POLITECNICO MILANO 1863

Repulsive Mixture Models with the Wasserstein Distance

Project Report

Team

Manuel Bressan, Leonardo Perelli, Federico Ravanetti,
Sebastiano Rossi, Edward Wiels, Daniela Zanotti

Tutor

Dr.Mario Beraha

M.Sc. MATHEMATICAL ENGINEERING
BAYESIAN STATISTICS
A.A. 2021-2022

February 14, 2022

Contents

1 Introduction

Mixture Model is a widespread probabilistic model that finds application in numerous different areas from density estimation, pattern identification, feature selection to clustering and many other more. Many times sets of independent and identically distributed observations cannot be described by a single distribution, but a combination of a small number of distributions belonging to the same parametric family is needed. In our report we will focus on the clustering problem. Bayesian clustering has been widely studied both from a model-based and non-parametric perspective, for example one can refer to Binder (1978) or Quintana & Iglesias (2003). By introducing a latent variable in the simple Gaussian Mixture Model we can identify the belonging of each point to a cluster.

Since a priori we do not know the exact (or the most probable) number of clusters that should be found, we have to overestimate the maximum number of clusters in our model ($\mathbf{H} = \mathbf{15}$). This approach works well if the different clusters are Gaussian distributed but if we pick more heterogeneous data we can immediately notice the fallback of this method: the mixture model tries to perfectly fit the data and for this reason we find many small clusters placed one near the other in place of a single, big cluster.

In this report we want to address this problem by introducing a penalization term for the joint prior of the component-specific parameters, i.e. mean $\boldsymbol{\mu}$ and shape $\boldsymbol{\Sigma}$ of each cluster. This penalization is based on the sum of Wasserstein distances among all the possible couple $(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)$. This metric is a sensible choice because it accounts for the similarity between two distribution in terms of location but also in terms of shape. In this way the Wasserstein distance can be large even if the distributions are centered in the same point because they can have different shape. This new model is then called Repulsive Mixture Model due to its behaviour. The use of models discouraging closeness between distributions has become more and more popular in the last decade, see for instance Petralia, Rao & Dunson (2012) or Xie & Xu (2017). In Beraha, Argiento, Møller & Guglielmi (2020) the prior of the 'cluster centres' is determined by a finite repulsive point process. Although the application of the Wasserstein distance in contexts such as Machine Learning or Optimal Transport is quite ordinary, its employment in Bayesian Statistics along with Mixture Models is reasonably recent, see for example Delon & Desolneux (2020).

Throughout this report, implemented models will be tested on two datasets, described in section 2. Section 3 shows the limits of conventional mixture models by implementing and testing a basic Gaussian Mixture Model. This model is then updated in section 4 to contain a Wasserstein-based penalization term. In this section we also explore and tackle many problems that arise with the simulation of the posterior distribution. Moreover we also implement some different kind of proposal that allow us to better explore the target space. Finally, to allow for more flexible models, with for instance heavier tails, Student-t based models can be used, rather than Gaussian ones. The description and implementation for such an approach are given in section 5.1.

Alongside of our effort to program this Repulsive Mixture Model we also pursued a parallel objective to implement all the models in a computationally efficient way, relying on the Python High Performance Computing library JAX. Thanks to the Just-in-Time code compilation feature, we are able to compile the code and obtain much faster computations. For example, before the optimization a typical run of 4000 total iterations would take 60 minutes. After optimizing the code, this takes just 20 seconds, yielding a speed up of around 180 times. This was not always trivial to implement and required modifications to the original code structure, but was fundamental to be able to perform several runs of the algorithm and detect potential problems, as well as finding a good set of hyperparameters.

2 Test Datasets

To evaluate and analyse the performance of the models we develop, we decide to fix two datasets to use as benchmark. One dataset is the so called 'Old Faithful Geyser' dataset. It consists of the waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA. It presents three major and well defined clusters and a smaller one.
The other test dataset is generated fictitiously, through perturbation of Normal Distributions. The starting point is a 4-component Normal Distribution with equal weights. The distributions are equally weighted and all have covariance matrix equal to the identity matrix.
The means of the components are: $[0, 0], [0, 4], [4, 0], [4, 4]$.
The two test datasets are shown in the scatterplot below :

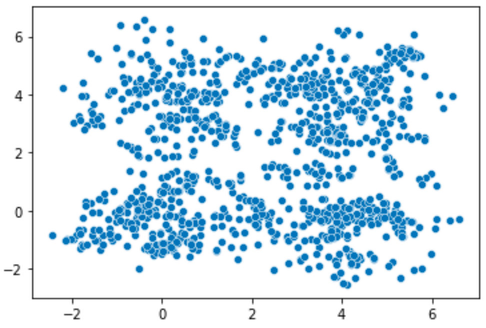


Figure 1: Perturbed Normals Dataset

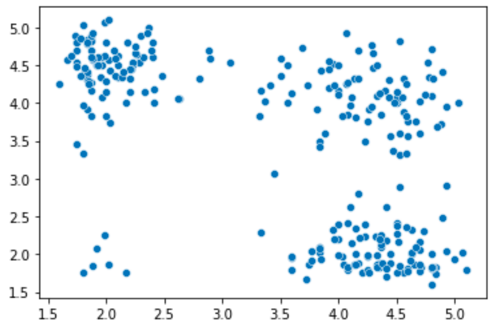


Figure 2: Faithful Dataset

To obtain the perturbed dataset, we fix a number of perturbated components, in our case 500. Each component is centered randomly on one of the original four means, with covariance given by $0.04I$. Finally, we sample 1000 points from the obtained mixture of 500 components. The dataset obtained is very challenging for a basic Gaussian Mixture Model, as will be shown in the following sections. Indeed, even if the perturbation may not seem so evident, many small clusters are picked up by the basic model, all very close one to another. A comparison between the non perturbed dataset and the corresponding perturbed dataset shown before can be seen in the following figure:

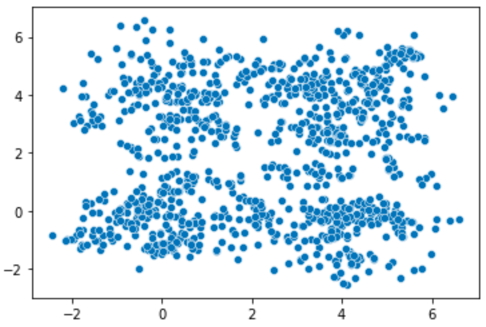


Figure 3: Perturbed Normals Dataset

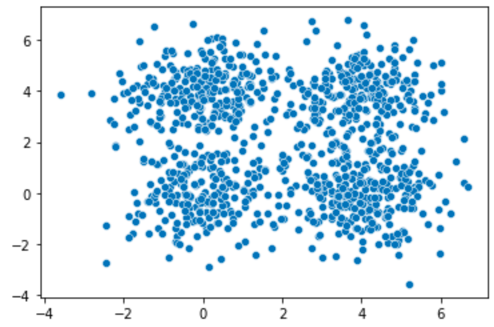


Figure 4: Non-Perturbed Normals Dataset

In subsections 3.4 and 3.5 we test the basic Gaussian Mixture model on such datasets. This is done mainly with the aim of highlighting the criticalities arising with the basic Gaussian Mixture model presented in subsection 3.1, i.e. when we do not take into account any kind of repulsiveness within the prior assumptions.
In subsections 4.7 and 4.8 we test the Repulsive model with Gaussian components, while in subsections 5.3 and 5.4 we analyze the results concerning the Student-t penalized model.

3 Standard Gaussian Mixture Models

3.1 Model

We start dealing with basic Gaussian Mixture Models parametrized with the Normal Inverse Wishart. The model is the following:

$$\begin{gathered} Y_1, ..., Y_n | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{w} \overset{iid}{\sim} \sum_{h=1}^{H} w_h \mathcal{N}(\boldsymbol{\mu}_h, \boldsymbol{\Sigma_h}) \\ (\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h) \overset{iid}{\sim} NIW(\boldsymbol{\mu}_0, \lambda_0, \boldsymbol{\Phi}_0, \nu_0) \\ \boldsymbol{w} \sim \mathit{Dirichlet}(\alpha_1, \ldots, \alpha_H) \end{gathered} \tag{1}$$

Since the goal is to apply the Mixture Model in order to find clusters, we make this connection to clustering more explicit by introducing "cluster assignment" variables c_i, $i = 1, \ldots, n$, indicating at which cluster, and so at which component of the model, the observation i belongs. We express this as:

$$P(c_i = h \mid \mathbf{w}) = w_h$$

With the introduction of the new variables, the Mixture Model can be restated as the following hierarchical model:

$$\begin{gathered} y_i \mid c_i = h, \{f_h\}_{h=1}^{H} \overset{iid}{\sim} \mathcal{N}(\boldsymbol{\mu}_h, \boldsymbol{\Sigma_h}) \\ (\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h) \sim NIW(\boldsymbol{\mu}_0, \lambda_0, \boldsymbol{\Phi}_0, \nu_0) \\ \boldsymbol{w} \sim \mathit{Dirichlet}(\alpha_1, \ldots, \alpha_H) \\ P(c_i = h \mid \mathbf{w}) = w_h \end{gathered} \tag{2}$$

The hyperparameters are set as follows:

- $\alpha_1, \ldots, \alpha_H$ are set to 0.1
- $\boldsymbol{\mu}_0$ is set to the mean of the data
- λ_0 is set to 0.1
- $\boldsymbol{\Phi}_0$ is set to $\boldsymbol{I}$
- ν_0 is set to 2

Moreover, we set $\mathbf{H = 15}$ as an upper bound for the number of components. This is justified by the fact that the datasets considered have a low true number of components, and so we do not want the algorithm to generate too many clusters.
The parameters of the Dirichlet distribution are set at the same value because there is no prior knowledge favoring one component over another, leading to a symmetric distribution in which α is defined as the concentration parameter. When the latter value is smaller than 1, sparse distributions are preferred, while when it's above 1, dense distributions are more likely.
As a consequence, we set α equal to 0.1, in order to have the vast majority of the points concentrated in a few components, obtaining most of the weights of the Mixture Model close to zero and ending up with a small number of clusters.

