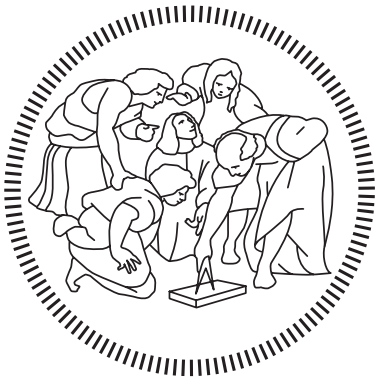


Politecnico di Milano
M.Sc. in Mathematical Engineering

Bayesian statistics
Academic Year 2021-2022

Distance-based probabilistic clustering for functional data

Authors: Giulia Caruso, Alessio Facincani, Giulia Romani, Pietro Spina, Matteo Vescovi [1]
Tutors: Mario Beraha [2], Riccardo Corradin [3]

Abstract

This is a project report of the *Bayesian Statistics* course, held by Professor A. Guglielmi at Politecnico di Milano during academic year 2021/2022.

Somatosensory Evoked Potential (SEP) are electrical activity occurring in the central nervous system following a stimulus, detected within 2 seconds after the stimulus over 1600 instants. SEP reflects the processing of information from the peripheral levels down to the cortical structures, and are perhaps useful in assessing the integrity of nerve conduction pathways and the different areas involved in the reception and processing of sensory stimuli.

The patients under study, in a state of coma, underwent a neuropsychological evaluation within 72 hours after surgery. We consider a dataset composed of 26 multivariate functional observation with 4 components, each of which represents a Somatosensory Evoked Potential, detected within 2 seconds after the stimulus over 1600 time instants.

The aim of the project is to implement a Bayesian functional clustering algorithm in a Generalized Bayes Product Partition Model framework (GB-PPM), considering a generalization of the Mahalanobis distance in a functional setting. At the beginning we will consider an uniform prior over the cluster partition with the number of clusters fixed a priori, then we will switch our focus over an EPPF Pitman-Yor process prior. Both algorithms will first be tested on simulated data and then on clinical ones.

[1] Mathematical Engineering Students at Politecnico di Milano.
[2] PHD Researcher at Politecnico di Milano
[3] Postodctoral Researcher at Università degli Studi di Milano-Bicocca

Contents

Chapter 1

Introduction

1.1 Dataset Presentation

The Clinical dataset consists of functional data describing the Somatosensory Evoked Potential (SEP) of 26 patients during coma state, patient information and outcome evaluations.
SEPs are electrical changes that occur in the central nervous system in response to an external electrical stimulus. The importance of their study lies in their usefulness in assessing the ability of the nervous system to receive and process information.

Functional data are collected through the use of four electrodes, placed in frontal and central positions on the skull of patients. The central positions reflect an initial processing of the stimulus, whereas in the frontal positions a stimulus subject to reprocessing is detected. An external stimulus produces an action potential in neurons, which is transmitted through nerve fibers and to the cerebral cortex. Electrodes placement allows the action potential to be detected and to record an electrical deflection, of certain amplitude and at a certain latency.

To assess the integrity of different areas of the brain, two separate quantities are measured for each patient. A first non-painful measurement with an intensity of 10-15 mA, to collect the SLSEP component (Short Latency SEP), and a second painful measurement with an intensity of 50-100 mA for the PMLSEP component (Pain-related Middle Latency SEP).
For each quantity, 1600 measurements are recorded.

Electrical deflections can occur downward, called negative (N) waves, or upward, called positive (P) waves. Moreover, the transmission of the potential occurs over a period of time proportional to the conduction velocity of the nerve fiber. Therefore, the waves have a specific latency period, and they are classified according to this as N20, P25, P45 and N60. The last two are later evoked potentials that reflect the activation of other brain areas and the reprocessing of information.

In addition to functional data representing the SLSEP and PMLSEP components, outcome assessments were measured for each patient. These measures are an evaluation of physical and cerebral recovery, after exit of the coma state and once rehabilitation is complete.
In addition, some information about patients are provided, such as age, gender, some characteristics of hospitalization and rehabilitation, etiology of the pathology, cerebral hemisphere in which pathology occurred, the duration of coma state and duration of rehabilitation.

1.2 Research Goal

The goal is to perform clustering of functional data representing Somatosensory Evoked Potential, based on a Generalized Bayes Product Partition Model, using Mahalanobis distance in the functional case. Our hope is to establish a relation between the clustering results and other medical parameters, like how well the patient recovered from the coma.

Chapter 2

Model

We provide the following setting for the model. Define:

- $x_i = (x_{i1}, ..., x_{id})^T$ as the vector of observations $(\forall i = 1, ..., n)$
- X as the collection of all data points
- K as the number of clusters (fixed a priori, according to the data)
- $C = (C_1, ..., C_K)$ as a partition of {1,...,n} into K sets
- $X_k = \{x_i : i \in C_k\}$ as the k-th cluster
- $c = (c_1, ..., c_n)$ as the cluster labels: $c_i = k$ iff $i \in C_k$ $(\forall i = 1, ..., n)$

Why not a typical Bayesian model?
Typical Bayesian models for clustering are based on the posterior distributions of the form

$$\pi(x|\theta) \propto \pi(c) \prod_{k=1}^{K} \left[\int_{\Theta} \pi(x_i|\theta)\pi(\theta)d\theta \right] \tag{2.1}$$

where $\pi(c)$ is the prior distribution of c, $\pi(x|\theta)$ is the within-cluster likelihood and $\pi(\theta)$ is the prior law generating the cluster specific parameter.

Although (2.1) forms the basis for a vast literature on Bayesian clustering, we aim to examine the possible connection between distance based and model based clustering. The attempt to bridge these two approaches lead us to choose a different model.

2.1 Gibbs Posterior

We consider the generalized posterior

$$\pi(c|\lambda, X) \propto \pi(c) \exp\{-\lambda \ell(c; X)\} \tag{2.2}$$

with parameter $\lambda > 0$ and loss function[1] $\ell(c; X) > 0$.

We now introduce a class of *generalized Bayes product partition models* (GB-PPM) for clustering such that it is characterized by a factorized loss function $\ell(C; X)$ of the form:

$$\ell(c; X) = \sum_{k=1}^{K} \sum_{i \in C_k} \mathcal{D}(x_i; X_k) \tag{2.3}$$

where $\mathcal{D}(x_i; X_k)$ is a function that quantifies the discrepancy of the i-th unit from the k-th cluster.

