 1) \log\left(\theta_w\right) - \log(\Delta(\boldsymbol{\gamma})) \right) d\theta, && (6.43)\\
&= \frac{1}{\Delta(\boldsymbol{\gamma})} \int_\Theta \prod_{w \in V} \theta_w^{\ell_{x,w} + \gamma_w - 1} d\theta, && (6.44)\\
&= \frac{\Delta(\boldsymbol{\ell}_x + \boldsymbol{\gamma})}{\Delta(\boldsymbol{\gamma})}, && (6.45)\\
&= \frac{\Gamma(\sum_{w=1}^V \gamma_w)}{\Gamma(\sum_{w=1}^V \ell_{x,w} + \gamma_w)} \prod_{w=1}^V \frac{\Gamma(\ell_{x,w} + \gamma_w)}{\Gamma(\gamma_w)}, && (6.46)\\
&= \frac{\Gamma(\sum_{w=1}^V \gamma_w)}{\Gamma(\sum_{w=1}^V \ell_{x,w} + \gamma_w)} \prod_{w \in x} \frac{\Gamma(\ell_{x,w} + \gamma_w)}{\Gamma(\gamma_w)}, && (6.47)\\
&= \frac{\prod_{w \in x} \prod_{s=1}^{\ell_{x,w}} (\gamma_w + s - 1)}{\prod_{s=1}^{\ell_x} (\sum_{w=1}^V \gamma_w + s - 1)}, && (6.48)\\
&= \frac{\prod_{w \in x} \prod_{s=1}^{\ell_{x,w}} (\gamma + s - 1)}{\prod_{s=1}^{\ell_x} (V\gamma + s - 1)}, && (6.49)
\end{aligned}$$

where in 6.43 we have use the fact that $\ell_{x,w} = 0, \forall w \notin x$, to obtain 6.45, we have used the fact that

$$\frac{1}{\Delta(\boldsymbol{\ell}_x + \boldsymbol{\gamma})} \int_\Theta \prod_{w \in V} \theta_w^{\ell_{x,w} + \gamma_w - 1} d\theta = 1.$$

Finally, to obtain 6.49, we have used the fact that that $\gamma_1 = \cdots = \gamma_V = \gamma$, and for all $S \in \mathbb{N}$ and $r > 0$, we have

$$\frac{\Gamma(r+S)}{\Gamma(r)} = \prod_{s=1}^{S} (r+s-1).$$

We now compute the posterior predictive distribution Equation (3.4). For $k \geq 1$ such that $x \notin \mathbf{x}_k$, let $\boldsymbol{n}_k = (n_{k,w})_{w \in V}$ denote the word counts in cluster k. By the conjugacy property of the Dirichlet and Multinomial distributions, we have:

$$\mathrm{p}(\theta \mid \mathbf{x}_k, G_0) = \frac{1}{\Delta(\boldsymbol{\gamma} + \boldsymbol{n_k})} \exp\left(\sum_{w \in V} (n_{k,w} + \gamma_w - 1) \log(\theta_w) \right). \tag{6.50}$$

Moreover, we have

$$\begin{aligned}
\mathrm{p}(x \mid z = k, \mathbf{x}_k, G_0) &= \int_\Theta \mathrm{p}(x \mid \theta) \mathrm{p}(\theta \mid \mathbf{x}_k, G_0) d\theta, && (6.51)\\
&= \frac{1}{\Delta(\boldsymbol{\gamma} + \boldsymbol{n_k})} \int_\Theta \exp\left(\sum_{w \in V} (n_{k,w} + \ell_{x,w} + \gamma_w - 1) \log(\theta_w) \right) d\theta, && (6.52)\\
&= \frac{1}{\Delta(\boldsymbol{\gamma} + \boldsymbol{n_k})} \int_\Theta \prod_{w \in V} \theta_w^{(n_{k,w} + \ell_{x,w} + \gamma_w - 1)} d\theta, && (6.53)\\
&= \frac{\Delta(\boldsymbol{\gamma} + \boldsymbol{n}_k + \boldsymbol{\ell}_x)}{\Delta(\boldsymbol{\gamma} + \boldsymbol{n}_k)}, && (6.54)\\
&= \frac{\Gamma(\sum_{w=1}^V \gamma_w + n_{k,w})}{\Gamma(\sum_{w=1}^V \gamma_w + n_{k,w} + \ell_{x,w})} \prod_{w=1}^V \frac{\Gamma(\gamma_w + n_{k,w} + \ell_{x,w})}{\Gamma(\gamma_w + n_{k,w})}, && (6.55)\\
&= \frac{\Gamma(\sum_{w=1}^V \gamma_w + n_{k,w})}{\Gamma(\sum_{w=1}^V \gamma_w + n_{k,w} + \ell_{x,w})} \prod_{w \in x} \frac{\Gamma(\gamma_w + n_{k,w} + \ell_{x,w})}{\Gamma(\gamma_w + n_{k,w})}, && (6.56)\\
&= \frac{\prod_{w \in x} \prod_{s=1}^{\ell_{x,w}} (\gamma + n_{k,w} + s - 1)}{\prod_{s=1}^{\ell_x} (V\gamma + N_k + s - 1)}. && (6.57)
\end{aligned}$$

It is important to note that the words present in the document x don't contribute to the computation of $n_{k,w}$ and thus of N_k.