In order to sample from the posterior distributions we use Gibbs sampling, computing the full conditionals of the model:

$$\boldsymbol{\Sigma}|\boldsymbol{y},\boldsymbol{\mu},\boldsymbol{c},\boldsymbol{w} \sim IW(\nu_n,\boldsymbol{\Phi}_n)$$

$$\boldsymbol{\mu}_h|\boldsymbol{y},\boldsymbol{\Sigma},\boldsymbol{c},\boldsymbol{w} \sim N_N(\boldsymbol{\mu}_n,\frac{\boldsymbol{\Sigma}}{(n+\lambda)})$$

$$\boldsymbol{w}|\boldsymbol{y},\boldsymbol{\mu},\boldsymbol{c},\boldsymbol{\Sigma} \sim Dirichlet(\alpha_1+n_1,\dots,\alpha_H+n_H)$$

$$c_i|\boldsymbol{\mu},\boldsymbol{\Sigma},\boldsymbol{w},\boldsymbol{y} \sim Cat(\frac{w_1 f(y_i|\boldsymbol{\mu_1},\boldsymbol{\Sigma_1})}{\sum_{h=1}^H w_h f(y_i|\boldsymbol{\mu_h},\boldsymbol{\Sigma_h})},\dots,\frac{w_H f(y_i|\boldsymbol{\mu_H},\boldsymbol{\Sigma_H})}{\sum_{h=1}^H w_h f(y_i|\boldsymbol{\mu_h},\boldsymbol{\Sigma_h})}) \tag{3}$$

$$\boldsymbol{\mu}_n = \frac{\lambda_0\boldsymbol{\mu}_0 + n\overline{\boldsymbol{y}}_h}{\lambda_0+n}$$

$$\lambda_n = \lambda_0 + n$$

$$\nu_n = \nu_0 + n$$

$$\boldsymbol{\Phi}_n = \boldsymbol{\Phi}_0 + \sum_{i=1}^{n}(\boldsymbol{y}_i-\overline{\boldsymbol{y}}_h)(\boldsymbol{y}_i-\overline{\boldsymbol{y}}_h)^T + \frac{\lambda_0 n}{\lambda_0+n}(\overline{\boldsymbol{y}}_h-\boldsymbol{\mu}_0)(\overline{\boldsymbol{y}}_h-\boldsymbol{\mu}_0)^T$$

3.2 Discussion

Mixture models have troubles trying to find the true clusters in a given dataset. It will be shown in the next sections that when we sample data from a Gaussian Mixture Model the algorithm seems to perform quite well in identifying the clusters, but as soon as we perturb a bit our initial distribution, the algorithm applied on the data sampled from the new distribution is not as precise. Specifically, an excessive number of clusters is identified when compared to the true one.
This is mainly due to the fact that the Mixture Model will contain overlapping components, regardless of the prior chosen for the component-specific parameters. Indeed, the posterior distribution will be highly influenced by the choice of the common prior for all the components and will struggle distinguishing between different clusters, resulting in substantial uncertainty on the true number of components that generated the observations.

To overcome this issue we shall take advantage of Repulsive Models, which are able to constrain the clustered distributions to stay away from each other: a natural consequence will be to obtain less, but better separated clusters.
The idea of discouraging closeness among component-specific parameters through repulsive priors has been widely studied, especially in combination with Gaussian Mixture models: for instance, in Petralia, Rao & Dunson (2012) two types of priors are considered, one penalizing redundant kernels having close to identical locations and scales, the other deflating closeness in terms of location only. In a similar way, Xie & Xu (2017) force the repulsive property only concerning the mean vectors of the cluster parameters, which is, they allow for non-vanishing densities even when distinct components share an identical covariance matrix.
In order to generate a Repulsive Model, we will add to the prior of the centers a penalization term built to be proportional to the Wasserstein distance, which takes both location and scale into consideration.

3.3 Binder Loss

Model-based clustering methods such as finite Mixture Models commonly tackle the problem of determining the number of clusters and the component probability distribution by comparing different information criteria. However, a typical problem is the one of choosing appropriate summaries of the posterior and therefore being able to characterize the uncertainty around the resulting clustering structure.
As we have already pointed out, many current clustering procedures produce a large number of partitions differing only in a few data points. This issue will be in part overcome by the repulsiveness induced in the prior of the model, but on the other hand the choice of an appropriate summary tool is crucial.
Indeed, as we have seen in subsection 3.1, the partition of items into subsets becomes a parameter of the model for the data through the clustering assignment variables c_i's, hence they are subject to prior assumptions so that inference about the clustering derives from properties of the corresponding posterior distribution. As a consequence, exploiting default choices as point estimators for the clustering structure parameter, such as the posterior expected value, can lead to misleading results, due for example to label-switching.

For this reason, we decide to adopt a different approach and solve this issue by selecting an adequate loss function $\mathcal{L}(\mathbf{c}, \hat{\mathbf{c}})$ over clustering parameters $\mathbf{c}$ and $\hat{\mathbf{c}}$, where the first parameter vector denotes the true clustering structure while the second denotes, for example, the output of an MCMC algorithm.
In this way, our optimal point estimator will be the one minimizing the posterior expectation of the loss function, say:

$$\mathbf{c}^* = \underset{\hat{\mathbf{c}}}{\operatorname{argmin}} \quad E\big[\mathcal{L}(\mathbf{c}, \hat{\mathbf{c}})|data\big] \tag{4}$$

Accordingly, we concentrate on loss functions defined on pairwise coincidences, meaning that we want to be able to measure the disagreements in all possible couples of observations between the true and estimated clusterings.
Binder's loss function, for instance, penalizes pairs of items that are assigned to different clusters when they should be in the same one, and vice-versa, so that the total loss is obtained by summing over all the pairs:

$$\mathcal{L}(\mathbf{c}, \hat{\mathbf{c}}) = \sum_{i,j} \Bigg(I_{c_i=c_j} I_{\hat{c}_i \neq \hat{c}_j} + I_{c_i \neq c_j} I_{\hat{c}_i = \hat{c}_j} \Bigg) \tag{5}$$

Then, the resulting posterior expected loss will be a linear function of posterior marginal coincidence probabilities, which is, for each pair, the posterior probability that they are clustered together:

$$E\big[\mathcal{L}(\mathbf{c}, \hat{\mathbf{c}})|data\big] = -2 \sum_{i,j} I_{\hat{c}_i=\hat{c}_j} \left(p_{i,j} - \frac{1}{2} \right) \tag{6}$$

where $p_{i,j} = P(c_i = c_j|data)$.

Consequently, minimizing the posterior expected loss is equivalent to find $\mathbf{c}^*$ such that:

$$\mathbf{c}^* = \underset{\hat{\mathbf{c}}}{\operatorname{argmax}} \quad \sum_{i,j} I_{\hat{c}_i=\hat{c}_j} \left(p_{i,j} - \frac{1}{2} \right) \tag{7}$$

The above maximization problem can be tackled in different ways, each of which lead to distinct complexity with respect to the number of observations n. For instance, in Lau & Green (2007) it is represented as a binary integer programming problem, having complexity $O(n^3)$. Other approaches are based on greedy algorithms, whose cost scales quadratically in the number of observations.
Instead, we attain the $\hat{\mathbf{c}}$'s directly as the output of our Markov Chain Monte Carlo algorithm: this approach is preferable in our setting as it has complexity $O(n^2)$, since we still have to compute the marginal coincidence probabilities $p_{i,j}$, but scales linearly in the number of MCMC iterations needed.

3.4 Dataset Test 1

We proceed by evaluating the behaviour of the basic Gaussian Mixture Model 2 on the fictitious dataset generated by perturbing the four-components Normal Distribution, as presented in section 2. As a reminder, let us notice that this test dataset will be quite challenging without assuming any kind of repulsiveness in the prior assumptions.
Indeed we expect the model to struggle separating the clusters and eventually ending up estimating a tangled cluster structure.
The best clustering structure $\mathbf{c}^*$ is chosen according to the result of the minimization procedure involving the posterior expected value of the Binder's loss function. In particular, at each iteration the output of the MCMC algorithm is exploited to compute the marginal coincidence probability $p_{i,j}$ and enters into the maximization problem (7) of subsection 3.3.
In particular, we show the number of clusters estimated during the iterations of one run of the MCMC algorithm, along with the best clustering structure: it is evident that the model ends up estimating an excessive number of components.

