Distributed Collapsed Gibbs Sampler for Dirichlet Process Mixture Models in Federated Learning

Reda Khoufache, Mustapha Lebbah, Hanene Azzag, Etienne Goffinet, Djamel Bouchaffra

Distributed Collapsed Gibbs Sampler for Dirichlet Process Mixture Models in Federated Learning

Reda Khoufache
DAVID Lab
University of Versailles, Paris-Saclay University
Versailles, France
reda.khoufache@uvsq.fr

Mustapha Lebbah
DAVID Lab
University of Versailles, Paris-Saclay University
Versailles, France
mustapha.lebbah@uvsq.fr

Hanene Azzag
LIPN (CNRS UMR 7030)
Sorbonne Paris Nord University
Villetaneuse, France
azzag@univ-paris13.fr

Etienne Goffinet
Technology Innovation Institute
Abu Dhabi, United Arab Emirates
etienne.goffinet@tii.ae

Djamel Bouchaffra
CDAT
Algiers, Algeria
djamel.bouchaffra@gmail.com

Abstract

Dirichlet Process Mixture Models (DPMMs) are widely used to address clustering problems. Their main advantage lies in their ability to automatically estimate the number of clusters during the inference process through the Bayesian non-parametric framework. However, the inference becomes considerably slow as the dataset size increases. This paper proposes a new distributed Markov Chain Monte Carlo (MCMC) inference method for DPMMs (DisCGS) using sufficient statistics. Our approach uses the collapsed Gibbs sampler and is specifically designed to work on distributed data across independent and heterogeneous machines, which habilitates its use in horizontal federated learning. Our method achieves highly promising results and notable scalability. For instance, with a dataset of 100K data points, 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 code source is publicly available at `https://github.com/redakhoufache/DisCGS`.

Keywords: Federated learning, Distributed computing, Dirichlet process mixture models, Markov Chain Monte Carlo, Bayesian non-parametric

1 Introduction

Clustering is an unsupervised learning method that aims to partition data into clusters such that elements of the same cluster are similar while those of different clusters are dissimilar. It plays a crucial role in various domains, such as data analysis, pattern recognition, and data mining. To address clustering problems, mixture models have emerged as popular generative probabilistic models. They assume the existence of a mixture of distributions over the observation space, where each cluster is associated with a latent component distribution. The flexibility of mixture models allows researchers and practitioners to analyze and interpret data effectively.

Dirichlet Process Mixture Models (DPMMs) represent an extension of mixture models into Bayesian non-parametric models. DPMMs assume a mixture model with an infinite number of latent components and make a prior distribution over the model parameters. In literature, two common processes are used to represent the Dirichlet Process (DP). The first is the Chinese Restaurant Process (CRP) [1], which induces a distribution over the space of partitions. The second representation is known as the Stick-Breaking (SB) process [25], which provides a construction of the Dirichlet Process distribution. In [21], the authors proved that each of these representations could be derived from the other.

The Gibbs Sampling algorithm [24] is a famous MCMC method that performs the inference of DPMM's parameters. This algorithm iteratively updates the membership vector and the parameters associated with each cluster. When the prior conjugacy assumption holds, it becomes possible to integrate out the parameters. Consequently, the inference process

only requires sampling the memberships, leading to a variant known as collapsed Gibbs sampling.

While the DPMM has the advantage of automatically estimating the number of clusters and discovering new structures, its inference process becomes prohibitively slow and non-scalable when the number of observations is large. This poses a significant inconvenience for handling massive datasets and limits the applicability of the DPMM in real-world scenarios.

Distributed computing consists in distributing data across multiple workers; the workers could be components in the edge network, cores in a cluster, computers in a network, processors in a computer, etc. This distribution enables them to execute parallel computations independently. It presents an opportunity to accelerate computations and circumvent memory limitations, making it particularly suitable for handling large datasets.