[1] Standard Bayesian inference is a particular case of (2.2), occurring when $-\lambda\ell(c; X)$ is a negative log-likelihood.

At this point of the project, we choose a uniform clustering prior (i.e. uniform over partitions having K components) for the vector of labels c:

$$\pi(c) = \frac{1}{\mathcal{S}(n, K)} \tag{2.4}$$

where $\mathcal{S}(n, K) = \frac{1}{K!} \sum_{k=0}^{K} (-1)^{K-k} K! \left\{(K-k)!k!\right\}^{-1} k^n$ is the Stirling number of the second kind.

Substituting the loss function (2.3) into the Gibbs posterior (2.2) and thanks to the uniform prior (2.4), *The generalized Bayes posterior under a GB-PPM* has the form:

$$\pi(c|\lambda, X) \propto \pi(c) \prod_{k=1}^{K} \rho(C_k; \lambda, X_k) \propto \prod_{k=1}^{K} exp\{-\lambda \sum_{i \in C_K} \mathcal{D}(x_i; X_k)\} \tag{2.5}$$

The main problem is now to decide which distance $\mathcal{D}(x_i; X_k)$ is the best one for the data we are dealing with.

2.2 Mahalanobis distance

Let X be a random variable taking values in $\mathbb{R}^d$ with non-singular covariance matrix Σ and let $x, y \in \mathbb{R}^d$ be two observations drawn from X. If we want to compute the distance between x and y, the *Euclidean distance* is not suitable because it disregards the standard deviations and the covariances of the components of X. Hence a good alternative is the *Mahalanobis distance*, defined as:

$$M(x, y) = ((x - y)' \Sigma^{-1} (x - y))^{1/2} \tag{2.6}$$

Mahalanobis distance in the functional case
The previously defined distance works really well when the data are *discrete*, but we need to adapt it to the case of functional data. In order to proceed, let $X(t) \in L^2[0, 1]$, $t \in [0, 1]$ be a stochastic process with mean $m(t) = \mathbb{E}(X(t))$ and let $X_1(t), ..., X_n(t)$ be a sample of observations (trajectories) drawn from $X(t)$. Assume that $\|\cdot\|$ and $\langle\cdot\rangle$ correspond, respectively, to the norm and inner product in $L^2[0, 1]$. Now define:

- The *covariance function* $K = K(s, t) = Cov(X(s), X(t))$,
- The *covariance operator* $\mathcal{K} : L^2[0, 1] \to L^2[0, 1]$ such that $\mathcal{K}f(t) = \int_0^1 K(t, s) f(s) ds$,

and we require/observe that:

- K is continuous, which implies $m \in L^2[0, 1]$ and $\mathbb{E}\|X\|^2 < \infty$,
- $\mathcal{K}$ is defined starting from K, is linear and injective, i.e. all its eigenvalues are strictly positive.

In order to adapt (2.6) to the functional case we should have an invertible covariance operator. The problem is that $\mathcal{K}$ is typically not invertible, in the sense that there is no linear and continuous operator $\mathcal{K}^{-1}$ such that $\mathcal{K}^{-1}\mathcal{K} = \mathcal{K}\mathcal{K}^{-1} = \mathbb{I}$, the identity operator. Indeed, when K is continuous, $\mathcal{K}$ is compact and this entails the non-invertibility.

A good solution to this problem is to change the space the data belong to. We move to the *Reproducing Kernel Hilbert Space* (RKHS)[2] associated with the covariance function K, i.e. $\mathcal{H}(K)$.

We can observe that the space $\mathcal{H}(K)$ is much smaller than $L^2[0, 1]$, and the trajectories of a stochastic process with continuous covariance function K do not belong with probability 1 to $\mathcal{H}(K)$. This implies that, if we want to compute the distance between two functions $x, y \in L^2[0, 1]$ with respect to the norm $\|\cdot\|_K$, we first need to approximate them in the space $\mathcal{H}(K)$ obtaining the functions x_α, y_α and only then we can proceed with the computation of the distance.

[2]To have more information about this space, refer to Appendix A.1.

How do we get this approximation?
The idea is to take x_α as the closest function to x in the space $\mathcal{H}(K)$, but this approach still fails since $\mathcal{H}(K)$ is dense in $L^2[0,1]$. An alternative is to look for x_α closest to the function x among those that are not too rough. In particular, the approximation x_α of a function $x \in L^2[0,1]$ in the space $\mathcal{H}(K)$ is uniquely defined as follows:

$$x_\alpha = \underset{f \in \mathcal{H}(K)}{\arg\min} \|x - f\|^2 + \alpha \|f\|_K^2 \tag{2.7}$$

where $\alpha > 0$ is a positive penalization parameter, with large values entailing a smoother curve x_α with respect to x. So in the end, while the term $\|x - f\|^2$ controls closeness to x, the term $\alpha\|f\|_K^2$ accounts for the roughness of the approximation.

Finally, considering all the assumptions that we have provided and given two functions $x, y \in L^2[0,1]$, the *α-Mahalanobi's distance* is defined as:

$$M_\alpha(x,y)^2 = \|x_\alpha - y_\alpha\|_K^2 \tag{2.8}$$

Exploiting the definitions of inner product and norm (A.2), we can rewrite both the expressions (2.7) and (2.8) as:

$$x_\alpha = (\mathcal{K} + \alpha\mathbb{I})^{-1}\mathcal{K}x = \sum_{j=1}^{\infty} \frac{\lambda_j}{(\lambda_j + \alpha)} \langle x, e_j \rangle e_j \tag{2.9}$$

$$M_\alpha(x,y)^2 = \sum_{j=1}^{\infty} \frac{\lambda_j}{(\lambda_j + \alpha)^2} \langle x - y, e_j \rangle^2 \tag{2.10}$$

The α-Mahalanobis distance has some interesting properties that we report in Appendix A.2.