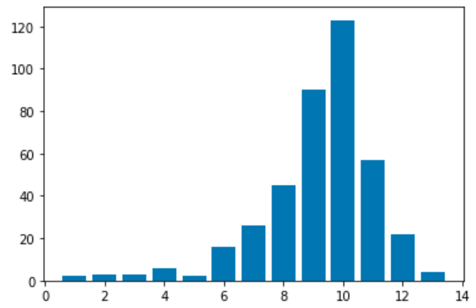


Figure 5: frequencies of estimated clusters during the iterations on the fictitious dataset

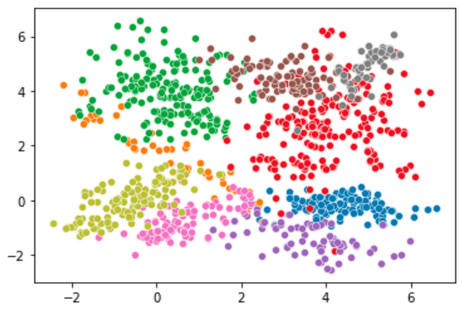


Figure 6: best clustering w.r.t. Binder's loss on the fictitious test dataset

3.5 Dataset Test 2

We now aim to evaluate the performance of the basic Mixture Model (2) presented in subsection 3.1 on the 'Old Faithful Geyser' dataset. As we have noticed before, there are three recognizable clusters of greater size and one small cluster. We therefore expect the basic model to well recognize such a clear-cut data structure.
We can immediately notice that the basic Mixture Model can identify correctly the four clusters:

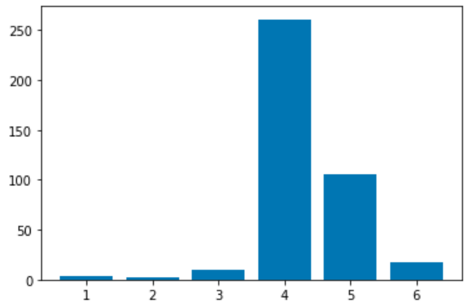


Figure 7: frequencies of estimated clusters during the iterations for Old Faithful Geyser

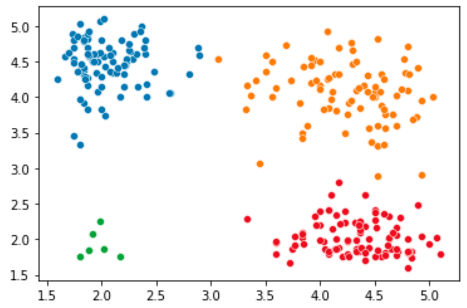


Figure 8: best clustering w.r.t. Binder's loss on the Old Faithful Geyser Dataset

4 Gaussian Mixture Models with the Wasserstein distance

4.1 The Wasserstein Distance

In order to penalize prior mixtures with components that are similar, it is necessary to have a metric that assesses 'similarity'. As it happens, Optimal Transport theory provides useful mathematical tools in order to compare probability distributions.
Indeed, given two probability distributions m_X and m_Y defined over some Euclidean space, say R^d, and given some positive cost function c over $R^d \times R^d$, we want to solve the following problem:

$$\inf_{X\sim m_X, Y\sim m_Y} E\Big(c(X,Y)\Big) \tag{8}$$

Notice that when the cost function has the form $c(x,y) = ||x-y||^p$ for $p \geq 1$, equation 8, once elevated to the power $\frac{1}{p}$, defines a metric between probability distributions having moment of order p, say the Wasserstein distance or order p, denoted by W_p.
Then, by denoting with $\Gamma(m_X, m_Y)$ the subset of probability distributions on $R^d \times R^d$ having marginal distribution m_X and m_Y, then we have that:

$$W_p^p := \inf_{X\sim m_X, Y\sim m_Y} E\Big(||X-Y||^p\Big) = \inf_{\gamma\in\Gamma(m_X,m_Y)} \int_{R^d\times R^d} ||x-y||^p \, d\gamma(x,y) \tag{9}$$

Typically, the Wasserstein metric is explained intuitively by considering two possible piles of earth. Starting with one pile, the goal is to transform it into the other pile with minimal work. The amount of work gets lower when the two piles are closer to each other and more similar in shape. Analogously, the Wasserstein distance is small when two distributions are placed close to each other and have a similar shape.

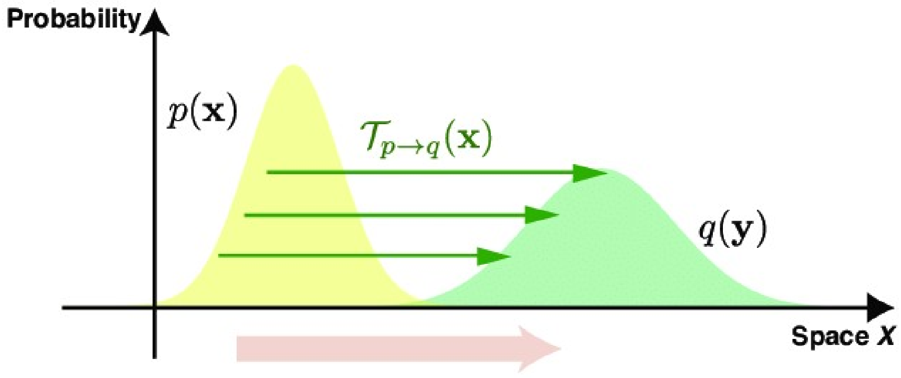


Figure 9: The Wasserstein metric can be seen as the amount of 'work' that is needed to transform one distribution in another.

Indeed, one can show that there always exists a couple (X, Y) of random variables attaining the minimum of (9), which is usually called the 'optimal coupling': the probability distribution γ corresponding to such a couple is called 'optimal transport plan' and is such that the Wasserstein distance W_p^p equals the total cost for distributing all the mass related to the distribution m_X onto that of m_Y with minimal cost.
We set $p = 2$ and therefore the Wasserstein distance becomes:

$$W_2^2(m_X, m_Y) := \inf_{\gamma\in\Gamma(m_X,m_Y)} \int_{R\times R} ||x-y||^2 \, d\gamma(x,y) \tag{10}$$

Notice that in general computing the optimal transport plan between two distributions is not an easy task. However, in some peculiar cases, for example the one involving two Gaussian distribution on R^d, we have a closed-form expression for W_2^2 which is given by:

$$W_2^2(m_X, m_Y) = ||\mu_X - \mu_Y||^2 + tr\Big(\Sigma_X + \Sigma_Y - 2({\Sigma_X}^{\frac{1}{2}}\Sigma_Y{\Sigma_X}^{\frac{1}{2}})^{\frac{1}{2}}\Big) \tag{11}$$

where μ_X and μ_Y denote the means of the two Gaussians, while Σ_X and Σ_Y denote the corresponding covariance matrices.
Actually, the above considerations can be extended to all the models belonging to the location-scale family of distributions, and hence we will be able to compute explicitly the Wasserstein metric for more flexible Bayesian models such as the one involving the Student-t distribution.

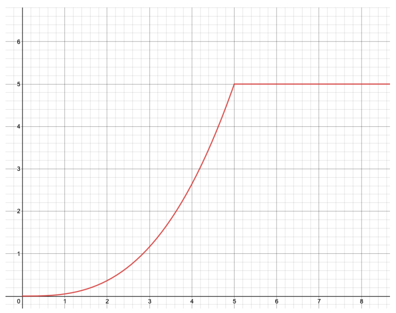
4.2 Model

As we have see in the previous subsection, the Wasserstein distance can give us information about how close two distributions are both in terms of location and scale.
A major issue concerning the basic Gaussian Mixture Model (2) is that it tries to fit the dataset under a Gaussian framework and this, as expected, leads to some clusters being identified by more than one distribution. The idea at this point is to employ the Wasserstein distance inside our Mixture Model in the prior for $(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)$ in order to penalize all the configuration of $(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)$ that are placed too close to each other.
In this way, whenever a distribution is not Gaussian distributed, even if the Gaussian Mixture Model will try to fit more than one Gaussian distribution, the penalization term will favour the spreading of the distributions, avoiding the aforementioned problem.
The penalized prior becomes the following:

$$\mathcal{L}([\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h]_1^H) \ \alpha \prod_{h=1}^{H} \mathcal{N}(\boldsymbol{\mu_h}|\boldsymbol{\mu_0}, \boldsymbol{\Sigma}_0)\mathcal{IW}(\boldsymbol{\Sigma_h}|\boldsymbol{\Phi}_0, \nu_0) \prod_{l<j} g(Wass(\mathcal{N}(\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l), \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j))) \tag{12}$$

where :

$$g(x) = \rho\Big(\frac{x-\delta}{\rho-\delta}\Big)^{\frac{1}{a}}\Big\{\delta \leq x \leq \rho\Big\} \text{ with } \delta = 0,\ \rho = 5 \text{ and } a = 0.35 \tag{13}$$