Joint prior predictive

Next, we provide the derivation of the joint prior predictive distribution.

$$\mathrm{p}\left(\mathbf{x}_h^j \mid G_0\right) = \int_\Theta \mathrm{p}(\mathbf{x}_h^j|\theta)\mathrm{p}(\theta|G_0)d\theta, \tag{6.58}$$

$$= \int_\Theta \prod_{x\in\mathbf{x}_h^j} \mathrm{p}(x|\theta)\mathrm{p}(\theta|G_0)d\theta, \tag{6.59}$$

$$= \int_\Theta \prod_{x\in\mathbf{x}_h^j}\prod_{w\in x} \mathsf{Mult}(w \mid \theta)\mathsf{Dir}(\theta \mid \boldsymbol{\gamma})d\theta, \tag{6.60}$$

$$= \int_\Theta \exp\left(\sum_{w\in V}\sum_{x\in\mathbf{x}_h^j} \ell_{x,w}\log\left(\theta_w\right) + \sum_{w\in V}\left(\gamma_w - 1\right)\log\left(\theta_w\right) - \log(\Delta(\boldsymbol{\gamma}))\right) d\theta, \tag{6.61}$$

$$= \int_\Theta \exp\left(\sum_{w\in V} n_{h,w}^j\log\left(\theta_w\right) + \sum_{w\in V}\left(\gamma_w - 1\right)\log\left(\theta_w\right) - \log(\Delta(\boldsymbol{\gamma}))\right) d\theta, \tag{6.62}$$

$$= \int_\Theta \exp\left(\sum_{w\in V}\left(n_{h,w}^j + \gamma_w - 1\right)\log\left(\theta_w\right) - \log(\Delta(\boldsymbol{\gamma}))\right) d\theta, \tag{6.63}$$

$$= \frac{1}{\Delta(\boldsymbol{\gamma})}\int_\Theta \prod_{w\in V} \theta_w^{n_{h,w}^j+\gamma_w-1} d\theta, \tag{6.64}$$

$$= \frac{\Delta(\boldsymbol{n}_h^j + \boldsymbol{\gamma})}{\Delta(\boldsymbol{\gamma})}, \tag{6.65}$$

$$= \frac{\Gamma(\sum_{w=1}^V \gamma_w)}{\Gamma(\sum_{w=1}^V n_{h,w}^j + \gamma_w)}\prod_{w=1}^V \frac{\Gamma(n_{h,w}^j + \gamma_w)}{\Gamma(\gamma_w)}, \tag{6.66}$$

$$= \frac{\Gamma(\sum_{w=1}^V \gamma_w)}{\Gamma(\sum_{w=1}^V n_{h,w}^j + \gamma_w)}\prod_{w\in\mathbf{x}_h^j} \frac{\Gamma(n_{h,w}^j + \gamma_w)}{\Gamma(\gamma_w)}, \tag{6.67}$$

$$= \frac{\prod_{w\in\mathbf{x}_h^j}\prod_{s=1}^{n_{h,w}^j}(\gamma + s - 1)}{\prod_{s=1}^{N_h^j}(V\gamma + s - 1)}. \tag{6.68}$$

Joint posterior predictive

We now compute the posterior predictive distribution Equation (3.5). For $k \geq 1$ such that $\mathbf{x}_h^j \notin \mathbf{x}_k$(this means that the local cluster h of worker j is not in the global cluster k), let $\boldsymbol{n}_k = (n_{k,w})_{w\in V}$ denote the word counts in global cluster k. By the conjugacy property of the Dirichlet and Multinomial distributions,

we have:

$$\mathrm{p}(\theta \mid \mathbf{x}_k, G_0) = \mathsf{Dir}(\theta \mid \boldsymbol{\gamma} + \boldsymbol{n}_k), \tag{6.69}$$

$$= \frac{1}{\Delta(\boldsymbol{\gamma} + \boldsymbol{n_k})} \exp\left(\sum_{w \in V} (n_{k,w} + \gamma_w - 1)\log(\theta_w)\right). \tag{6.70}$$

Moreover, we have $\mathrm{p}(\mathbf{x}_h^j \mid z_h^j = k, \mathbf{x}_k, G_0)$

$$= \int_\Theta \mathrm{p}(\mathbf{x}_h^j \mid \theta)\mathrm{p}(\theta \mid \mathbf{x}_k, G_0)d\theta, \tag{6.71}$$

$$= \int_\Theta \prod_{x \in \mathbf{x}_h^j} \mathrm{p}(x \mid \theta)\mathrm{p}(\theta \mid \mathbf{x}_k, G_0)d\theta, \tag{6.72}$$

$$= \int_\Theta \prod_{x \in \mathbf{x}_h^j} \prod_{w \in x} \mathsf{Mult}(w \mid \theta)\mathsf{Dir}(\theta \mid \boldsymbol{\gamma} + \boldsymbol{n}_k)d\theta, \tag{6.73}$$

$$= \frac{1}{\Delta(\boldsymbol{\gamma} + \boldsymbol{n_k})} \int_\Theta \exp\left(\sum_{w \in V} \sum_{x \in \mathbf{x}_h^j} \ell_{x,w} \log(\theta_w) + \sum_{w \in V} (n_{k,w} + \gamma_w - 1)\log(\theta_w)\right) d\theta, \tag{6.74}$$

$$= \frac{1}{\Delta(\boldsymbol{\gamma} + \boldsymbol{n_k})} \int_\Theta \exp\left(\sum_{w \in V} n_{h,w}^j \log(\theta_w) + \sum_{w \in V} (n_{k,w} + \gamma_w - 1)\log(\theta_w)\right) d\theta, \tag{6.75}$$

$$= \frac{1}{\Delta(\boldsymbol{\gamma} + \boldsymbol{n_k})} \int_\Theta \exp\left(\sum_{w \in V} (n_{k,w} + n_{h,w}^j + \gamma_w - 1)\log(\theta_w)\right) d\theta, \tag{6.76}$$

$$= \frac{1}{\Delta(\boldsymbol{\gamma} + \boldsymbol{n_k})} \int_\Theta \prod_{w \in V} \theta_w^{(n_{k,w} + n_{h,w}^j + \gamma_w - 1)} d\theta, \tag{6.77}$$