Federated Learning (FL), a pioneering concept introduced by [14], revolutionized large-scale distributed learning by harnessing the machine intelligence residing in local devices. Federated learning is used to train machine learning models directly on edge devices or computing nodes. In the following, we will refer to a component of the edge network or a network node as a "worker". One of the defining characteristics of FL is that the data distributions may possess very different properties across the workers. Hence, any potential FL clustering method is explicitly required to be able to work under the heterogeneous data setting. Therefore, the availability of DPMM models tailored to these distributions becomes an indispensable requirement in our case. However, due to its highly decentralized nature, FL poses numerous statistical and computational challenges. In addition, the complexity of the clustering problem increases significantly when applied in a federated context. Furthermore, in many real-world applications, data is actually stored at the edge, spanning nodes, phones, and computers.

In this paper, we address a specific DPMM problem that arises in federated learning, contributing to the advancement of this exciting field. We propose a new distributed MCMC-based approach for DPMM inference (DisCGS) using sufficient statistics. It is important to emphasize that our approach focuses on a specific inference algorithm for the DPMM, the Collapsed Gibbs Sampler proposed in [24]. The scalability of this algorithm has not been addressed yet.

We highlight our key contributions as follows: **(1)** In the federated context, we have developed a new formulation of distributed MCMC inference of the DPMM using the Master/Worker architecture. The data is evenly distributed among the workers, ensuring a balanced workload. The workers operate independently and do not share information with each other. Instead, they communicate solely with the master, exchanging minimal information. **(2)** Each worker executes a local collapsed Gibbs sampler, which enables the discovery of local clusters and the inference of a local DPMM. Then, the sufficient statistics associated with each local cluster are sent to the master. **(3)** At the master level, a global collapsed Gibbs sampler is performed using only sufficient statistics. This allows the synchronization and estimation of the global DPMM and global clustering structure without accessing the content of each cluster. Theoretical background and computational details are provided. The general workflow of our model is illustrated in Figure 1. **(4)** We conduct several experiments to validate our method's merits and scalability with the number of nodes on large datasets. For a dataset of 100K data points, the centralized collapsed Gibbs sampler takes 12 hours to perform 100 iterations, whereas our approach does in just 3 minutes. **(5)** Distributing the collapsed Gibbs sampler for DPMMs is a crucial intermediate step towards the distribution of Bayesian non-parametric coclustering [18] and Multiple coclustering [10].

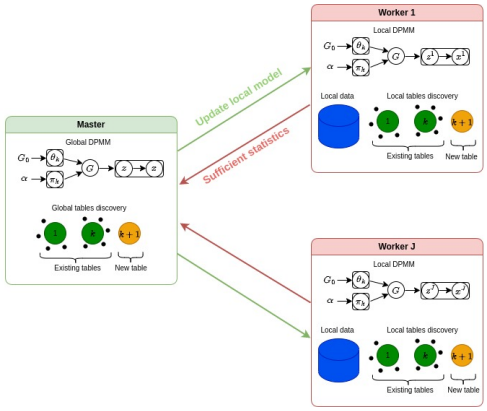

Figure 1: DisCGS workflow. Workers have only access to their own local data. Each worker infers a local DPMM and discovers local tables (clusters). Sufficient statistics are sent from each worker to the master, which infers the global DPMM and estimates the global partition. The global model is shared to each worker.

2 Related Work

Clustering problems in federated learning have drawn attention in recent studies. For instance, [15] introduced an approach for federated K-Means clustering, [27] proposed a Federated Fuzzy c-Means algorithm. In [13], the authors presented a federated clustering method using probabilistic models.

Several parallelized and distributed approaches have been proposed to address the slow inference of

DPMMs. [17] introduced a reparameterization of the Dirichlet Process that facilitates the learning of clusters representing the data and super-clusters defining the granularity of parallelization. [31] 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, [7] demonstrated that these approaches are impractical due to extremely unbalanced data distributions. Another parallelized MCMC sampler was proposed by [3], which combines a non-ergodic, restricted Gibbs sampler in the workers with split/merge proposals [12] at the master level to ensure ergodicity. However, this method is implemented only for a single machine. This approach is extended in [5] to a distributed version across multiple machines using a distributed-memory model.