$g(x)$ is the Diggle-Graton function chosen in order to modulate the effect of the Wasserstein factor both to inflate the penalty effect for close clusters and to maintain the same contribution whenever they are adequately distant, making the penalizing effect virtually vanishing.
The hyperparameters have been set in order to fit the range of values usually taken by the Wasserstein distance.
Moreover we also decided to substitute the $\mathcal{NIW}$ showing in model (2) for $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with the product $\mathcal{N}\times\mathcal{IW}$, where the first distribution accounts for the mean, while the second accounts for the covariance matrix. This has been done to simplify our model and to study independently the simulation of the two terms. The full model then becomes:

$$y_i \mid c_i = h \overset{ind}{\sim} \mathcal{N}(\boldsymbol{\mu}_h, \boldsymbol{\Sigma_h})$$

$$\mathcal{L}([\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h]_1^H) \ \alpha \prod_{h=1}^{H} \mathcal{N}(\boldsymbol{\mu_h}|\boldsymbol{\mu_0}, \boldsymbol{\Sigma}_0)\mathcal{IW}(\boldsymbol{\Sigma_h}|\boldsymbol{\Phi}_0, \nu_0) \prod_{l<j} g(Wass(\mathcal{N}(\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l), \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j))) \tag{14}$$

$$P(c_i = h \mid \mathbf{w}) = w_h$$

$$\boldsymbol{w} \sim \mathit{Dirichlet}(\alpha_1, \dots, \alpha_H)$$

The hyperparameters are set as follows:

- $\boldsymbol{\mu}_0$ is centered on the mean of the data
- $\boldsymbol{\Sigma}_0$ is set to $\boldsymbol{I}$
- ν_0 is set to 3
- $\boldsymbol{\Phi}_0$ is set to $\boldsymbol{I}$

In order to sample from the posterior distributions we still rely on the Gibbs Sampler we used for the standard Mixture Model. The problem, in this case, is that we lost conjugacy for the full conditional $(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)|(\boldsymbol{\mu}, \boldsymbol{\Sigma})_{-h}, \boldsymbol{c}, \boldsymbol{y}$, and hence we cannot simulate directly from its posterior. Instead, we have to build a Metropolis-within-Gibbs step.
In particular, the full conditionals read:

$$(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)|(\boldsymbol{\mu}, \boldsymbol{\Sigma})_{-h}, \boldsymbol{c}, \boldsymbol{y} \; \alpha \; \mathcal{N}(\boldsymbol{\mu_h}|\boldsymbol{\mu_0}, \boldsymbol{\Sigma}_0)\mathcal{IW}(\boldsymbol{\Sigma_h}|\boldsymbol{\Phi}_0, \nu_0) \prod_{i:c_i=h} \mathcal{N}(\boldsymbol{y}_i|\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h) \prod_{l \neq h} g(Wass(\mathcal{N}(\boldsymbol{\mu}_l, \boldsymbol{\Sigma_l}), \mathcal{N}(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)))$$

$$\boldsymbol{w}|\boldsymbol{c} \sim \textit{Dirichlet}(\alpha_1 + n_1, \ldots, \alpha_H + n_H) \tag{15}$$

$$c_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{w}, \boldsymbol{y} \sim Cat(\frac{w_1 f(y_i|\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})}{\sum_{h=1}^{H} w_h f(y_i|\boldsymbol{\mu_h}, \boldsymbol{\Sigma_h})}, \ldots, \frac{w_H f(y_i|\boldsymbol{\mu_H}, \boldsymbol{\Sigma_H})}{\sum_{h=1}^{H} w_h f(y_i|\boldsymbol{\mu_h}, \boldsymbol{\Sigma_h})})$$

where n_i denotes the cardinality of cluster i.

Notice that if the cluster is empty, the term

$$\prod_{i:c_i=h} \mathcal{N}(\boldsymbol{y}_i|\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)$$

vanishes from the full conditional of the mean and covariance.

4.3 Metropolis-Hastings Step

In order to perform this simulation step we have to choose a proper proposal distribution to sample from. Since we want to use a random-walk Metropolis Hastings algorithm, the idea is to center our proposal distribution on the previous values of mean and covariance, that we will denote by $\boldsymbol{\mu}^{(n)}$ and $\boldsymbol{\Sigma}^{(n)}$.

The following pseudo-code describes how to sample from $(\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)|(\boldsymbol{\mu}, \boldsymbol{\Sigma})_{-h}, \boldsymbol{c}, \boldsymbol{y}$ via a M-H step:

1. Let $\boldsymbol{\theta}^{(n)} = (\boldsymbol{\mu}^{(n)}, \boldsymbol{\Sigma}^{(n)})$ be the actual state of $\boldsymbol{\theta}$.
2. Generate a candidate $\boldsymbol{\theta}^* = (\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$ from a proposal distribution $q(\boldsymbol{\theta}^{(n)}, \cdot)$ centered on $\boldsymbol{\theta}^{(n)}$.
3. Set $\boldsymbol{\theta}^{(n+1)} = \boldsymbol{\theta}^*$ with probability $min(1, \alpha)$ where:

$$\alpha = \frac{\pi(\boldsymbol{\theta}^*)q(\boldsymbol{\theta}^*, \boldsymbol{\theta}^{(n)})}{\pi(\boldsymbol{\theta}^{(n)})q(\boldsymbol{\theta}^{(n)}, \boldsymbol{\theta}^*)} \tag{16}$$

   In the case the proposal distribution is symmetric, α reduces to:

$$\alpha = \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta}^{(n)})} \tag{17}$$

From the above pseudo-code we can see that the choice of the proposal distribution is of paramount importance for the efficiency of the whole sampler. In the following paragraphs are listed some failing and some successful proposal distributions that we implemented.

4.3.1 Failed Attempts

Our first idea was based on the use of a Normal distribution to sample from the previous mean and a Inverse-Wishart with mean $\mathbf{\Sigma}^*$ to sample the covariance matrix. These proposals immediately arise some issues: firstly, the overall proposal is not symmetric due to the Inverse-Wishart and hence it shows up during the computation of parameter α. Moreover, it was not straightforward to control the oscillations of the covariance matrix and this behaviour led to some draws that were always rejected or some other that were always accepted.
In conclusion, the algorithm as it was firstly designed did not bring any satisfactory result and we identified the problem in the way we were sampling the covariance matrix.

To overcome this issue, we thought as first solution to try to sample each one of the three components of the covariance matrix (diagonal terms, plus the extra-diagonal term) from a Normal distribution, forcing them to maintain the crucial property of symmetry and positive definiteness of the matrix. Unfortunately, also in this case the result were not satisfactory and the behaviour of the algorithm was erratic, with some runs converging and other diverging.

4.3.2 Unconstrained Representation

Since using a Normal distribution to sample $\boldsymbol{\mu}^*$ provides some good traceplots, we would like to use a Normal also to sample $\mathbf{\Sigma}^*$. Notice that the precision parameter is distributed over positive semi-definite matrices, therefore we have to maintain this constrain throughout some suitable reparametrization.
To achieve such a sampling strategy, we use a bijector $\mathcal{B} : \mathcal{R}^{d^2} \to \mathcal{R}^{d(d+1)/2}$ between the space of $d \times d$ symmetric, positive definite matrices and the space of $d(d+1)/2$ vectors.
We can therefore perform a change of variables with respect to the covariance matrix $\mathbf{\Sigma}_h$ as follows:

$$(\boldsymbol{\mu}_h, \boldsymbol{\gamma}_h)|(\boldsymbol{\mu}, \mathbf{\Sigma})_{-h}, \boldsymbol{c}, \boldsymbol{y} \;\alpha\; \mathcal{N}(\boldsymbol{\mu_h}|\boldsymbol{\mu_0}, \mathbf{\Sigma}_0)\mathcal{IW}(\mathbf{\Sigma_h}|\mathbf{\Phi}_0, \nu_0) det\mathcal{J}(\mathcal{B}^{-1}(\boldsymbol{\gamma}_h))$$
$$\prod_{i:c_i=h} \mathcal{N}(\boldsymbol{y}_i|\boldsymbol{\mu}_h, \mathbf{\Sigma}_h) \prod_{l\neq h} g(Wass(\mathcal{N}(\boldsymbol{\mu}_l, \mathbf{\Sigma_l}), \mathcal{N}(\boldsymbol{\mu}_h, \mathbf{\Sigma}_h))) \tag{18}$$

where $\boldsymbol{\gamma}_h = \mathcal{B}(\mathbf{\Sigma}_h)$.
In this way, any covariance matrix can be sampled using a Multivariate Normal centered on the vectorized covariance matrix itself.
Moreover we also attain a symmetric proposal that simplifies the computation of α.