$$= \frac{\Delta(\boldsymbol{\gamma} + \boldsymbol{n}_k + \boldsymbol{n}_h^j)}{\Delta(\boldsymbol{\gamma} + \boldsymbol{n}_k)}, \tag{6.78}$$

$$= \frac{\Gamma(\sum_{w=1}^V \gamma_w + n_{k,w})}{\Gamma(\sum_{w=1}^V \gamma_w + n_{k,w} + n_{h,w}^j)} \prod_{w=1}^V \frac{\Gamma(\gamma_w + n_{k,w} + n_{h,w}^j)}{\Gamma(\gamma_w + n_{k,w})}, \tag{6.79}$$

$$= \frac{\Gamma(\sum_{w=1}^V \gamma_w + n_{k,w})}{\Gamma(\sum_{w=1}^V \gamma_w + n_{k,w} + n_{h,w}^j)} \prod_{w \in \mathbf{x}_h^j} \frac{\Gamma(\gamma_w + n_{k,w} + n_{h,w}^j)}{\Gamma(\gamma_w + n_{k,w})}, \tag{6.80}$$

$$= \frac{\prod_{w \in \mathbf{x}_h^j} \prod_{s=1}^{n_{h,w}^j} (\gamma + n_{k,w} + s - 1)}{\prod_{s=1}^{N_h^j} (V\gamma + N_k + s - 1)}. \tag{6.81}$$

6.8 . Feature extraction

Cores	NMI	ARI
2	0.610	0.330
4	0.646	0.430
6	0.651	0.456
8	0.633	0.410
10	0.670	0.472

Table 6.1: Clustering quality metrics (NMI and ARI) of DisCGS on the DBpedia dataset with varying numbers of CPU cores.

In the image datasets, a feature extraction step is performed using a variational auto-encoder, which encodes each image into an 8-dimensional vector. The optimizer used for training is ADAM, with a learning rate of 0.005. The training process spans 100 epochs. The neural network is trained on eight Nvidia Tesla V100 GPUs with a capacity of 32 GiB. These GPUs are available on the Gemini machine, hosted within the cluster grid5000. The architecture of the auto-encoder is illustrated in Figure 6.1.

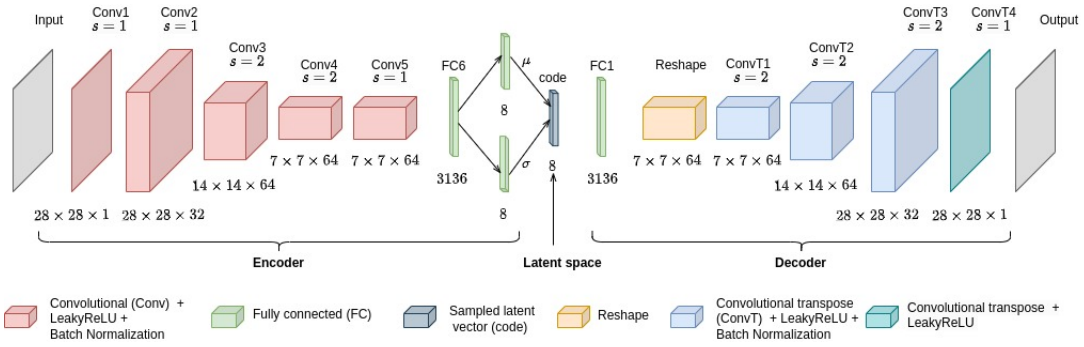

Figure 6.1: The architecture of the variational auto-encoder used to extract features from the images.

6.9 . Clustering quality when increasing number of cores

The Table 6.1 presents the clustering quality metrics—Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI)—for DisCGS on the DBpedia dataset containing 300000 documents as the number of CPU cores increases.

The results show that the clustering quality of DisCGS remains stable or improves as the number of CPU cores increases, based on both Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) metrics. Specifically, NMI values stay consistently high, rising slightly from 0.61 to 0.67, which confirms that the distributed inference approach preserves robust clustering accuracy. These findings demonstrate that DisCGS effectively utilizes additional computational resources to speed up inference without compromising clus-

tering performance.

List of Figures

List of Tables

List of Algorithms

Glossary

ACC	Accuracy
AIC	Akaike Information Criterion
AMI	Adjusted Mutual Information
ARI	Adjusted Rand Index
BIC	Bayesian Information Criterion
BNP	Bayesian non-parametric
CGS	Collapsed Gibbs Sampler
CRP	Chinese Restaurant Process
DPMM	Dirichlet Process Mixture Model
EM	Expectation-Maximization
FN	False Negatives
FP	False Positives
GEM	Griffiths-Engen-McCloskey
GMM	Gaussian Mixture Model
HDFS	Hadoop Distributed File System
i.i.d.	independent and identically distributed
ICL	Integrated Completed
MCMC	Markov Chain Monte Carlo
MH	Metropolis-Hastings
MI	Mutual Information
MLE	Maximum Likelihood Estimator
MoG	Mixture of Gaussians
NMI	Normalized Mutual Information

NPLBM Non Parametric Latent Block Model

RDD Resilient Distributed Dataset

RI Rand Index

SB Stick-Breaking

TN True Negatives

TP True Positives

VI Variation of Information

Bibliography

[1] Wael Abd-Almageed and Larry S Davis. Density estimation using mixtures of mixtures of gaussians. In *European Conference on Computer Vision*, pages 410–422. Springer, 2006.

[2] Nigatu A. Adossa, Kalle T. Rytkönen, and Laura L. Elo. Dirichlet process mixture models for single-cell rna-seq clustering. *Biology Open*, 11(4), April 2022.

[3] Nigatu A. Adossa, Kalle T. Rytkönen, and Laura L. Elo. Dirichlet process mixture models for single-cell RNA-seq clustering. *Biology Open*, 11(4), 04 2022.