In [8], the authors proposed a distributed inference algorithm for DPMMs based on a slice sampler developed by [29]. They addressed the scalability by developing a distributed architecture that estimates the Dirichlet Process and Hierarchical Dirichlet Process mixture models. [30] 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 [19], the authors introduced another distributed Markov Chain Monte Carlo (MCMC) 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 emphasizes known variances within and between clusters due to the prior, which limits the method's applicability.

The concept of using sufficient statistics is well-known and has been widely used to distribute the DPMM inference in many works, including [5,8,20,30]. However, it has not been applied to distribute the Collapsed Gibbs Sampler, which constitutes the main focus of this paper. Moreover, none of the existing methods is exploitable to enhance the scalability and the distribution of Bayesian non-parametric coclustering [18] and Multiple coclustering [10]. Moreover, to the best of our knowledge, no prior work has been proposed on federated clustering using Dirichlet Process models.

3 Background

Model definition

Let n and d be two positive integers, $\mathbf{x} = (x_1, \cdots, x_n)^T \in \mathbb{R}^{n \times d}$ the observed data, where $(\cdot)^T$ denotes the transpose operator. Let $\mathbf{z} = (z_1, \cdots, z_n)$ be the membership vector, where z_i is a latent variable such that $z_i = k$ means that the observation x_i belongs to the cluster k. The DPMM assumes that the observed data are generated according to the following model:

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

Under this model assumption, an observation x_i is generated by first drawing z_i from the Multinomial distribution parameterized by the vector of weights $\pi = (\pi_k)_{k=1}^{\infty}$ (also called the mixture proportions), and then x_i is sampled from $f(x_i, \theta_{z_i})$, where $f(\cdot, \theta_k)$ is the parameterized component distribution associated to the cluster k. The components parameters θ_k follow a prior distribution G_0 (also called the base distribution); in the multivariate Gaussian case, we have $\theta_k = (\mu_k, \Sigma_k)$. We let $\Theta = \{\theta_k,\, k > 1\}$ be the set of components parameters. The mixture proportions follow the Stick-Breaking process [25] parameterized by a concentration parameter $\alpha > 0$. We recall that the Stick-Breaking process is defined as follows:

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

We have $\pi_k > 0$ for every $k > 1$ and $\sum_{k=1}^{\infty} \pi_k = 1$. It is proved by [25] that the distribution $G = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}$ (where θ_k are sampled i.i.d according to G_0 and δ is the indicator function) follows the Dirichlet Process with a concentration parameter α and a base distribution G_0, noted $DP(\alpha, G_0)$. In the following, we denote $\Omega = (\alpha, G_0)$ the hyper-parameter set.

Inference.

The Gibbs Sampling algorithm [24] is a popular algorithm used for the DPMM inference based on Monte-Carlo sampling. It alternates between updating the membership vector and updating the parameters associated with each cluster. In the membership update step, the Gibbs sampling simulates the posterior distribution $\mathrm{p}(\mathbf{z} \mid \mathbf{x}, \Theta, \Omega)$ by sampling each z_i from the conditional distribution $\mathrm{p}(z_i \mid \mathbf{x}, \mathbf{z}_{-i}, \Theta, \Omega)$, where $\mathbf{z}_{-i} = \{z_l,\ l \neq i\}$ is the set of the remaining memberships. In the parameter update step, the component

parameter of each cluster k is updated by sampling according to the posterior distribution $\mathrm{p}(\theta_k|\mathbf{x}_k, G_0)$, with $\mathbf{x}_k = \{x_i,\ z_i = k\}$ the set of observations that belong to cluster k. The computing of such distributions is analytically tractable when G_0 is the conjugate prior to the density f.