At this point the pseudo-code presented before becomes:

1. Let $\boldsymbol{\theta}^{(n)} = (\boldsymbol{\mu}^{(n)}, \boldsymbol{\gamma}^{(n)})$ be the actual state of $\boldsymbol{\theta}$.
2. Generate a candidate $\boldsymbol{\theta}^* = (\boldsymbol{\mu}^*, \boldsymbol{\gamma}^*)$ such that $\boldsymbol{\mu}^* \sim \mathcal{N}(\boldsymbol{\mu}^{(n)}, \beta_1 I))$ and $\boldsymbol{\gamma}^* \sim \mathcal{N}(\boldsymbol{\gamma}^{(n)}, \beta_2 I))$ where β_1, β_2 are some hyper-parameters.
3. Set $\boldsymbol{\theta}^{(n+1)} = \boldsymbol{\theta}^*$ with probability $min(1, \alpha)$ where

$$\alpha = \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta}^{(n)})} =$$
$$= \frac{\mathcal{N}(\boldsymbol{\mu^*}|\boldsymbol{\mu_0}, \mathbf{\Sigma}_0)\mathcal{IW}(\mathbf{\Sigma^*}|\mathbf{\Phi}_0, \nu_0) det\mathcal{J}(\mathcal{B}^{-1}(\boldsymbol{\gamma}^*)) \prod_{l\neq h} g(W(\mathcal{N}(\boldsymbol{\mu}_l^*, \mathbf{\Sigma_l^*}), \mathcal{N}(\boldsymbol{\mu}_h^*, \mathbf{\Sigma}_h^*)))}{\mathcal{N}(\boldsymbol{\mu^{(n)}}|\boldsymbol{\mu_0}, \mathbf{\Sigma}_0)\mathcal{IW}(\mathbf{\Sigma^{(n)}}|\mathbf{\Phi}_0, \nu_0) det\mathcal{J}(\mathcal{B}^{-1}(\boldsymbol{\gamma}^{(n)})) \prod_{l\neq h} g(W(\mathcal{N}(\boldsymbol{\mu}_l^{(n)}, \mathbf{\Sigma_l^{(n)}}), \mathcal{N}(\boldsymbol{\mu}_h^{(n)}, \mathbf{\Sigma}_h^{(n)})))} \tag{19}$$

4.4 Shoot away proposal

The proposal used up to now displayed some problems. In particular, when the algorithm was near to convergence, the non-emtpy clusters basically did not update anymore, having an extremely low acceptance rate. This is probably due to the fact that, once near to convergence, the sampled proposal covariance matrices are not acceptable anymore and lower values of hyper-parameters β_1 and β_2 would be needed to raise the acceptance rate at a more suitable value. This leads to bad traceplots, as can be seen in the figure below:

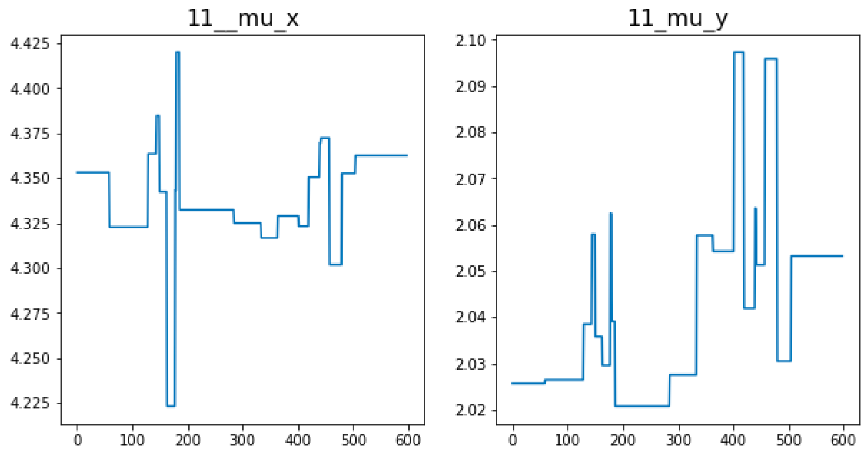


Figure 10: Bad traceplot behaviour.

Aiming to solve this issue and obtain better separation between components, we choose a new proposal made up of a mixture of two Gaussians, both with the same mean but a different covariance matrix. The proposal is the following:

$$0.9\mathcal{N}(\boldsymbol{\mu}_h, \alpha_1 I) + 0.1\mathcal{N}(\boldsymbol{\mu}_h, 3I) \tag{20}$$

with $\alpha_1 = 0.1$.

The idea is that in average, one proposal for the mean out of ten will be 'shot' away and consequently more distanced from the other components, helping to obtain quicker convergence and better separation between clusters.
An example of traceplots obtained with this proposal is the following:

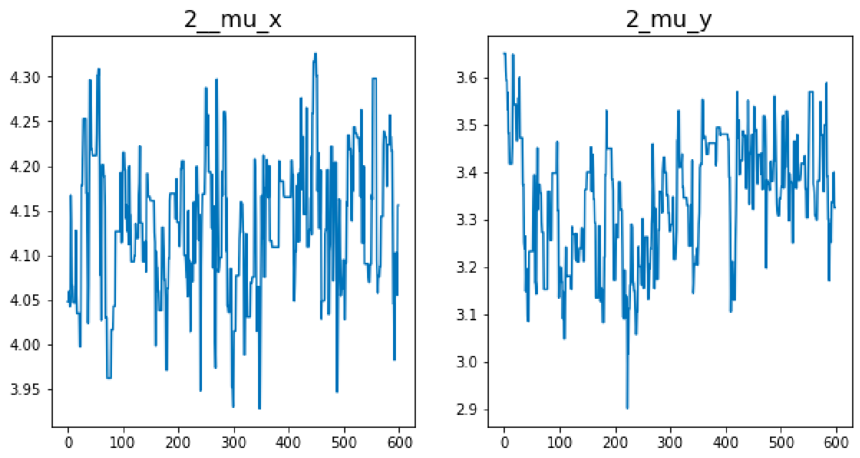


Figure 11: Better traceplot behaviour.

4.5 MALA Proposal

The MALA, which stands for Metropolis-adjusted Langevin algorithm, is a type of MCMC algorithm. It combines two methods to propose and decide whether to accept or reject a new state for the Markov Chain. In particular, the proposal uses Langevin dynamics, which is based on the evaluation of the gradient of the logarithm of the target distribution. In our case, this is once again the full conditional of the scale and location parameters of each component of the mixture. A general form for the MALA proposal is the following:

$$\boldsymbol{\theta}^* := \boldsymbol{\theta}^{(n)} + \tau \nabla \log \pi(\boldsymbol{\theta}^{(n)}) + \sqrt{2\tau}\epsilon \tag{21}$$

where τ is a fixed parameter, π the target distribution and ϵ a vector from a Gaussian distribution in $\mathcal{R}^d$ with mean $\mathbf{0}$ and I covariance matrix. This formulation is handy when $\boldsymbol{\theta}^*$ is a vector, however our state space is defined both by the mean and the covariance matrix. We can see this proposal equivalently as a Gaussian centered on $\boldsymbol{\theta}^{(n)} + \tau \nabla \log \pi(\boldsymbol{\theta}^{(n)})$ with covariance matrix $\sqrt{2\tau}I$.
With the above considerations in mind, we can finally treat the covariance using once again the bijection between symmetric, positive definite matrices and vectors. In our specific case, the proposal for $\boldsymbol{\mu}_h$ and $\boldsymbol{\gamma}_h$ hence becomes:

$$\boldsymbol{\mu}_h^* := \boldsymbol{\mu}_h^{(n)} + \tau \nabla \log \pi(\boldsymbol{\mu}_h^{(n)}, \boldsymbol{\gamma}_h^{(n)}) + \sqrt{2\tau}\epsilon \tag{22}$$

$$\boldsymbol{\gamma}_h^* := \boldsymbol{\gamma}_h^{(n)} + \tau \nabla \log \pi(\boldsymbol{\mu}_h^{(n)}, \boldsymbol{\gamma}_h^{(n)}) + \sqrt{2\tau}\epsilon \tag{23}$$

where $\boldsymbol{\gamma}_h = \mathcal{B}(\boldsymbol{\Sigma}_h)$.

It is important to notice that to sample $\boldsymbol{\gamma}_h^*$ two different ways can be used. Indeed, it is possible to compute the gradient of the full conditional with respect to the covariance matrix $\boldsymbol{\Sigma}_h$ instead of the vectorized covariance, $\boldsymbol{\gamma}_h$. In the former case, the bijection would be applied only after summing the gradient to the old covariance matrix. This step however can be very problematic, as we experienced. In fact, if the gradients, for any reason, have sufficiently big values, the resulting :

$$\boldsymbol{\Sigma}_h^{(n)} + \tau \nabla \log \pi(\boldsymbol{\mu}_h^{(n)}, \boldsymbol{\Sigma}_h^{(n)}) \tag{24}$$

might not be a s.p.d. matrix, and the bijection cannot be applied.
Computing the gradient with respect to $\boldsymbol{\gamma}_h$ and adding the gradient $\tau \nabla \log \pi(\boldsymbol{\mu}_h^{(n)}, \boldsymbol{\gamma}_h^{(n)})$ instead yields a vector which can be surely transformed back to a s.p.d. matrix, regardless of the value of τ.
The acceptance-rejection phase of the Metropolis algorithm is carried out exactly as outlined in the previous sections.