[4] Stefan Aeberhard and M. Forina. Wine. UCI Machine Learning Repository, 1991. DOI: https://doi.org/10.24432/C5PC7J.

[5] Séverine Affeldt, Lazhar Labiod, and Mohamed Nadif. Spectral clustering via ensemble deep autoencoder learning (sc-edae). *Pattern Recognition*, 108:107522, 2020.

[6] Hirotugu Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 2003.

[7] LN Fred Ana and Anil K Jain. Robust data clustering. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–II. IEEE, 2003.

[8] Michael R. Anderberg. Chapter 6 - hierarchical clustering methods. In Michael R. Anderberg, editor, *Cluster Analysis for Applications*, Probability and Mathematical Statistics: A Series of Monographs and Textbooks, pages 131–155. Academic Press, 1973.

[9] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.

[10] Charles E Antoniak. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The annals of statistics*, pages 1152–1174, 1974.

[11] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[12] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1027–1035, 2007.

[13] Tevfik Aytekin and Mahmut Özge Karakaya. Clustering-based diversity improvement in top-n recommendation. *J. Intell. Inf. Syst.*, 42(1):1–18, February 2014.

[14] Sivaraman Balakrishnan, Martin J Wainwright, and Bin Yu. Statistical guarantees for the em algorithm: From population to sample-based analysis. 2017.

[15] James C. Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2):191–203, 1984.

[16] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE transactions on pattern analysis and machine intelligence*, 22(7):719–725, 2002.

[17] Christophe Biernacki, Julien Jacques, and Christine Keribin. A survey on model-based co-clustering: high dimension and estimation challenges. *Journal of Classification*, 40(2):332–381, 2023.

[18] David Blackwell. Conditional expectation and unbiased sequential estimation. *The Annals of Mathematical Statistics*, pages 105–110, 1947.

[19] David M. Blei and Michael I. Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121 – 143, 2006.

[20] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[21] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[22] Christian Borgelt. *Prototype-Based Classification and Clustering*. PhD thesis, University of Magdeburg, 2005. Habilitation thesis, 2006.

[23] George EP Box and David R Cox. An analysis of transformations. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 26(2):211–243, 1964.

[24] T Caliński and JA Harabasz. dendrite method for cluster analysiscommun. *Stat. Theory Methods1974311*, 1974.

[25] Bagher Rahimpour Cami, Hamid Hassanpour, and Hoda Mashayekhi. User preferences modeling using dirichlet process mixture model for a content-based recommender system. *Knowledge-Based Systems*, 163:644–655, 2019.

[26] Ricardo JGB Campello, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(2):e1343, 2020.

[27] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[28] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache flink: Stream and batch processing in a single engine. *The Bulletin of the Technical Committee on Data Engineering*, 38(4), 2015.

[29] M. Emre Celebi. *Partitional Clustering Algorithms*. Springer, 2014.

[30] Saptarshi Chakraborty, Debolina Paul, and Swagatam Das. Hierarchical clustering with optimal transport. *Statistics & Probability Letters*, 163:108781, 2020.

[31] Jason Chang and John W Fisher III. Parallel sampling of dp mixture models using sub-cluster splits. *Advances in Neural Information Processing Systems*, 26, 2013.

[32] Jianlong Chang, Yiwen Guo, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep discriminative clustering analysis. *arXiv preprint arXiv:1905.01681*, 2019.

[33] Timothy Chappell, Shlomo Geva, and James Hogan. K-means clustering of biological sequences. In *Proceedings of the 22nd Australasian Document Computing Symposium*, ADCS '17, New York, NY, USA, 2017. Association for Computing Machinery.

[34] Xin Chen and Anderson Ye Zhang. Achieving optimal clustering in gaussian mixture models with anisotropic covariance structures. *Advances in Neural Information Processing Systems*, 37:113698–113741, 2024.

[35] Dongdong Cheng, Qingsheng Zhu, Jinlong Huang, Quanwang Wu, and Lijun Yang. A local cores-based hierarchical clustering algorithm for data sets with complex structures. *Neural Computing and Applications*, 31(11):8051–8068, 2019.

[36] Xiang Cheng, Sen Su, Lixin Gao, and Jiangtao Yin. Co-clusterd: A distributed framework for data co-clustering with sequential updates. *IEEE Transactions on Knowledge and Data Engineering*, 27(12):3231–3244, 2015.

[37] Dondapati Chowdary, Jessica Lathrop, Joanne Skelton, Kathleen Curtin, Thomas Briggs, Yi Zhang, Jack Yu, Yixin Wang, and Abhijit Mazumder. Prognostic gene expression signatures can be measured in tissues collected in rnalater preservative. *J Mol Diagn*, 8(1):31–39, Feb 2006.

[38] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: Extending mnist to handwritten letters. *International Joint Conference on Neural Networks*, pages 2921–2926, 2017.

[39] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[40] Sanjoy Dasgupta and Leonard Schulman. A probabilistic analysis of em for mixtures of separated, spherical gaussians. *Journal of Machine Learning Research*, 8(2), 2007.

[41] D Davies and D Bouldin. A cluster separation measure: Ieee transactions on pattern analysis and machine intelligence. itpidj 0162-8828, pami-1, 2 224–227. *Crossref Web of Science*, 21868852, 1979.

[42] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. volume 51, pages 137–150, 01 2004.

[43] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B*, 39(1):1–22, 1977.

[44] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[45] Meghana Deodhar, Clinton Jones, and Joydeep Ghosh. Parallel simultaneous co-clustering and learning with map-reduce. In *2010 IEEE International Conference on Granular Computing*, pages 149–154. IEEE, 2010.

[46] Or Dinari, Angel Yu, Oren Freifeld, and John W Fisher III. Distributed mcmc inference in dirichlet process mixture models using julia. *19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 518–525, 2019.

[47] Dheeru Dua and Casey Graff. UCI machine learning repository.