The collapsed Gibbs sampler, which corresponds to the third algorithm proposed in [24] skips the parameter sampling step because under the conjugacy assumption, it is possible to integrate out the parameters θ_k which allows to directly compute the predictive (prior and posterior) without the parameter's values. Each membership z_i is sampled according to $\mathrm{p}(z_i \mid \mathbf{z}_{-i}, \mathbf{x}, \Omega) \propto$:

$$\begin{cases} n_k \mathrm{p}(x_i \mid z_i = k, \mathbf{x}_k, G_0), & \text{existing cluster } k, \quad (1) \\ \alpha \mathrm{p}(x_i \mid \Omega), & \text{new cluster}, \quad (2) \end{cases}$$

where n_k is the cardinal of cluster k. The posterior predictive $\mathrm{p}(x_i \mid z_i = k, \mathbf{x}_k, G_0)$ and prior predictive $\mathrm{p}(x_i \mid \Omega)$ can be obtained by integrating over θ.

In the multivariate Gaussian case with unknown mean and variance, we choose the Normal Inverse Wishart [9] (NIW) to be the conjugate prior G_0, with hyper-parameters $\lambda_0 = (\mu_0, \kappa_0, \Psi_0, \nu_0)$, where $\mu_0 \in \mathbb{R}^d$ is the prior mean, $\kappa_0 \in \mathbb{R}$ is the number of prior measurements, ν_0 is the degrees of freedom and $\Psi_0 \in \mathbb{R}^{d\times d}$ is the precision matrix. In this case, the probabilities in equations 1 and 2 can be computed analytically [22].

To summarize, the collapsed Gibbs sampler is an efficient MCMC method because it avoids component parameter sampling. This technique, known as Rao-Blackwellization, is due to the Rao-Blackwell theorem [2]. This theorem ensures that the estimator's variance obtained by integrating out θ is always lower than or equal to the one of a direct estimator. This theorem remains true in collapsed Gibbs sampling [16].

4 The proposed method

The collapsed version of the Gibbs sampling inference inspires our approach. The main objective of DisCGS is to make the inference scalable while keeping the MCMC's precision and the DPMM's flexibility. Our approach consists in distributing the data evenly over the workers and alternating at each iteration between two steps: Worker level and Master level. Below, we provide a theoretical detailed of the collapsed Gibbs samplers executed at each level. The workflow of our approach is described in figure 1

4.1 DisCGS at worker level

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

In this level, the local memberships are updated one by one using the collapsed version of Gibbs sampling detailed in section 3. Each z_i^j is updated by sampling from $\mathrm{p}(z_i^j \mid \mathbf{z}_{-i}^j, \mathbf{x}^j, \Omega) \propto$

$$\begin{cases} n_k^j \mathrm{p}(x_i^j \mid z_i^j = k, \mathbf{x}_k^j, G_0), & \text{existing cluster } k \quad (3) \\ \alpha \mathrm{p}(x_i^j \mid \Omega), & \text{new cluster}, \quad (4) \end{cases}$$

where n_k^j is the size of cluster k in worker j and $\mathbf{x}_k^j$ is the contents of cluster k in worker j. The posterior and prior predictive distributions are computed analytically using the same global prior G_0, as detailed in [22]. After updating the local membership vector, we compute the sufficient statistics [26] associated with each cluster. In the multivariate Gaussian case, the sufficient statistics (T_k^j, S_k^j) for a cluster $\mathbf{x}_k^j$ are given by [9]:

$$T_k^j = \frac{1}{n_k^j} \sum_{x \in \mathbf{x}_k^j} x \in \mathbb{R}^d, \quad (5)$$

$$S_k^j = \sum_{x \in \mathbf{x}_k^j} (x - T_k^j)(x - T_k^j)^T \in \mathbb{R}^{d\times d}. \quad (6)$$

Finally, sufficient statistics and the sizes of each cluster are sent to the master. The DisCGS inference process at the worker level is described in Algorithm 1.