2.3 New prior distribution: Pitman-Yor EPPF

The Pitman-Yor EPPF (Exchangeable Partition Probability Function) derives from the Pitman-Yor Process that is the most popular generalization of the Dirichlet Process[3].
It depends on two parameters σ, θ such that $\sigma \in [0,1)$ and $\theta > -\sigma$ and it is defined as:

$$\pi_{PY}(C, \sigma, \theta) = \mathbb{P}(\Psi_n = \{C_1, .., C_K\}) = \frac{\prod_{j=1}^{K-1}(\theta + j\sigma)}{(\theta+1)_{n-1}} \prod_{j=1}^{K} (1-\sigma)_{n_j - 1} \tag{2.11}$$

with $(b)_n = \frac{\Gamma(b+n)}{\Gamma(b)}$, for any integer $n \geq 0$, and n_j equal to the cardinality of the j-th cluster such that $\sum_{j=1}^{K} n_j = n$.

There are two main reasons leading to the choice of this distribution: on one hand, the potential to penalize small clusters through the parameter σ and, on the other hand, its independence from the number of clusters K. Indeed, the support of this prior distribution is given by the space of all possible partition of n observations into a certain number of groups, that is not fixed a priori.

2.4 Target of inference

In our GP-PPM framework, the target of inference is an optimal and unknown partition c_{opt} that could be get using a "Maximum a Posteriori" (MAP) estimation approach.

With uniform prior
c_{opt} is obtained as the partition c which maximizes the posterior with the constraint that the clusters number is fixed and equal to K. But due to the uniform prior assumption and (-λ) in front of the loss, this maximization problem is converted into the minimization of the loss function:

$$c_{opt} = \underset{c\ :\ |C|=K}{\arg\max} \pi(c|\lambda, X) = \underset{c\ :\ |C|=K}{\arg\min}\, \ell(c; X) = \underset{c\ :\ |C|=K}{\arg\min} \sum_{k=1}^{K} \sum_{i \in C_K} M_\alpha(x_i; X_k)$$

[3]To know more about PY and DP processes, refer to Appendix A.3.

With PY-EPPF prior
In this case, to obtain c_{opt} we need to maximize the posterior distribution, and to make it more readable we expand the Pitman-Yor EPPF prior as follows:

$$\pi_{PY}(C,\sigma,\theta) = \frac{\theta\Gamma(\theta)}{\Gamma(\theta+n)} \cdot \prod_{j=1}^{K-1}(\theta+j\sigma) \cdot \frac{1}{\Gamma(1-\sigma)} \cdot \prod_{j=1}^{K}\Gamma(n_j-\sigma)$$

Then, using the proportionality symbol we can delete all the constants that do not depend on the clusters, getting:

$$\begin{aligned}\pi(c|\lambda,\sigma,\theta,X) &\propto \pi_{PY}(c,\sigma,\theta)\cdot\exp\{-\lambda\sum_{k=1}^{K}\sum_{i\in C_K} M_\alpha(x_i;X_k)\}\\ &\propto \prod_{j=1}^{K-1}(\theta+j\sigma)\cdot\prod_{j=1}^{K}\Gamma(n_j-\sigma)\cdot\exp\{-\lambda\sum_{k=1}^{K}\sum_{i\in C_K} M_\alpha(x_i;X_k)\}\\ &\propto \frac{1}{(\theta+K\sigma)}\cdot\prod_{j=1}^{K}(\theta+j\sigma)\cdot\Gamma(n_j-\sigma)\cdot\exp\{-\lambda\sum_{k=1}^{K}\sum_{i\in C_K} M_\alpha(x_i;X_k)\}\end{aligned}$$

2.5 Gibbs sampling

An alternative to MAP estimation is represented by Gibbs sampling. The idea is to cyclically re-allocate the indicators c_i by sampling from their full-conditionals.
Let $\boldsymbol{c}_{-i} = (c_1,...,c_{i-1},c_{i+1},...,c_n)$ and $\{C_{1,-i},...,C_{K,-i}\}$ be the collection of cluster indicators and the associated partition, both without unit 'i'.

With uniform prior
As reported in [5], the full conditional for c_i reads:

$$\mathbb{P}(c_i=k|\boldsymbol{c}_{-i},-) \propto \exp\left[-\lambda\left[\sum_{j\in C_k}\mathcal{M}_\alpha(\boldsymbol{x}_j;\boldsymbol{X_k}) - \sum_{j\in C_{k,-i}}\mathcal{M}_\alpha(\boldsymbol{x}_j;\boldsymbol{X_{k,-i}})\right]\right] = \frac{\rho(C_k;\lambda,X_k)}{\rho(C_{k,-i};\lambda,X_{k,-i})}$$

where if an observation is the only element in a cluster, it can't be reallocated. This has an intuitive interpretation: the i-th unit is likely to be allocated in the k-th cluster if the cohesion of the newly created cluster is higher than the old cohesion.

With PY-EPPF prior
In this case the full conditional for c_i has a different expression whether the destination cluster 'k' is a "new" cluster or not:

$$\mathbb{P}(c_i=k|\boldsymbol{c}_{-i},-) \propto \begin{cases}(n_k-\sigma)\frac{\rho(C_k;\lambda,X_k)}{\rho(C_{k,-i};\lambda,X_{k,-i})}, & k=1,...,K\\ \frac{(\theta+k\sigma)}{\theta+n}, & k=K+1\end{cases}$$

We can notice that the likelihood degenerate to 1 for atomic cluster allocation. The contribution of the prior is coherent with the known behaviour of the EPPF for the Pitman-Yor process [4]. Even without having performed the sampling yet, this expression give us some concerns. The fact that for an atomic cluster the likelihood disappears can lead to troubles when comparing the case of a "new" cluster with the case of already existing clusters. This will require not only to have extra care when setting the parameters θ,σ but also to over-exploit the parameter λ: using it not only as weight of the prior belief with respect to the data information, but also as a coefficient to re-balance the numerical value assumed by the cohesion ratio(case existing cluster). For a future development could be significant the use of a modified likelihood, able to better accommodate the case of atomic cluster.

2.5.1 Interpretation of the MCMC chain

To interpret the samples of the MCMC chain when the sampled values are cluster labels can be a tricky business. We could be tempted to use estimator like the mode of the sampled values, but since the label switching is part of these sampling techniques, the results would be totally meaningless. Furthermore under the sight of decision theory those estimators could be (actually, they are) associated to inappropriate loss functions.

Alternative methods based on the posterior similarity matrix S have been proposed in the literature. S is a n by n matrix, with n sample size, representing the probability that two data points are in the same cluster.