[48] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[49] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[50] Michael B Eisen, Paul T Spellman, Patrick O Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[51] Michael D. Escobar. Estimating normal means with a dirichlet process prior. *Journal of the American Statistical Association*, 89(425):268–277, 1994.

[52] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231. AAAI Press, 1996.

[53] Absalom E. Ezugwu, Abiodun M. Ikotun, Olaide O. Oyelade, Laith Abualigah, Jeffery O. Agushaka, Christopher I. Eke, and Andronicus A. Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022.

[54] Francesco Folino, Gianluigi Greco, Antonella Guzzo, and Luigi Pontieri. Scalable parallel co-clustering over multiple heterogeneous data types. In *2010 International Conference on High Performance Computing & Simulation*, pages 529–535. IEEE, 2010.

[55] Florent Forest, Jérôme Lacaille, Mustapha Lebbah, and Hanene Azzag. A generic and scalable pipeline for large-scale analytics of continuous aircraft engine data. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1918–1924, 2018.

[56] Pasi Fränti and Sami Sieranoja. Clustering accuracy. 2024.

[57] Yarin Gal and Zoubin Ghahramani. Pitfalls in the use of parallel inference for the dirichlet process. In *International Conference on Machine Learning*, pages 208–216. PMLR, 2014.

[58] Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data clustering: theory, algorithms, and applications*. SIAM, 2020.

[59] Hong Ge, Yutian Chen, Moquan Wan, and Zoubin Ghahramani. Distributed inference for dirichlet process mixture models. In *International Conference on Machine Learning*, pages 2276–2284. PMLR, 2015.

[60] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*. 1995.

[61] Etienne Goffinet. *Multi-Block Clustering and Analytical Visualization of Massive Time Series from Autonomous Vehicle Simulation*. Theses, Université Paris 13 Sorbonne Paris Nord, December 2021.

[62] Etienne Goffinet, Mustapha Lebbah, Hanane Azzag, Loïc Giraldi, and Anthony Coutant. Multivariate time series multi-coclustering. application to advanced driving assistance system validation. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2021.

[63] Etienne Goffinet, Mustapha Lebbah, Hanane Azzag, Giraldi Loïc, and Anthony Coutant. Functional non-parametric latent block model: A multivariate time series clustering approach for autonomous driving validation. *Comput. Stat. Data Anal.*, 176(C), dec 2022.

[64] Gérard Govaert and Mohamed Nadif. Clustering with block mixture models. *Pattern Recognition*, 36(2):463–473, 2003.

[65] Gérard Govaert and Mohamed Nadif. *Co-clustering: models, algorithms and applications*. John Wiley & Sons, 2013.

[66] Gianluigi Greco, Antonella Guzzo, and Luigi Pontieri. Coclustering multiple heterogeneous domains: Linear combinations and agreements. *IEEE Transactions on Knowledge and Data Engineering*, 22(12):1649–1663, 2010.

[67] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: An efficient clustering algorithm for large databases. *SIGMOD Record*, 27(2):73–84, 1998.

[68] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

[69] Jiaxun Guo, Wentao Fan, Manar Amayri, and Nizar Bouguila. Deep clustering analysis via variational autoencoder with gamma mixture latent embeddings. *Neural Networks*, 183:106979, 2025.

[70] Daniel Hanisch, Alexander Zien, and Ralf Zimmer. Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18, 05 2002.

[71] Satoshi Hara and Yuichi Yoshida. Average sensitivity of decision tree learning. In *The 11th International Conference on Learning Representations (ICLR)*, 2023.

[72] Miss Harshada, S Deshmukh, and L Ramteke. Comparing the techniques of cluster analysis for big data. *International Journal of advanced Research in Computer Engineering & Technology (IJARCET)*, page 17, 2015.

[73] Reinhard Heckel, Michail Vlachos, Thomas Parnell, and Celestine Dünner. Scalable and interpretable product recommendations via overlapping co-clustering. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1033–1044. IEEE, 2017.

[74] Katherine A. Heller and Zoubin Ghahramani. Bayesian hierarchical clustering. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 297–304, New York, NY, USA, 2005. ACM.

[75] Alexander Hinneburg and Hans-Henning Gabriel. Denclue 2.0: Fast clustering based on kernel density estimation. In *International symposium on intelligent data analysis*, pages 70–80. Springer, 2007.

[76] Alexander Hinneburg, Daniel A Keim, et al. *An efficient approach to clustering in large multimedia databases with noise*, volume 98. Bibliothek der Universität Konstanz Konstanz, Germany, 1998.

[77] Joshua Zhexue Huang. A fast clustering algorithm to cluster very large categorical data sets in data mining. In *Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1997.

[78] Zhexue Huang et al. Clustering large data sets with mixed numeric and categorical values. In *Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 21–34. Citeseer, 1997.

[79] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[80] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[81] Sonia Jain and Radford M Neal. A split-merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of computational and Graphical Statistics*, 13(1):158–182, 2004.

[82] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

[83] Chi Jin, Yuchen Zhang, Sivaraman Balakrishnan, Martin J Wainwright, and Michael I Jordan. Local maxima in the likelihood of gaussian mixture models: Structural results and algorithmic consequences. *Advances in neural information processing systems*, 29, 2016.

[84] Stephen C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[85] Leonard Kaufman and Peter Rousseeuw. *Clustering Large Data Sets*, pages 425–437. December 1986.

[86] Leonard Kaufman and Peter Rousseeuw. Clustering by means of medoids. *Data Analysis Based on the L1-Norm and Related Methods*, pages 405–416, January 1987.

[87] Leonard Kaufman and Peter Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990.

[88] Reda Khoufache, Anisse Belhadj, Hanene Azzag, and Mustapha Lebbah. Distributed mcmc inference for bayesian non-parametric latent block model. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 271–283. Springer, 2024.