Algorithm 1 DisCGS inference at worker level

1: **Input**: Dataset $\mathbf{x}^j$, concentration parameter α and prior G_0.
2: **For** $i \leftarrow 1$ **to** n^j **do**:
3: Remove x_i^j from its local cluster.
4: Compute $\mathrm{p}(z_i^j \mid \mathbf{z}_{-i}^j, \mathbf{x}^j, \Omega)$ as defined by eq. 3 and 4.
5: Sample z_i^j.
6: Add x_i^j to its new cluster.
7: **For** $k \leftarrow 1$ **to** K **do**:
8: Compute the sufficient statistics (T_k^j, S_k^j) using eq. 5 and 6.
9: **Output**: Sufficient statistics $\{(T_1^j, S_1^j) \cdots, (T_K^j, S_K^j)\}$ and cluster sizes $(n_1^j, \cdots, n_K^j)$.

4.2 DisCGS at 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 hyper-parameters 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 observations assigned to the same global cluster will share the same label. We sample the global membership z_h^j of the cluster $\mathbf{x}_h^j$ (the local cluster h of worker j) according to $\mathrm{p}(z_h^j \mid \mathbf{z}_{-h}^{-j}, \mathbf{x}, \Omega) \propto$

$$\begin{cases} n_k \mathrm{p}(\mathbf{x}_h^j \mid z_h^j = k, \mathbf{x}_k, G_0), & \text{existing cluster k} \quad (7) \\ \alpha \mathrm{p}(\mathbf{x}_h^j \mid G_0), & \text{new cluster,} \quad (8) \end{cases}$$

In practice, the joint posterior predictive and the joint prior predictive distributions (equations 7 en 8 respectively) are computed analytically by only using sufficient statistics, i.e. without having access to the content of cluster $\mathbf{x}_h^j$. In fact, we have:

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

where $|\cdot|$ is the determinant, and the hyper-parameter values $(\mu_h^j, \kappa_h^j, \Psi_h^j, \nu_h^j)$ are obtained as follows:

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

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

where T_h^j and S_h^j are the sufficient statistics obtained from the workers. Moreover, we have $\mathrm{p}(\mathbf{x}_h^j \mid z_h^j = k, \mathbf{x}_k, G_0) =$

$$\pi^{\frac{-dn_h^j}{2}} \cdot \frac{\kappa_k^{d/2}}{\left(\kappa_h^j\right)^{d/2}} \cdot \frac{\Gamma_d\left(\nu_h^j/2\right)}{\Gamma_d\left(\nu_k/2\right)} \cdot \frac{|\Psi_k|^{\nu_k/2}}{\left|\Psi_h^j\right|^{\nu_h^j/2}}$$

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:

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

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

The inference process at the master level is described in Algorithm 2.

Algorithm 2 DisCGS inference at master level

1: **Input**: Sufficient statistics, cluster sizes, α and prior G_0.
2: **For** each (j, h) **do**:
3: Remove $\mathbf{x}_h^j$ from its global cluster.
4: Compute $\mathrm{p}(z_h^j \mid \mathbf{z}_{-i}^{-h}, \mathbf{x}_h^j, \Omega) \propto$ as defined by eq. 7 and 8.
5: Sample z_h^j.
6: Add $\mathbf{x}_h^j$ to its new global cluster.
7: Update the membership vector $\mathbf{z}$.
8: **Output**: Membership vector $\mathbf{z}$.

4.3 Collapsed Gibbs sampler in federated learning

In federated learning, the observations are distributed on different workers "components". This corresponds to the horizontal decomposition of the dataset. In this context, each worker is initialized with the same global model, which is identical to the model present on the server. Then, each worker updates its model using its private data through the collapsed Gibbs sampler detailed in section 4.1; this step allows the estimation of the local clusters and the local model. Then, the sufficient statistics and cluster sizes associated with each local cluster are computed and transmitted to the master "server". The master proceeds to update the global model and to estimate the global clustering structure without having access to the data. This process is achieved using the collapsed Gibbs sampler detailed in section 4.2. These updates are then shared with each worker, which allows them to update the local model with the global model. This iterative process continues alternating between the worker and master steps until the global model is fully estimated.