4.6 Discussion

In our case studies, MALA proposal brought improvements to the Metropolis step. The traceplots show much more mixing of the chain, as can be seen in the following figures:

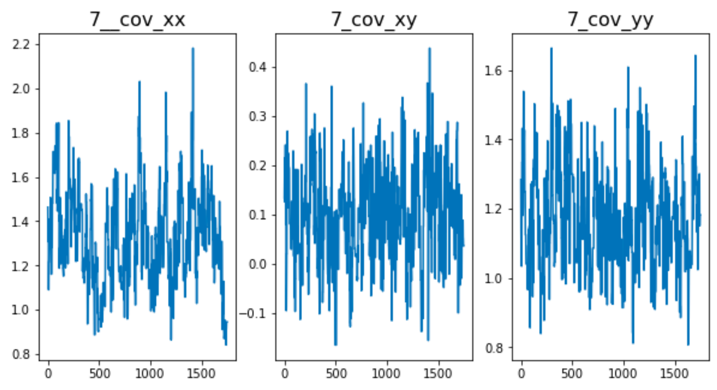


Figure 12: Traceplot of covariance terms in component 7.

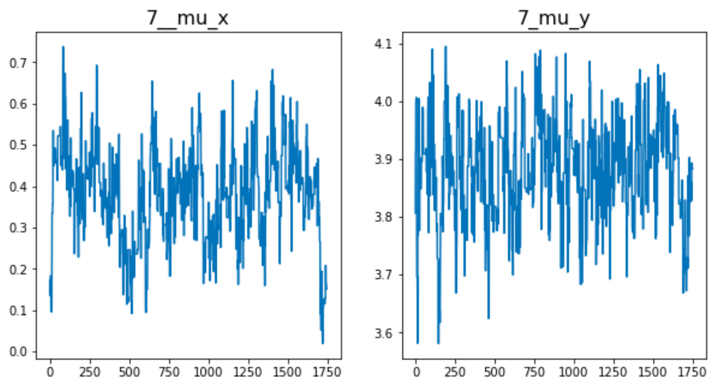


Figure 13: Traceplot of mean terms in component 7.

As will be discussed in the following sections, the overall performance of the clustering does not change significantly, however the state exploration is much more exhaustive. This is extremely important and helps to avoid the problem of a falling acceptance rate we experienced with other proposals. Setting the τ parameter is not always straightforward, because it determines both the step size taken when adding the gradient, both the covariance matrices of the proposal. For this reason, controlling the acceptance rate seems to be more tricky.

4.7 Dataset Test 1

As with the other methods, we tested the algorithm with the MALA proposal on our two test datasets. We ran the algorithm with several values of τ and found a good value of $\tau = 10^{-4}$, with the acceptance rate centered on $(0.2, 0.3)$. The resulting clustering seemed to behave better than before, with a few runs showing quite satisfying results, as can be seen in the following figure, where a good proportion of the cluster allocations is devoted to just four components:

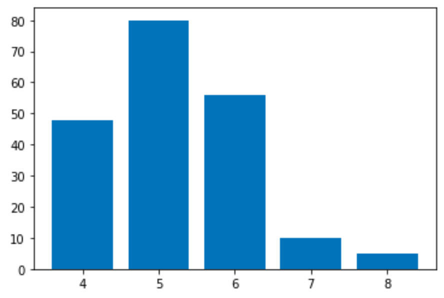


Figure 14: Barplot of number of components used during the iterations.

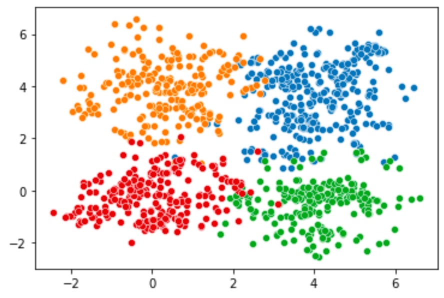


Figure 15: Resulting best cluster selected by Binder Loss minimization.

Most of the runs actually displayed slightly worse results, generally with the following patterns:

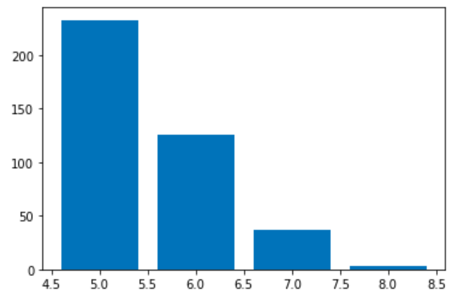


Figure 16: Barplot of number of components used during the iterations.

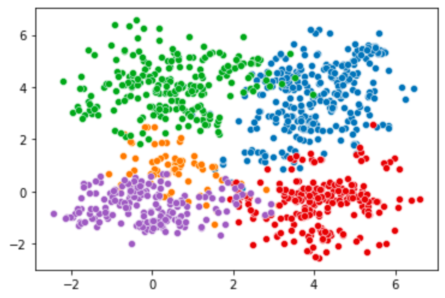


Figure 17: Resulting best cluster selected by Binder Loss minimization.

Still, we must highlight that the penalized model struggles identifying the true cluster structure of the fictitious test dataset, which seems to be affected more heavily by the initial random initializations. As future work, one could suggest to better design such a crucial step.

4.8 Dataset Test 2

The performance on the Faithful dataset is very good. Below are the results from a run of the algorithm.

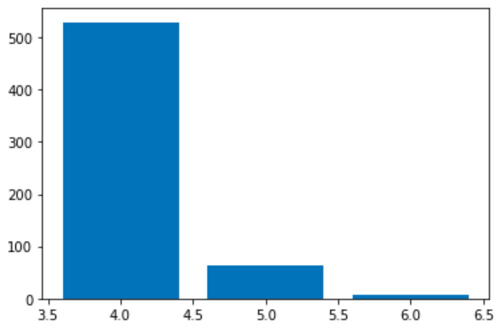


Figure 18: Barplot of number of components used during the iterations.

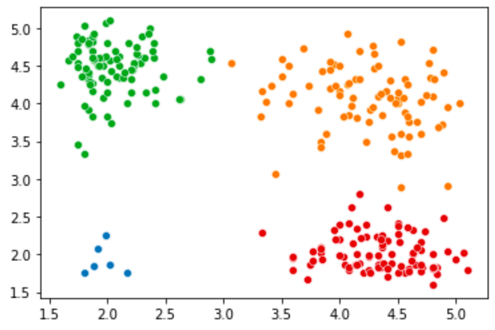


Figure 19: Resulting best cluster selected by Binder Loss minimization.

As we can see in the figures above, all four clusters are correctly identified and the algorithm almost always proposes four clusters during the iterations, showing high consistency during several runs. As expected, this result is perfectly aligned with that obtained using the standard Mixture Model, although being more consistent estimating the true number of clusters.

4.9 Some comments on the traceplots

During most of the runs, there are some recurrent behaviours of the traceplots which seem to be always present after the introduction of the Wasserstein penalization term in the prior. More specifically, it is very frequent to observe some components of the mixture to reach very high values in the covariance matrix terms, which subsequently appear to diverge.
An example of this can be seen in the following traceplot:

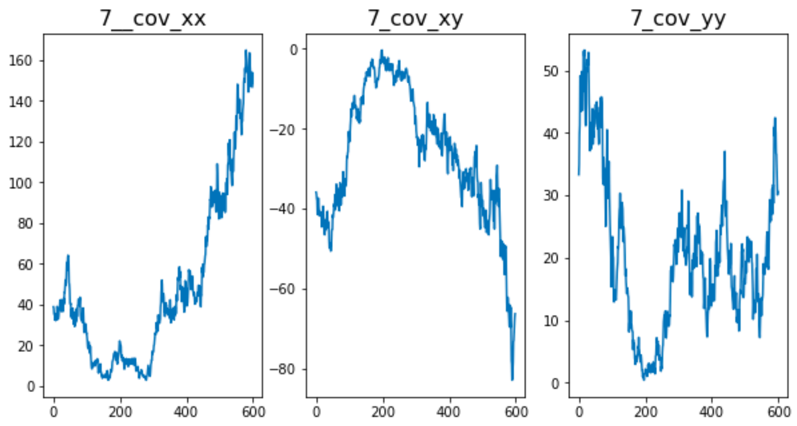


Figure 20: traceplots of diverging covariance matrix.

Of course the effect on the overall clustering is non detectable, since to these components are not assigned any data points. Moreover, this only seems to happen when components are empty, hence it is probably a consequence of the prior structure: the full conditionals related to empty clusters are missing the data-likelihood term, and are therefore susceptible to be allocated as spread away as possible.

Interestingly, this behaviour does not appear in the traceplots related to the means. For example, the following traceplot shows a mean component which during burn-in had very high values but, during the convergence, is pulled back near the other components.