[89] Reda Khoufache, Mustapha Lebbah, Hanene Azzag, Etienne Goffinet, and Djamel Bouchaffra. Distributed collapsed gibbs sampler for dirichlet process mixture models in federated learning. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*, pages 815–823. SIAM, 2024.

[90] Reda Khoufache, Mustapha Lebbah, Hanene Azzag, Etienne Goffinet, and Djamel Bouchaffra. A distributed inference algorithm for dirichlet process mixture models with exponential family components. *Neurocomputing*, page 131119, 2025.

[91] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[92] Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13(4):703–716, 2003.

[93] Athanasios Kottas. Dirichlet process mixtures of beta distributions, with applications to density and intensity estimation. 2006.

[94] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(3):231–240, 2011.

[95] Soh Kumabe and Yuichi Yoshida. Average sensitivity of dynamic programming. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1925–1961, 2022.

[96] Soh Kumabe and Yuichi Yoshida. Average sensitivity of the knapsack problem. In *30th Annual European Symposium on Algorithms (ESA)*, pages 75:1–75:14, 2022.

[97] Uğurhan Kutbay et al. Partitional clustering. *Recent Applications in Data Clustering*, 10, 2018.

[98] Jeongyeol Kwon and Constantine Caramanis. The em algorithm gives sample-optimality for learning mixtures of well-separated gaussians. In *Conference on Learning Theory (COLT)*, pages 2425–2487, 2020.

[99] Yuping Lai, Yuan Ping, Ke Xiao, Bin Hao, and Xiufeng Zhang. Variational bayesian inference for a dirichlet process mixture of beta distributions and application. *Neurocomputing*, 278:23–33, 2018.

[100] Dao Lam and Donald Wunsch. Clustering. In *Academic Press Library in Signal Processing*, volume 1, pages 1115–1149. Academic Press, 2014.

[101] Michael Lewis-Beck, Alan Bryman, and Tim Futing Liao, editors. *The SAGE Encyclopedia of Social Science Research Methods*. SAGE Publishing, United States, 2004.

[102] Lishuai Li, Santanu Das, R. John Hansman, Rafael Palacios, and Ashok N. Srivastava. Analysis of flight data using clustering techniques for detecting abnormal operations. *Journal of Aerospace Information Systems*, 12(9):587–598, 2015.

[103] Chen Liang, Wenguan Wang, Jiaxu Miao, and Yi Yang. Gmmseg: Gaussian mixture based generative semantic segmentation models. *Advances in Neural Information Processing Systems*, 35:31360–31375, 2022.

[104] Jun S Liu, Wing Hung Wong, and Augustine Kong. Covariance structure of the gibbs sampler with applications to the comparisons of estimators and augmentation schemes. *Biometrika*, pages 27–40, 1994.

[105] Tyler J. Loftus, Benjamin Shickel, Jeremy A. Balch, Patrick J. Tighe, Kenneth L. Abbott, Brian Fazzone, Erik M. Anderson, Jared Rozowsky, Tezcan Ozrazgat-Baslanti, Yuanfang Ren, Scott A. Berceli, William R. Hogan, Philip A. Efron, J. Randall Moorman, Parisa Rashidi, Gilbert R. Upchurch, and Azra Bihorac. Phenotype clustering in health care: A narrative review for clinicians. *Frontiers in Artificial Intelligence*, 5, 2022.

[106] Dan Lovell, Jonathan Malmaud, Ryan P Adams, and Vikash K Mansinghka. Clustercluster: parallel markov chain monte carlo for dirichlet process mixtures. *arXiv preprint arXiv:1304.2302*, 2013.

[107] Zhentai Lu, Jun Jiang, Wei Yang, Qianjin Feng, and Wufan Chen. Multimodal brain-tumor segmentation based on dirichlet process mixture model with anisotropic diffusion and markov random field prior. *Computational and Mathematical Methods in Medicine.*, September 2014.

[108] E. Laxmi Lydia, P. Govindaswamy, S. Lakshmanaprabu, and D. Ramya. Document clustering based on text mining k-means algorithm using euclidean distance similarity. *Journal of Advanced Research in Dynamical & Control Systems*, 10, 2018.

[109] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. University of California Press, 1967.

[110] Jocelyn Mazarura, Alta de Waal, and Pieter de Villiers. A gamma-poisson mixture topic model for short text. *Mathematical Problems in Engineering*, 2020(1), 2020.

[111] Ryan McConville, Raul Santos-Rodriguez, Robert J Piechocki, and Ian Craddock. N2d:(not too) deep clustering via clustering the local manifold of an autoencoded embedding. In *2020 25th international conference on pattern recognition (ICPR)*, pages 5145–5152. IEEE, 2021.

[112] Geoffrey J McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2000.

[113] Edward Meeds and Sam Roweis. Nonparametric bayesian biclustering. Technical report, Technical report, University of Toronto, 2007.

[114] Khadidja Meguelati, Bénédicte Fontez, Nadine Hilgert, and Florent Masseglia. Dirichlet process mixture models made scalable and effective by means of massive distribution. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 502–509, 2019.

[115] Jeffrey W Miller. An elementary derivation of the chinese restaurant process from sethuraman's stick-breaking process. *Statistics & Probability Letters*, 146:112–117, 2019.

[116] Shogo Murai and Yuichi Yoshida. Sensitivity analysis of centralities on unweighted networks. In *The World Wide Web Conference*, pages 1332–1342, 2019.

[117] Kevin P Murphy. Conjugate bayesian analysis of the gaussian distribution. *def*, 1(2σ2):16, 2007.

[118] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[119] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms, 2011.

[120] Zahra Nazari and Dongshik Kang. A new hierarchical clustering algorithm with intersection points. In *2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, pages 1–5, 2018.

[121] Radford M. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[122] Radford M Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

[123] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. Technical report, University of British Columbia, Canada, 1994.