In our analysis we exploited the estimator minimizing the Binder's loss. In the case the two types of errors are penalised equally, it is given by the formula:

$$B(\boldsymbol{c}, \hat{\boldsymbol{c}}) = \frac{1}{2}\left(\sum_{i=1}^{K} n_{i+}^2 + \sum_{j=1}^{\hat{K}} n_{+j}^2 - 2\sum_{i=1}^{K}\sum_{j=1}^{\hat{K}} n_{ij}^2\right),$$

where n_{ij} is the cardinality of the intersection between the set of data point indices in cluster i under $\boldsymbol{c}$ and the set of data point indices in cluster j under $\hat{\boldsymbol{c}}$, $n_{i+} = \sum_j n_{ij}$ and $n_{+j} = \sum_i n_{ij}$.
The optimal partition is the one that minimizes the posterior expected loss. Binder's loss prefers to split clusters over merging, for this reason also Variation of Information (VI) loss function is considered during the interpretation of the MCMC results.
Variation of information compares two clusterings with the information shared between them and is described by:

$$VI(\boldsymbol{c}, \hat{\boldsymbol{c}}) = H(\boldsymbol{c}) + H(\hat{\boldsymbol{c}}) - 2I(\boldsymbol{c}, \hat{\boldsymbol{c}}) = -\sum_{i=1}^{K} \frac{n_{i+}}{n} log\left(\frac{n_{i+}}{n}\right) - \sum_{j=1}^{\hat{K}} \frac{n_{+j}}{n} log\left(\frac{n_{+j}}{n}\right) - 2\sum_{i=1}^{K}\sum_{j=1}^{\hat{K}} \frac{n_{ij}}{n} log\left(\frac{n_{ij}n}{n_{i+}n_{+j}}\right),$$

where the first two terms represent the entropy of the two clusterings, and the last term is the mutual information between them.

Chapter 3

Clustering Algorithms

In this chapter we only report the structure of the main clustering algorithm implemented. The other commented pseudo-codes are reported in Appendix C.

3.1 Maximum a posteriori estimation (MAP)

We show three slightly different codes to get the optimal partition c_{opt} of the data applying the method reported in 2.4.
Using a compact notation, define:

- K, X and cov as the number of clusters, data and covariance matrix of X, respectively
- $c, \{m_k\}_k$ and ℓ as the label vector, list of centroids and loss value, respectively
- *toll* a suitable tolerance parameter (usually set equal to 10^{-2} or 10^{-6})

Clustering algorithm with uniform prior and fixed covariance structure

Algorithm 1: Mahalanobis distance clustering with fixed covariance

Input : K, cov, α, *toll*, X
Output: Optimal partition (c, $\{m_k\}_k$, ℓ)

Sample random observations as initial centroids $m_1, ..., m_K$;
Compute eigenvalues and eigenvectors of the covariance matrix;
Initialize $\ell_1(c; X)$ and $\ell_2(c; X)$;
while $|\ell_1(c; X) - \ell_2(c; X)| > toll$ **do**
  $\ell_1(c; X) = \ell_2(c; X)$
  for *i=1,...,n* **do**
    Set the cluster indicator c_i equal to k, so that $M_\alpha(x_i, m_k)$ is minimum
  for *k=1,...,K* **do**
    Set m_k as the functional mean of the observations belonging to cluster k
  Update $\ell_2(c; X)$

To get the optimal partition is sufficient to execute the algorithm just once.

Clustering algorithm with uniform prior and covariance updating within clusters

Algorithm 2: Mahalanobis distance clustering with covariance updating

Input : K, cov, α, *toll*, X
Output: Optimal partition (c, $\{m_k\}_k$, ℓ)

Sample random observations as initial centroids $m_1, ..., m_K$;
Compute eigenvalues and eigenvectors of the covariance matrix;
Initialize $\ell_1(c, X)$, $\ell_2(c, X)$ and $n_{iter} = 0$;
while $|\ell_1(c, X) - \ell_2(c, X)| > toll$ *or* $n_{iter} < 50$ **do**
  $n_{iter} += 1$;
  $\ell_1(c, X) = \ell_2(c, X)$;
  for *i=1,...,n* **do**
    Set the cluster indicator c_i equal to k, so that $M_\alpha(x_i, m_k)$ is minimum
  for *k=1,...,K* **do**
    Set m_k as the functional mean of the observations belonging to cluster k;
    Set cov_k as the covariance matrix of the k-th cluster and compute its eigenvalues and eigenvectors;
  Update $\ell_2(\boldsymbol{c}; \boldsymbol{X})$

The while loop condition is different from the one of Algorithm (1) due to the behaviour of the loss function once we perform the update of the covariances. Indeed, we found three possible trends:

- *Non-convergence*: This happens when the absolute value of the difference between the current and previous loss value stays the same at each iteration, that means that the algorithm continues to oscillate between the same two values with no possibility of converging to one of the two, being both above tolerance.
  To face this problem and to force the algorithm to exit the while loop, we introduce an iteration counter forcing the loop to break after 50 iterations, returning the partition currently obtained. Usually, it is not the correct one.
- *Convergence to a wrong partition*: the code exits the while loop after some iterations returning a partition that is not the right one and with a loss value that is not the smallest one.
- *Convergence to the right partition*: here the loss assumes the smallest value.

The reason of this behaviour is the stochasticity of the initial partitions: hence, to get the optimal result it is convenient to execute the algorithm several times and then extract the partition with the smallest loss value.

Clustering algorithm with PY-EPPF prior and fixed covariance

Algorithm 3: Mahalanobis distance clustering with PY-EPPF prior and fixed covariance

Input : K, α, σ, θ, λ, cov, X
Output: Optimal partition (c, $\{m_k\}_k$, ℓ_1, $post_1$)

Compute eigenvalues and eigenvectors of the covariance matrix;
Sample random observations as initial centroids $m_1, ..., m_K$;
Initialize $\ell_1(c, X)$ and $post_1(c, \ell_1, X)$, $\ell_2(c, X)$ and $post_2(c, \ell_2, X)$;

while $post_2(c, \ell_2, X) > post_1(c, \ell_1, X)$ **do**
  $\ell_1(c, X) = \ell_2(c, X)$;
  $post_1(c, \ell_1, X) = post_2(c, \ell_2, X)$;
  for *i=1,...,n* **do**
    Set the cluster indicator c_i equal to k, so that $M_\alpha(\mathbf{x}_i, \mathbf{m}_k)$ is minimum
  for *k=1,...,K* **do**
    Check that there are no empty clusters: otherwise, sample an observation and assign it to the empty k;
    Set m_k as the functional mean of the observations belonging to cluster k;
  Update $\ell_2(c, X)$ and $post_2(c, \ell_2, X)$

Due to the stochasticity of the initial centroids, it is not guaranteed to find the optimal partition at the first simulation; hence, it is recommended to execute Algorithm (3) for several simulations (called n_{simul}) and to return the best result get (i.e the simulation with the highest value of the posterior). In this way we are really sure to have sufficiently explored the space of the initial centroids.