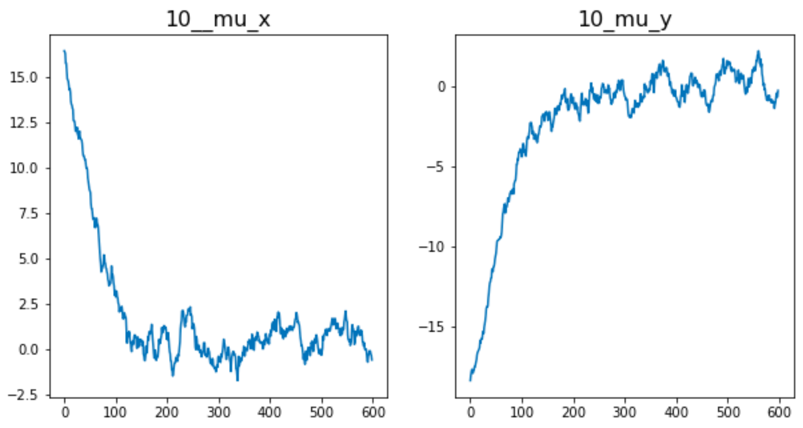


Figure 21: Mean component not diverging.

5 Student-t Mixture Models with the Wasserstein Distance

In order to capture clusters of data generated by more flexible distributions, we implemented another version of our algorithm, based on a mixture of Student-t distributions.
The student-t distributions has heavier tails with respect to the Gaussian distribution. As the degrees of freedom grow, the Student-t approximates a Gaussian distribution, while smaller values make the tails heavier so that we can identify more spread-out clusters.

5.1 Model

We now have a different parametrization, based on the scale matrix $\mathbf{\Lambda}$ and on the degrees of freedom ν. We decided to fix the degrees of freedom to the default value 7, in this way the covariance matrix is simply computed as:

$$\mathbf{\Sigma}_{\boldsymbol{h}} = \frac{\nu}{\nu - 2} \mathbf{\Lambda}_{\boldsymbol{h}} \tag{25}$$

so that there is no need to change the prior distributions for $\boldsymbol{\mu}_h$ and $\mathbf{\Sigma}_{\boldsymbol{h}}$.

The model is the following:

$$y_i \mid c_i = h \overset{ind}{\sim} Student - t(\boldsymbol{\mu}_h, \mathbf{\Lambda}_{\boldsymbol{h}}, \nu) \tag{26}$$

$$\mathcal{L}([\boldsymbol{\mu}_h, \mathbf{\Sigma}_h]_1^H) \ \alpha \prod_{h=1}^{H} \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{h}} | \boldsymbol{\mu}_{\mathbf{0}}, \mathbf{\Sigma}_0) \mathcal{IW}(\mathbf{\Sigma}_{\boldsymbol{h}} | \mathbf{\Phi}_0, \nu_0) \prod_{l<j} g(Wass(Student{-}t(\boldsymbol{\mu}_l, \mathbf{\Lambda}_l, \nu), Student{-}t(\boldsymbol{\mu}_j, \mathbf{\Lambda}_j, \nu)))$$

$$P(c_i = h \mid \mathbf{w}) = w_h$$

$$\boldsymbol{w} \sim Dirichlet(\alpha_1, \dots, \alpha_H)$$

The hyperparameters are set as follows:

- $\nu = 7$
- $\boldsymbol{\mu}_0$ is centered on the mean of the data
- $\mathbf{\Sigma}_0$ is set to $\boldsymbol{I}$
- ν_0 is set to 3
- $\mathbf{\Phi}_0$ is set to $\boldsymbol{I}$

5.2 Discussion

Substituting the Gaussian distribution with the Student-t, the entire procedure to sample from the posterior is equivalent to the one described in section 4, hence a Gibbs sampler with a Metropolis-Hastings step is used.
We also maintained the same proposal distributions for $\boldsymbol{\mu}_{\boldsymbol{h}}$ and for $\mathbf{\Sigma}_h$ (i.e. a mixture of Gaussian distributions for the location parameter, and a Gaussian distribution for the covariance matrix).
Moreover, as we noticed in subsection 4.1, the Wasserstein metric between two Student-t distributions can be computed in closed-form and therefore easily implemented.

5.3 Dataset Test 1

We tested the algorithm with the new model on the same, fictitious datasets of the previous simulations. The number of iterations is set to 4000, with a burn-in of 1000 and thinning equal to 5.
The results are quite satisfying, even if they are worse than the ones obtained with the Gaussian components. The algorithm identifies five clusters for the most of the iterations, one more than the original number of clusters of the dataset.

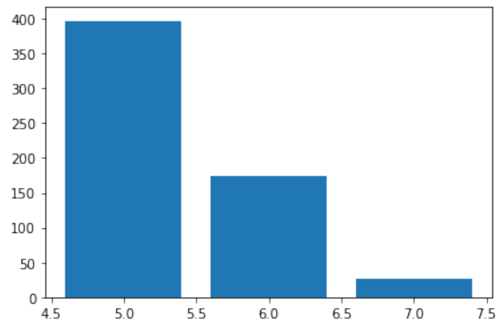


Figure 22: Barplot of number of components used during the iterations

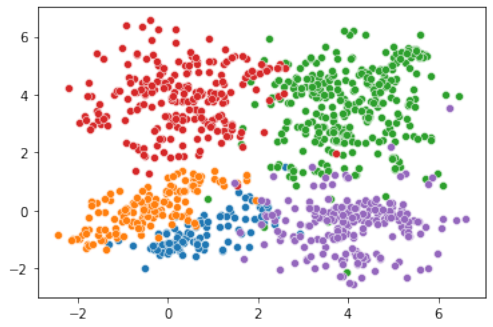


Figure 23: Resulting best cluster selected by Binder Loss minimization.

5.4 Dataset Test 2

Contrary to the Gaussian Mixture, the performance on the Old Faithful Geyser dataset is not really satisfactory. Surprisingly, the algorithm identifies three clusters, joining together the two numerous clusters on the right in Figure 25.
This behaviour could be caused by the heavy tails of the Student-t distribution, which give higher weight to the points distant from the center with respect to Gaussian distribution.
Moreover, the above issue could depend on the value assigned to the d.o.f. of the Student-t distribution: more details on this topic are discussed in the subsequent sections.

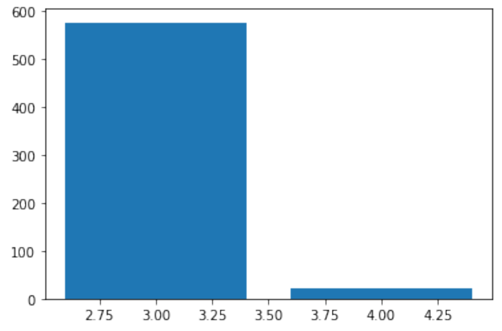


Figure 24: Barplot of number of components used during the iterations

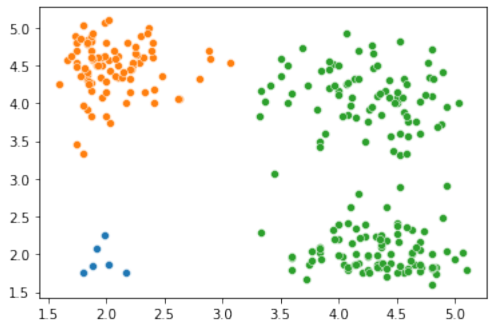


Figure 25: Resulting best cluster selected by Binder Loss minimization.

5.5 Discussion

5.5.1 Traceplots

Regarding the exploration of the parameter space, we obtained satisfactory results for the location parameter, even if not as good as with the MALA proposal. An example of traceplot related to the location parameter of one component can be seen in Figure 26. Such a traceplot was obtained during testing on the fictitious dataset.