[124] Nisha and Puneet Jai Kaur. Cluster quality based performance evaluation of hierarchical clustering method. In *Proceedings of the 1st International Conference on Next Generation Computing Technologies (NGCT)*, pages 649–653, 2015.

[125] Catherine L Nutt, D. R. Mani, Rebecca A Betensky, Pablo Tamayo, J. Gregory Cairncross, Christine Ladd, Ute Pohl, Christian Hartmann, Margaret E McLaughlin, Tracy T Batchelor, Peter M Black, Andreas von Deimling, Scott L Pomeroy, Todd R Golub, and David N Louis. Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. *Cancer Res*, 63(7):1602–1607, Apr 2003.

[126] Gbeminiyi John Oyewole and George Alex Thopil. Data clustering: Application and trends. *Artificial Intelligence Review*, 56(7):6439–6475, 2022.

[127] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[128] Spiros Papadimitriou and Jimeng Sun. Disco: Distributed co-clustering with map-reduce: A case study towards petabyte-scale end-to-end mining. In *2008 Eighth IEEE International Conference on Data Mining*, pages 512–521. IEEE, 2008.

[129] Pan Peng and Yuichi Yoshida. Average sensitivity of spectral clustering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1132–1140, 2020.

[130] Xingcheng Ran, Yue Xi, Yonggang Lu, Xiangwen Wang, and Zhenyu Lu. Comprehensive survey on hierarchical clustering algorithms and the recent developments. *Artificial Intelligence Review*, 56(8):8219–8264, 2022.

[131] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[132] Aarthi Reddy, Meredith Ordway-West, Melissa Lee, Matt Dugan, Joshua Whitney, Ronen Kahana, Brad Ford, Johan Muedsam, Austin Henslee, and Max Rao. Using gaussian mixture models to detect outliers in seasonal univariate network traffic. In *2017 IEEE Security and Privacy Workshops (SPW)*, pages 229–234. IEEE, 2017.

[133] Chandan K. Reddy and Bhanukiran Vinzamuri. A survey of partitional and hierarchical clustering algorithms. In *Data Clustering*, pages 87–110. Chapman and Hall/CRC, 2018.

[134] Yazhou Ren, Jingyu Pu, Zhimeng Yang, Jie Xu, Guofeng Li, Xiaorong Pu, Philip S Yu, and Lifang He. Deep clustering: A comprehensive survey. *IEEE transactions on neural networks and learning systems*, 2024.

[135] Yazhou Ren, Jingyu Pu, Zhimeng Yang, Jie Xu, Guofeng Li, Xiaorong Pu, Philip S. Yu, and Lifang He. Deep clustering: A comprehensive survey. *IEEE Transactions on Neural Networks and Learning Systems*, 36(4):5858–5878, 2025.

[136] Yazhou Ren, Ni Wang, Mingxia Li, and Zenglin Xu. Deep density-based image clustering. *Knowledge-Based Systems*, 197:105841, 2020.

[137] Eréndira Rendón, Itzel Abundez, Alejandra Arizmendi, and Elvia M Quiroz. Internal versus external cluster validation indexes. *International Journal of computers and communications*, 5(1):27–34, 2011.

[138] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005.

[139] Clara Rocha and Luis Dias. Mpoc: An agglomerative algorithm for multi-criteria partially ordered clustering. *4OR: Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 11, February 2013.

[140] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *science*, 344(6191):1492–1496, 2014.

[141] Mayra Z. Rodriguez, Cesar H. Comin, Dalcimar Casanova, Odemir M. Bruno, Diego R. Amancio, Luciano da F. Costa, and Francisco A. Rodrigues. Clustering algorithms: A comparative approach. *PLOS ONE*, 14(1):1–34, 2019.

[142] Frédéric Ros, Serge Guillaume, Mohamed El Hajji, and Rabia Riad. Kd-mutual: A novel clustering algorithm combining mutual neighboring and hierarchical approaches using a new selection criterion. *Knowledge-Based Systems*, 204:106220, 2020.

[143] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[144] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[145] Maurice Roux. A comparative study of divisive and agglomerative hierarchical clustering algorithms. *Journal of Classification*, 35(2):345–366, 2018.

[146] Joerg Sander, Xuejie Qin, Zhiyong Lu, Nan Niu, and Alex Kovarsky. Automatic extraction of clusters from hierarchical clustering representations. In *Advances in Knowledge Discovery and Data Mining*, pages 75–87, 2003.

[147] Amit Saxena, Mukesh Prasad, Akshansh Gupta, Neha Bharill, Om Prakash Patel, Aruna Tiwari, Meng Joo Er, Weiping Ding, and Chin-Teng Lin. A review of clustering techniques and developments. *Neurocomputing*, 267:664–681, 2017.

[148] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[149] Shokri Z. Selim and Mohamed A. Ismail. K-means-type algorithms: A generalized convergence theorem and characterization of local opti-

mality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6:81–87, 1984.

[150] Jayaram Sethuraman. A constructive definition of dirichlet priors. *Statistica sinica*, pages 639–650, 1994.

[151] Christopher W. Seymour, Jason N. Kennedy, Shu Wang, Chung-Chou H. Chang, Corrine F. Elliott, Zhongying Xu, Scott Berry, Gilles Clermont, Gregory Cooper, Hernando Gomez, David T. Huang, John A. Kellum, Qi Mi, Steven M. Opal, Victor Talisa, Tom van der Poll, Shyam Visweswaran, Yoram Vodovotz, Jeremy C. Weiss, Donald M. Yealy, Sachin Yende, and Derek C. Angus. Derivation, validation, and potential treatment implications of novel clinical phenotypes for sepsis. *JAMA*, 321(20):2003–2017, May 2019.

[152] Jeff Shafer, Scott Rixner, and Alan Cox. The hadoop distributed filesystem: Balancing portability and performance. pages 122–133, 03 2010.