5 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 our distributed algorithm DisCGS and the centralized CGS (Collapsed Gibbs Sampler from [24]) on synthetic datasets of different sizes. Lastly, we investigate the scalability of DisCGS by increasing the number of nodes while keeping the number of observations fixed. For this purpose, we execute our distributed algorithm on 10^6 data points multiple times, varying the number of cores.

5.1 Implementation settings and distributed environment

In the following experiments, we use an uninformative prior NIW for both CGS and DisCGS algorithms. Therefore, both methods are implemented by setting 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, 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.

5.2 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] [8] and SubC[3] [5], 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 DisCGS. It is important to mention that SubC and M-R are the only approaches of distributed inference for DPMM for which open-source working code is available.

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 1. Due to the smaller size of the EngyTime dataset, we distributed the data only on two workers. The image datasets are encoded using a variational auto-encoder, and each image is encoded into an 8-dimensional vector.

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

[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

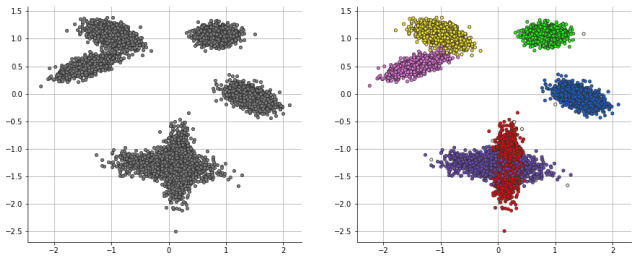

Figure 2: Unlabeled (left) and labeled data (right), after 100 iterations of DisCGS on a synthetic dataset with overlapped clusters, ARI = 0.85, NMI = 0.89, and ACC = 0.91. The number of estimated clusters is 7.

To evaluate the clustering performance, we compute the three clustering metrics: Adjusted Rand Index (ARI) [11], Normalized Mutual Information (NMI) [28], and clustering accuracy (ACC) [33].

Table 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 proposed method (DisCGS) outperforms other methods or achieved the second-best score in almost all the datasets for the three clustering metrics. 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.

Figure 2 illustrates the estimated clusters obtained by our distributed approach on the synthetic 10K dataset. This figure represents the best partition achieved over the ten trials. The purpose of using this dataset is to evaluate DisCGS's performance when confronted with complex datasets that exhibit overlapping clusters. The results demonstrate that our method performs remarkably well even when dealing with such datasets. DisCGS achieved an ARI, NMI, and ACC score of 0.85, 0.89, and 0.91, respectively. It estimated seven clusters, while the number of clusters is six.

5.3 Convergence

In this experiment, we examine the convergence of both likelihood and ARI score of three methods: DisCGS, CGS (Collapsed Gibbs Sampler), and SubC. These evaluations are performed on three datasets: Synthetic 100K, Fashion-mnist, and Balanced. It's important to note that only CGS and SubC share the same model assumption as DisCGS, resulting in a comparable likelihood.

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, Lecun et al.
Fashion mnist	70000	8	10	Zalando's article images [32].
Letter	103600	8	26	Handwritten letters [4].
Balanced	131600	8	47	Both of handwritten digits and letters [4].
Digits	280000	8	10	Handwritten digits [4].
UrbanGB	360177	3	469	Coordinates (longitude and latitude) of road accidents [6].

Table 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.

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

Table 2: The mean and the standard deviation of the three clustering metrics, ARI, NMI, and ACC, over 10 runs on different datasets. The best result within each row is marked as bold, and the runner-up is underlined.