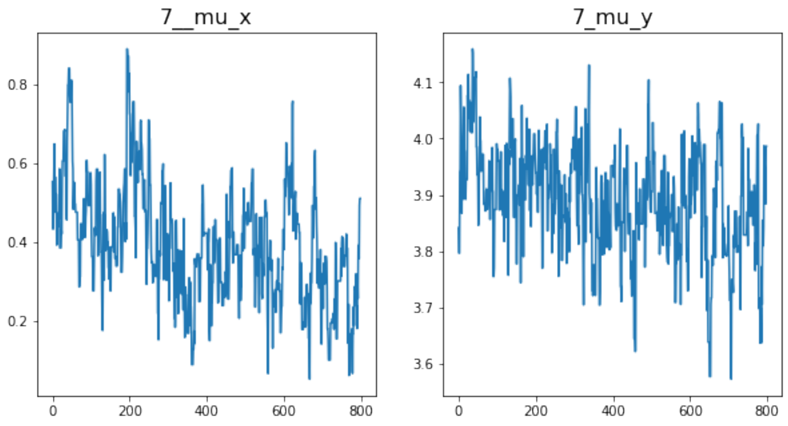


Figure 26: Traceplot of a location parameter

For what concerns the covariance matrix we have a good exploration of the parameter space for the non-empty clusters, while we still notice the divergent trend of the covariance related to the empty clusters. The two different behaviours can be seen in two examples in the figures below:

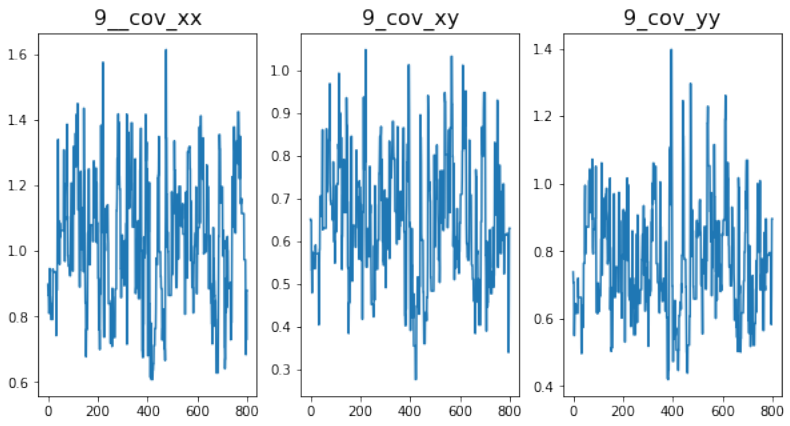


Figure 27: Traceplot of a covariance matrix

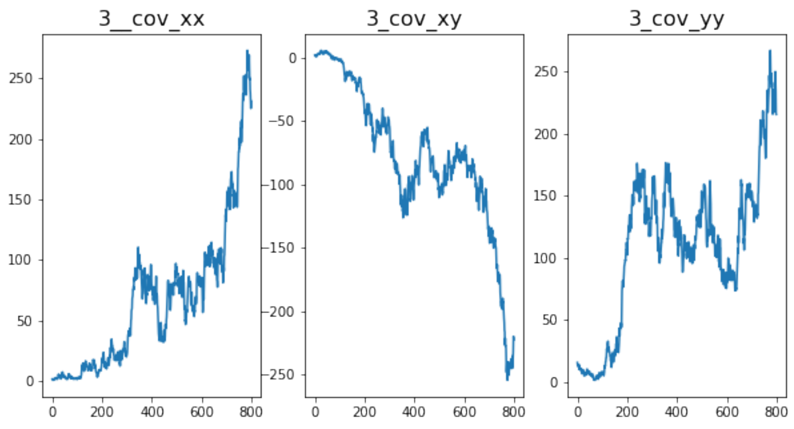


Figure 28: Traceplot of a diverging covariance matrix

5.5.2 Degrees of freedom

We notice that by choosing the default value for the degrees of freedom of the distribution, the algorithm becomes highly unstable with respect to this parameter. In fact, as shown in the figures below, the clustering results on the fictitious dataset are really different when we use different values for the degrees of freedom, for instance $d.o.f. = 7$ and $d.o.f. = 9$.

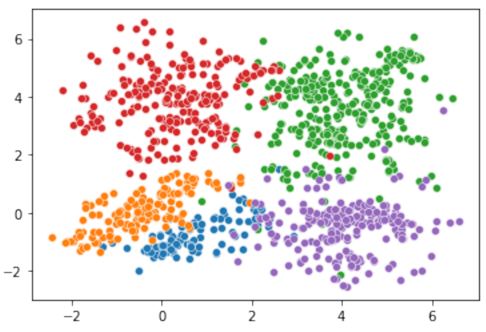


Figure 29: Resulting best cluster selected by Binder Loss minimization, degrees of freedom equal to 7

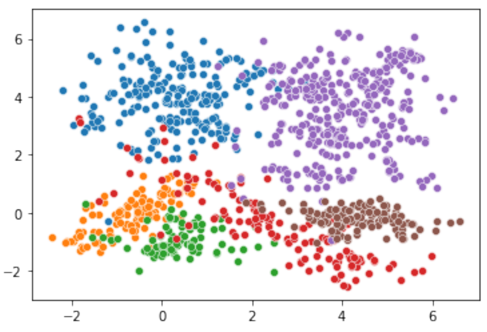


Figure 30: Resulting best cluster selected by Binder Loss minimization, degrees of freedom equal to 9

In order to make the algorithm more robust with respect to the initialization of the degrees of freedom, a further step could be to set a prior distribution also for this parameter. In this way, the algorithm may adapt its value depending on the data to analyze.
In addition to this, another improvement in the results could come from the adaptation of the proposal distributions to the case of the Student-t distribution. Indeed, for the simulation results we are showing in this section, we used the exact same shoot-away proposal (20) we exploited for the Gaussian model, and even if the results are quite satisfying, we think that they could improve with more research.

6 Conclusions and Further Developments

In this report we tackled the problem of Bayesian clustering using Mixture Models. First, we emphasized the criticalities arising when standard Gaussian Mixture Models are used to estimate the true number of clusters in the observations. Indeed, especially when the data becomes more and more complex, such a finite Mixture struggles separating the components and eventually over-estimates the underlying cluster structure.

To overcome this fallback, we initially proposed a Repulsive Gaussian Mixture Model taking advantage of the Wasserstein distance into the prior assumptions. In this way, the model could naturally discourage closeness between distinct components and therefore perform better in estimating the true number of clusters. However, the price to pay for introducing such a repulsive prior is that the complexity of the model dramatically increases, both from the analytical and computational point of view.
Indeed, the use of a Metropolis-within-Gibbs step in order to simulate from the posterior of the component-specific parameters introduces several difficulties, related for example to the choice of an efficient proposal distribution. In this sense, the combination of an unconstrained state-space representation and a MALA proposal remarkably improved our results, which were quite satisfying both on the Old Faithful Geyser and the fictitious dataset. Also, the modulation of the Wasserstein distance through the Diggle-Graton function made the whole procedure more robust.
Moreover, we highlighted how the presence of a Wasserstein-dependent term in the model influences the traceplots of the covariance matrices related to empty clusters.

Then, to allow for more flexibility, we proposed a Repulsive Student-t Mixture Model still relying on a repulsive, Wasserstein-dependent prior. The main idea was again that of favouring better separation between cluster. In addition, the existence of a closed-form expression for the Wasserstein metric between Student-t distributions represented a supplemental motivation for considering such an updated model.
On the other hand, many features have to be taken into account when dealing with Student-t distributions. For instance, the degrees of freedom hugely determines the behaviour of the distribution and must be tuned consequently. In this sense, we adopted a fixed, default value for the d.o.f., but as we noticed in the discussion of results, it may be interesting to assume a prior for it so that the algorithm becomes more robust. Another improvement could be represented by the choice of a different proposal distribution, which better suits the problem at hand.

It is mandatory to mention the effort we put in efficiently implementing all the simulation procedures involved in this project. In particular, by means of the Python High Performance Computing library JAX, we are able to compile the code and obtain much faster computations. Facilitating simulation was a crucial aspect since we managed to perform several runs of our algorithms and therefore be able to detect anomalies with much faster pace.

Finally, we can consider some natural future developments of this work: one may consider to allow for even more flexibility in the model by considering a Mixture of Skewed-Normals. This approach, however, is hard to tackle since such a distribution is not described in terms of location and scale, and hence we do not have a closed-form expression for the Wasserstein distance. However, one could still approximate the Wasserstein integral by means of a quadrature rule (preferably in low-dimensional spaces) or a Monte Carlo method.

7 References

- [1] Jessica Franzén. *Bayesian Inference for a Mixture Model using the Gibbs Sampler* (2006). See : http://gauss.stat.su.se/rr/RR2006_1.pdf
- [2] F.A. Quintana, P.L. Iglesias. *Bayesian clustering and product partition models.* (2003). Journal of the Royal Statistical Society: Series B, pp.557–574.
- [3] Mario Beraha, Raffaele Argiento, Jesper Møller, Alessandra Guglielmi. *MCMC computations for Bayesian mixture models using repulsive point processes* (2020). See : arXiv:2011.06444
- [4] Francesca Petralia, Vinayak Rao, David B. Dunson. *Repulsive Mixtures* (2012). See : arXiv:1204.5243
- [5] Julie Delon, Agnès Desolneux. *A Wasserstein-type distance in the space of Gaussian Mixture Models* (2020). SIAM Journal on Imaging Sciences, Society for Industrial and Applied Mathematics, pp. 936-970
- [6] M. Gelbrich. *On a Formula for the L2 Wasserstein Metric between Measures on Euclidean and Hilbert Spaces* (1990).
- [7] Lau, J. W., Green, P. J. *Bayesian Model-Based Clustering Procedures* (2007). Journal of Computational and Graphical Statistics, pp.526-558. See: http://www.jstor.org/stable/2759425
- [8] Sara Wade, Zoubin Ghahramani. *Bayesian cluster analysis: Point estimation and credible balls* (2018).
- [9] D.A. Binder. *Bayesian Cluster Analysis* (1978). Biometrika, 65:31–38.
- [10] F. Xie, Y. Xu. *Bayesian Repulsive Gaussian Mixture Model* (2017).