[153] Sohil Atul Shah and Vladlen Koltun. Robust continuous clustering. *Proceedings of the National Academy of Sciences*, 114(37):9814–9819, 2017.

[154] S.D. Silvey. *Statistical Inference*. Monographs on statistics and applied probability. 2003.

[155] Jaswinder Singh and Damanpreet Singh. A comprehensive review of clustering techniques in artificial intelligence for knowledge discovery: Taxonomy, challenges, applications and future prospects. *Advanced Engineering Informatics*, 62:102799, 2024.

[156] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy: The Principles and Practice of Numerical Classification*. W.H. Freeman and Company, 1973.

[157] Velmurugan Thambusamy and Santhanam T. A survey of partition based clustering algorithms in data mining: An experimental approach. *Information Technology Journal*, 10(3):478–484, March 2011.

[158] Ankit Toshniwal, Siddarth Taneja, Amit Shukla, Karthik Ramasamy, Jignesh M Patel, Sanjeev Kulkarni, Jason Jackson, Krishna Gade, Maosong Fu, Jake Donham, et al. Storm@ twitter. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 147–156, 2014.

[159] Nithin Varma and Yuichi Yoshida. Average sensitivity of graph algorithms. *SIAM Journal on Computing*, 52(4):1039–1081, 2023.

[160] Erica Vidal, Pablo Miguel Granitto, and Ariel Bayá. Discussing a new divisive hierarchical clustering algorithm. In *XLIII Jornadas Argentinas de*

Informática e Investigación Operativa (43JAIIO)-XV Argentine symposium on artificial intelligence (ASAI)(Buenos Aires, 2014), 2014.

[161] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. 2journal of machine learning research 11 (2010), 2837–2854, 2010.

[162] Stephen G Walker. Sampling the dirichlet mixture model with slices. *Communications in Statistics—Simulation and Computation®*, 36(1):45–54, 2007.

[163] Hongjun Wang, Yi Song, Wei Chen, Zhipeng Luo, Chongshou Li, and Tianrui Li. A survey of co-clustering. *ACM Trans. Knowl. Discov. Data*, 18(9), November 2024.

[164] Ruohui Wang and Dahua Lin. Scalable estimation of dirichlet process mixture models on distributed data. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, pages 4632–4639, 2017.

[165] Xiuxi Wei, Zhihui Zhang, Huajuan Huang, and Yongquan Zhou. An overview on deep clustering. *Neurocomputing*, 590:127761, 2024.

[166] Tom White. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[167] Sinead Williamson, Avinava Dubey, and Eric Xing. Parallel markov chain monte carlo for nonparametric mixture models. In *International Conference on Machine Learning*, pages 98–106. PMLR, 2013.

[168] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[169] Liu Xiaojun. An improved clustering-based collaborative filtering recommendation algorithm. *Cluster Computing*, 20(2):1281–1288, June 2017.

[170] Tengke Xiong, Shengrui Wang, Andre Mayers, and Ernest Monga. Dhcc: Divisive hierarchical clustering of categorical data. *Data Mining and Knowledge Discovery*, 24:103–135, 2012.

[171] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2, 2015.

[172] Ji Xu, Daniel J Hsu, and Arian Maleki. Global analysis of expectation maximization for mixtures of two gaussians. *Advances in Neural Information Processing Systems*, 29, 2016.

[173] Lei Xu and Michael I Jordan. On convergence properties of the em algorithm for gaussian mixtures. *Neural Computation*, 8(1):129–151, 1996.

[174] Miin-Shen Yang, Chien-Yo Lai, and Chih-Ying Lin. A robust em clustering algorithm for gaussian mixture models. *Pattern Recognition*, 45(11):3950–3961, 2012.

[175] Xiaojiang Yang, Junchi Yan, Yu Cheng, and Yizhe Zhang. Learning deep generative clustering via mutual information maximization. *IEEE Transactions on Neural Networks and Learning Systems*, 34(9):6263–6275, 2022.

[176] Jianhua Yin and Jianyong Wang. A dirichlet multinomial mixture model-based approach for short text clustering. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 233–242, New York, NY, USA, 2014. Association for Computing Machinery.

[177] Jianhua Yin and Jianyong Wang. A dirichlet multinomial mixture model-based approach for short text clustering. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 233–242, 2014.

[178] Jianhua Yin and Jianyong Wang. A dirichlet multinomial mixture model-based approach for short text clustering. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*, pages 233–242, 2014.

[179] Jianhua Yin and Jianyong Wang. A model-based approach for text clustering with outlier detection. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 625–636. IEEE, 2016.

[180] Yuichi Yoshida and Shinji Ito. Average sensitivity of euclidean k-clustering. *Advances in Neural Information Processing Systems*, 35:32487–32498, 2022.

[181] Yuichi Yoshida and Samson Zhou. Sensitivity analysis of the maximum matching problem. In *12th Innovations in Theoretical Computer Science Conference (ITCS)*, 2021.

[182] Guoshen Yu and Guillermo Sapiro. Statistical compressed sensing of gaussian mixture models. *IEEE Transactions on Signal Processing*, 59(12):5842–5858, 2011.

[183] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. pages 2–2, 04 2012.

[184] Matei Zaharia, Mosharaf Chowdhury, Michael Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 10:10–10, 07 2010.

[185] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, SIGMOD '96, pages 103–114, New York, NY, USA, 1996. Association for Computing Machinery.

[186] Shi Zhong, Taghi M Khoshgoftaar, and Naeem Seliya. Clustering-based network intrusion detection. *International Journal of Reliability, Quality and Safety Engineering*, 14(02):169–187, 2007.

[187] Sheng Zhou, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Zhao Li, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, and Martin Ester. A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions. *ACM Comput. Surv.*, 57(3), 2024.