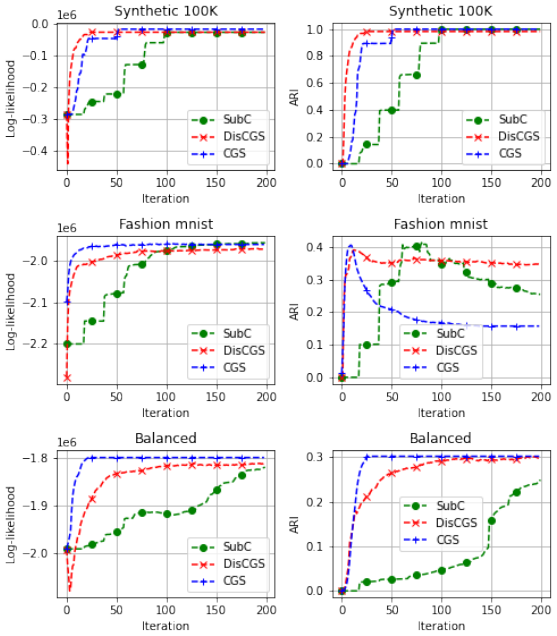

Figure 3: Log-likelihood and ARI score every iteration

Figure 3 illustrates the evolution of the log-likelihood and ARI score at each iteration. We observe that our algorithm converges almost at a similar 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 [30]. 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.

5.4 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 (DisCGS) and the centralized collapsed Gibbs sampler (CGS) from [24], 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 4 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

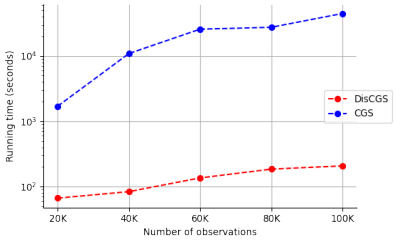

Figure 4: Running time (in logarithmic scale) for 100 iterations of DisCGS and CGS inference for DPMM on synthetic datasets of different sizes.

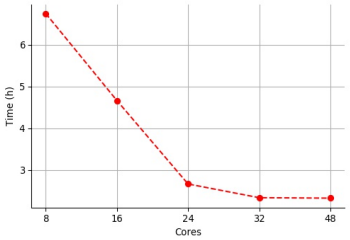

Figure 5: Running time (hours) as a function of the number of cores of DisCGS on 10^6 data points.

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 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 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 3: Clustering metrics ARI, NMI, and ACC obtained by the distributed (Dis.) and centralized (Cen.) inference on synthetic datasets of different sizes.

5.5 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 5 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 workers. Figure 6 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 [23].

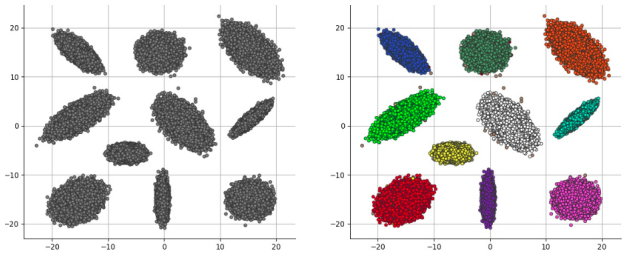

Figure 6: Unlabeled data (left) and labeled data (right), after $M = 100$ iterations of DisCGS on 10^6 data points, using 32 cores, ARI = 0.98, NMI = 0.98, and ACC = 0.98. The number of inferred clusters is 14.

6 Conclusion and perspectives

This article presents a novel distributed MCMC inference method, called DisCGS, for DPMMs. DPMMs are highly useful for clustering problems, especially when the number of clusters is unknown. However, the inference process of DPMMs tends to become significantly slow as the dataset size increases. To overcome this limitation, our proposed DisCGS is specifically designed to handle distributed data across independent and heterogeneous machines, making it suitable for horizontal federated learning scenarios, i.e., when the workers are the different components of the edge network. The experimental results showed highly promising outcomes. The proposed method significantly reduces the inference time while maintaining accurate

results. Our ongoing research exploits the DisCGS approach to massively distribute the Non-Parametric Latent Block Model (NPLBM) [18] and the multiple Coclustering model.

7 Acknowledgments

This work has been supported by the Paris Île-de-France Région in the framework of DIM AI4IDF.

References

[1] C. E. Antoniak. Mixtures of Dirichlet Processes with Applications to Bayesian Nonparametric Problems. *The Annals of Statistics*, 2(6):1152 – 1174, 1974.

[2] D. Blackwell. Conditional Expectation and Unbiased Sequential Estimation. *The Annals of Mathematical Statistics*, 18(1):105 – 110, 1947.