Moreover, this prior has support on all the partitions of n elements where the number of clusters goes from 1 up to $\bar{K}$, that we choose. This means that to get the final partition over the n elements we have to execute the algorithm n_{simul} with the number of clusters K fixed and then change it in $\{1, ..., \bar{K}\}$. Summing up, the overall Algorithm is the following:

Algorithm 4: Mahalanobis distance clustering with PY-EPPF and fixed covariance (overall)

for *K=1,...,$\bar{K}$* **do**
  Run Algorithm (3) n_{simul} times with K clusters;
  Return the partition $\hat{c}_K$ with the highest posterior $\widehat{post}_K$;
Return (c_{opt}, $post_{opt}$, K_{opt}) with $post_{opt}$ highest among $\widehat{post}_K$

3.2 Gibbs sampling

The following pseudo-code is representative of both cases of PY-EPPF prior and uniform prior, with or without updating of the covariance matrix, which affects the computation of the α-Mahalanobis distance M_α. The difference in updating or not the covariance matrix affects the function we use to evaluate the cohesion $\rho(C_k; \lambda, X_k)$ and consequently how we sample c_i^t in the algorithm below.

Gibbs sampling for the allocation of the indices $\boldsymbol{c}_i = (c_1^i, ..., c_n^i)$

Algorithm 5: Gibbs sampling for $\boldsymbol{c}_i$

Input : N_{iter}, $N_{burn-in}$, c_0, X, λ, α, K
Output: Matrix $C = (c_{ij})_{ij}$, clust. allocation of obs j at iteration i

Initialize $t = 1$ and $p = 0$;
while $p < N_{iter}$ **do**
  for *i=1,...,n* **do**
    Sample c_i^t from its full-conditional, i.e. the allocation for obs i at iteration t
  if $t > N_{burn-in}$ **then**
    $p = t - N_{burn-in}$
    Insert $\boldsymbol{c}_t$ at p-th row of C
  t = t+1

Chapter 4

Simulated data

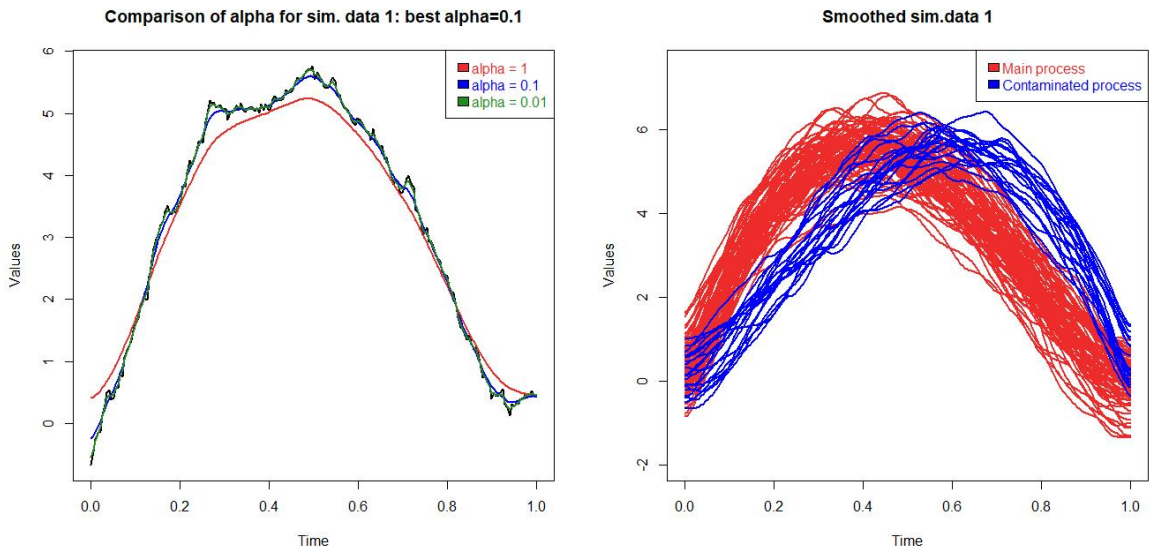
In order to test and tune our algorithms, we tested them on two simulated functional data before trying to apply on the real dataset we were provided. More information and plots about these data are reported in Appendix B.

Simulated data 1
The data are generated by the sum of a function with a noise resulting from a Gaussian Process with a certain mean and covariance function. In order to obtain two clusters two different functions are used.

Simulated data 2
In a similar manner, these simulated data are generated considering both two different functions (to have two clusters) and covariance matrices in the Gaussian process.

Figures (B.1a) and (B.2c) show respectively the first and the second type of data.

The role of the parameter α
We consider one function of the simulated data at a time and for each one we apply Algorithm (6) with several values of α: we chose the best one in order to avoid overfitting or underfitting with respect to the original function.
For all the functions of Simulated data 1 - and the same works for Simulated data 2 - we notice that we can use the same value of α and hence we decided to adopt $\alpha = 0.1$.
Figure (4.1a) shows the results get applying different values of α and Figure (4.1b) contains the functions once they have been smoothed by the best α.



(a) Smoothing with different α (b) Smoothed simulated data 1 with $\alpha = 0.1$

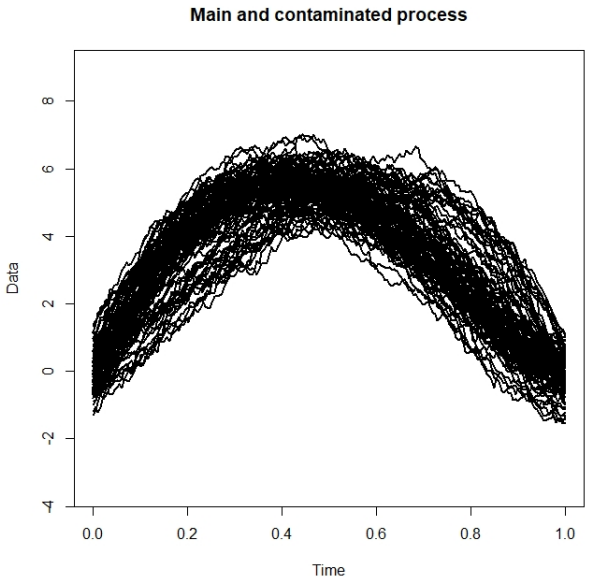
Figure 4.1: Simulated data 1 and the effect of α

Application of the algorithms

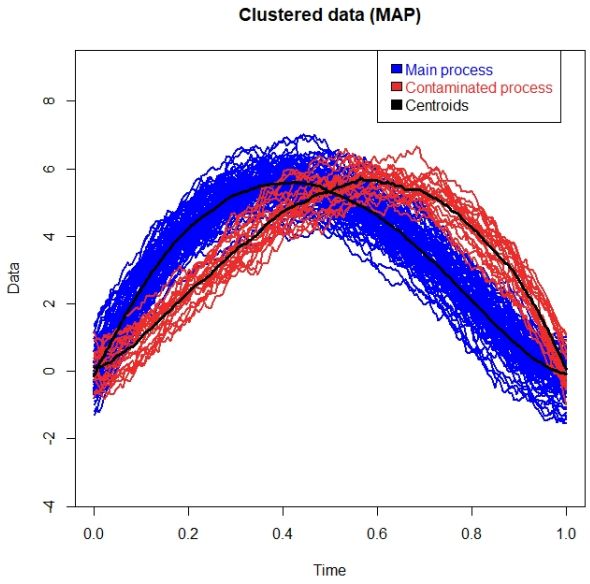
Firstly, we apply Algorithm (1) and (3) on Simulated data 1 since these data have a unique covariance matrix in the Gaussian process. The optimal partition found by each function is shown in Figure (4.2b) and (4.2c).

Then we apply Algorithms (2) on Simulated data 2, getting the clustering of Figure (4.3). To obtain this result several combination of the parameters (λ, σ and θ) can be used and some of them are reported in Appendix D.

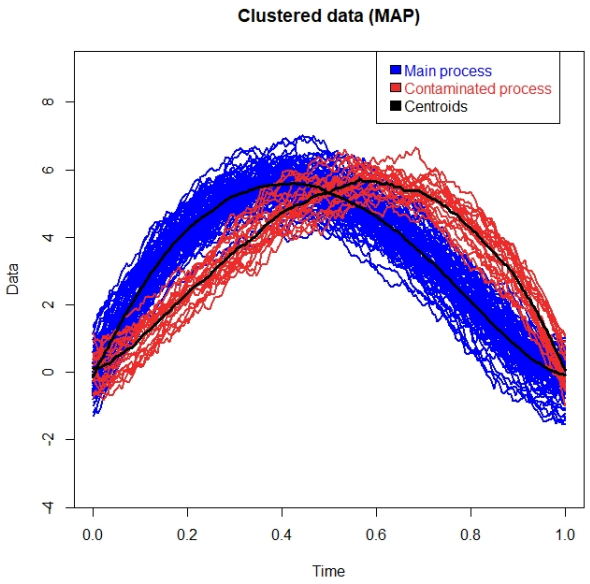
Comparing the results get here with the original plot, we can observe that the optimal partition is the correct one with all the algorithms. Of course we encounter the label switching phenomenon but at this stage it does not represent a problem.