[3] J. Chang and J. W. Fisher III. Parallel sampling of dp mixture models using sub-cluster splits. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[4] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. Emnist: Extending mnist to handwritten letters. In *International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926, 2017.

[5] O. Dinari, A. Yu, O. Freifeld, and J. W. Fisher III. Distributed mcmc inference in dirichlet process mixture models using julia. In *19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 518–525, 2019.

[6] D. Dua and C. Graff. UCI machine learning repository.

[7] Y. Gal and Z. Ghahramani. Pitfalls in the use of parallel inference for the dirichlet process. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 208–216, Bejing, China, 22–24 Jun 2014. PMLR.

[8] H. Ge, Y. Chen, M. Wan, and Z. Ghahramani. Distributed inference for dirichlet process mixture models. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2276–2284, Lille, France, 07–09 Jul 2015.

[9] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian data analysis*. 1995.

[10] E. Goffinet, M. Lebbah, G. Azzag, H., L., and A. Coutant. Multivariate time series multicoclustering. application to advanced driving assistance system validation. *29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. ESANN.*, 2021.

[11] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[12] S. Jain and R. 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.

[13] C. Jichan, K. Lee, and R. Kannan. Federated unsupervised clustering with generative models. *In AAAI 2022 International Workshop on Trustable, Verifiable and Auditable Federated Learning*, 2022.

[14] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence, 2016.

[15] H. H. Kumar, K. V R, and M. K. Nair. Federated k-means clustering: A novel edge ai based approach for privacy preservation. In *IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 52–56, 2020.

[16] J. S. Liu, W. H. Wong, and A. Kong. Covariance structure of the gibbs sampler with applications to the comparisons of estimators and augmentation schemes. *Biometrika*, 81(1):27–40, 1994.

[17] D. Lovell, J. Malmaud, R. Adams, and V. Mansinghka. Clustercluster: Parallel markov chain monte carlo for dirichlet process mixtures. In *In Workshop on Big Learning, NIPS, 2012*.

[18] E. Meeds, S. Roweis, E. Meeds, and S. Roweis. Nonparametric bayesian biclustering, 2007.

[19] K. Meguelati, B. Fontez, N. Hilgert, and F. Masseglia. Dirichlet process mixture models made scalable and effective by means of massive distribution. pages 502–509, 04 2019.

[20] K. Meguelati, B. Fontez, N. Hilgert, and F. Masseglia. Dirichlet Process Mixture Models made Scalable and Effective by means of Massive Distribution. In *SAC 2019 - 34th ACM/SIGAPP Symposium on Applied Computing*, pages 502–509, Limassol, Cyprus, Apr. 2019. ACM/SIGAPP.

[21] J. W. Miller. An elementary derivation of the chinese restaurant process from sethuraman's stick-breaking process, 2018.

[22] K. P. Murphy. Conjugate bayesian analysis of the gaussian distribution. *def*, 1(2σ2):16, 2007.

[23] K. P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013.

[24] R. M. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[25] J. Sethuraman. A constructive definition of dirichlet priors. *Statistica Sinica*, 4(2):639–650, 1994.

[26] S. Silvey. *Statistical Inference*. CRC Press, 2017.

[27] M. Stallmann and A. Wilbik. Towards federated clustering: A federated fuzzy c-means algorithm (FFCM). *CoRR*, abs/2201.07316, 2022.

[28] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 01 2002.

[29] S. G. Walker. Sampling the dirichlet mixture model with slices. *Communications in Statistics - Simulation and Computation*, 36(1):45–54, 2007.

[30] R. Wang and D. Lin. Scalable estimation of dirichlet process mixture models on distributed data. In *International Joint Conference on Artificial Intelligence*, 2017.

[31] S. Williamson, A. Dubey, and E. Xing. Parallel Markov chain Monte Carlo for nonparametric mixture models. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 98–106, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[32] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[33] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing*, 19(10):2761–2773, 2010.