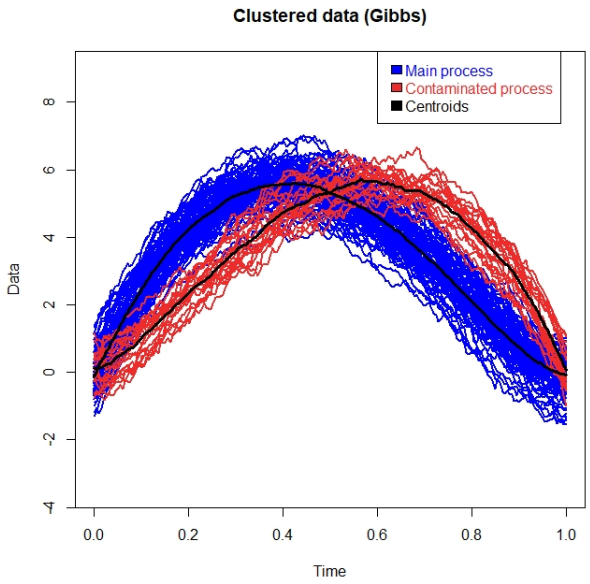
(a) Data before clustering



(b) Clustering with uniform prior and fixed covariance



(c) Clustering with PY-EPPF prior and fixed covariance



(d) Clustering after MCMC sampling

Figure 4.2: Clustering on Simulated data 1

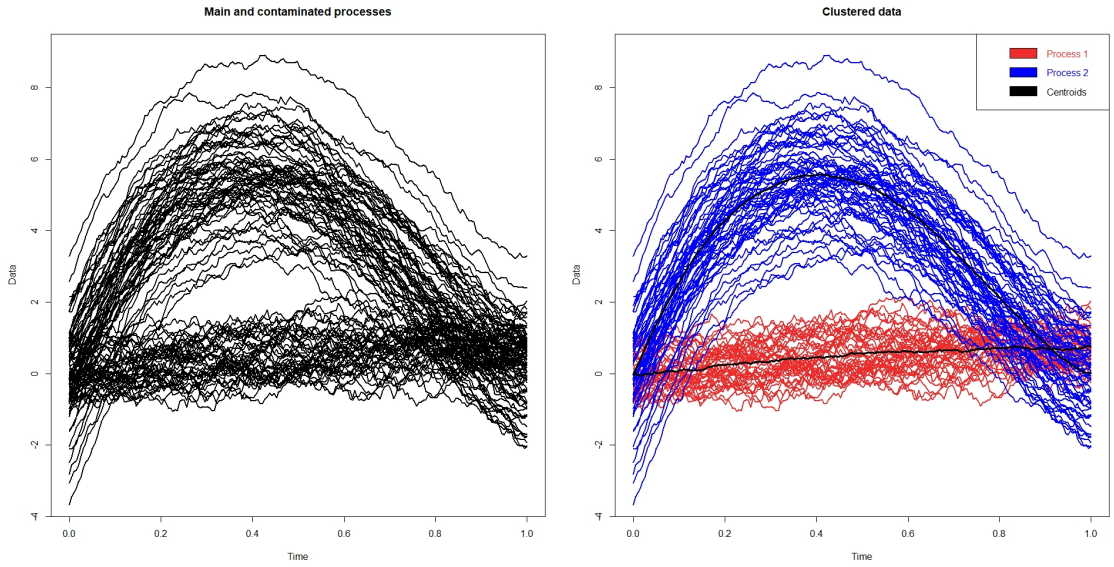
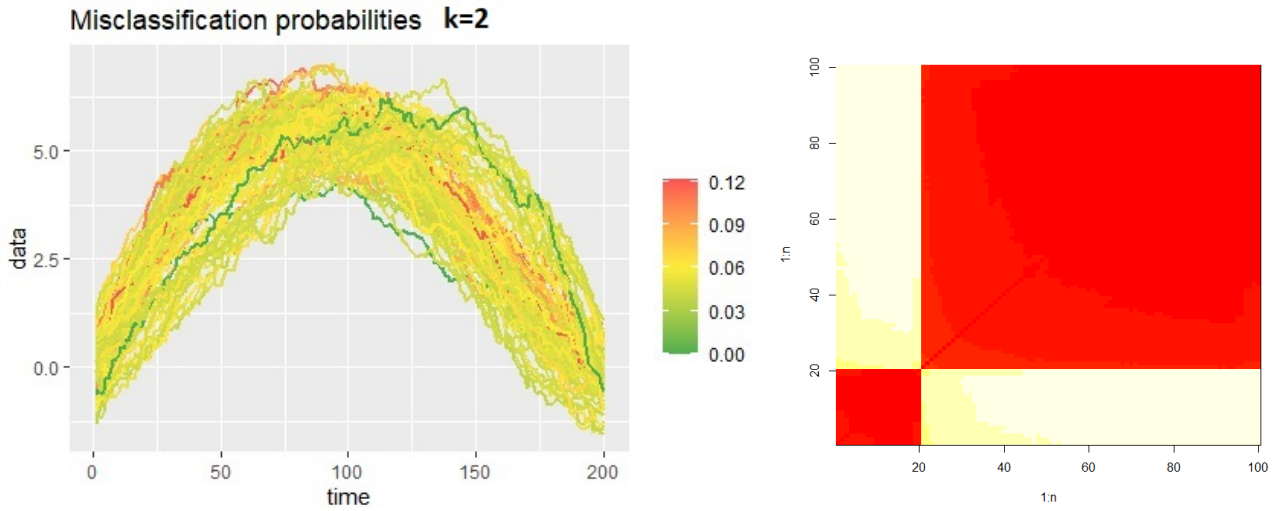


Figure 4.3: Clustering with uniform prior and covariance updating

The optimal partition is then given to the Gibbs sampling Algorithm (5) as initial value. We performed $N_{burn-in} = 800$ iterations and sampled the following $(N_{iter} - N_{burn-in}) = 8000$ values for the indexes $c_i = (c_1^i, ..., c_n^i)$.
In Figure (4.4b) we have the (co-clustering) similarity matrix and in Figure (4.2d) the binder cluster estimator. The similarity matrix is coherent with what expected and the cluster estimate identify perfectly the two originating processes.
In Table 4.1 we report also the evaluation of both posterior expected Binder's and Variation of Information loss function with K=2 clusters at the optimal partition.
A very readable summary of the information obtained from the Gibbs sampling is in Figure (4.4a), where each datum has a different color based on its misclassification probability. The misclassification probability is computed as $1 - s_{i,m}$, where $S = (s_{ij})_{ij}$ is the similarity matrix and observation m is the medoid of the relative cluster. The medoid is computed as the observation of the cluster with the smallest cumulative distance from other elements of the same cluster. The quantity $1 - s_{i,m}$ approximates the probability that the i-th unit is allocated to a cluster different from the one to which was assigned.



(a) Misclassification probability

(b) Similarity matrix

Figure 4.4: Simulated data 1

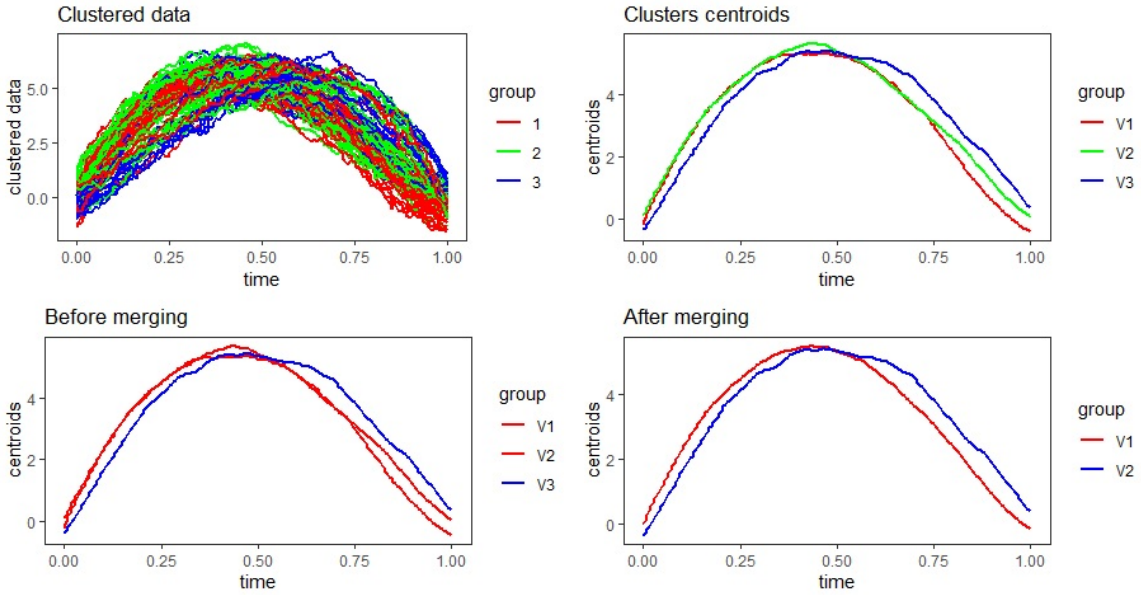
Loss	$\hat{K}$	$E[Loss \mid D]$
B	2	0.047
VI	2	0.132

Table 4.1: Greedy search algorithm with B and VI

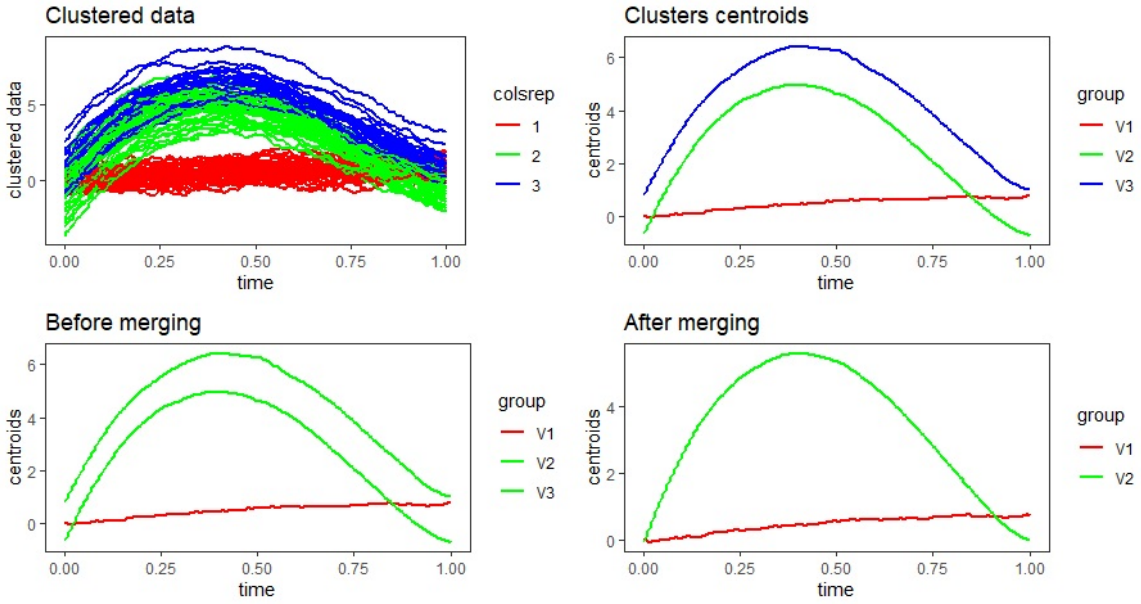
Union of similar clusters

A common problem in clustering algorithms is that it's often necessary to fix the number of clusters a priori. This is unfortunate since we may not know for sure the exact number of clusters that we want. In order to overcome this issue, we developed an additional algorithm that attempts to merge clusters that are too close to each other, in terms of both centroid and covariance structure. It is run after the previous clustering algorithms, taking their outputs as input, and returns the same output if no merging is needed, the merged clusters otherwise.

An example of application of this algorithm to simulated data is shown in Figure (4.5), and the details on how this algorithm works are shown in Appendix C, Algorithm (10).



(a) On Simulated data 1



(b) On Simulated data 2

Figure 4.5: The effect of a-posteriori clusters union

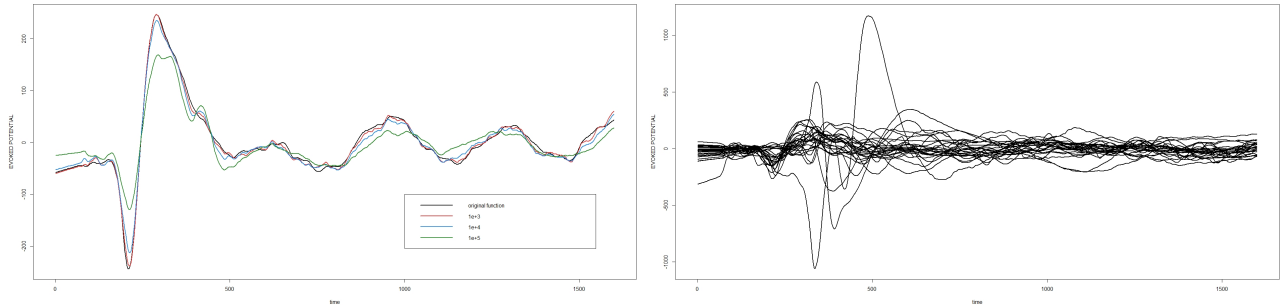
Chapter 5

Clinical data

5.1 Data Processing

Given the high density of data points at our disposal we first proceed by reducing them, keeping 1 out of 3 data point, obtaining 534 data points for each signal (from the original 1600).

The main issue we encountered with this clinical data is the extremely high variability (the highest eigenvalue is 2,472,250) this means that to obtain a nice smoothing of the original functions we have to opt for high values of the smoothing parameter α. After considering different values for the smoothing parameter (cf. Picture (5.1)) we set $\alpha = 10^4$.



(a) *Smoothing with different* α (b) *Smoothed data with* $\alpha = 10^4$

Figure 5.1: Selection of the smoothing parameter α

5.2 Clustering

We then performed each clustering algorithm presented in Chapter 3 on the new clean and smoothed data.

5.2.1 Clustering with uniform prior

MAP

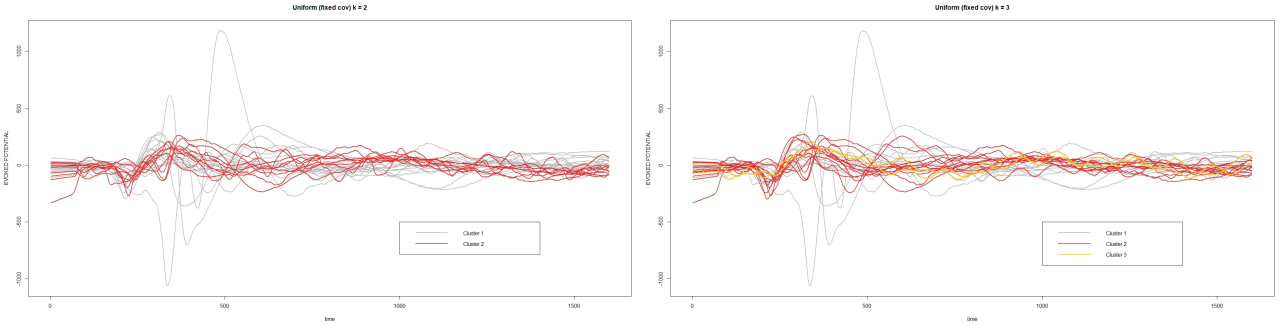
First we consider the GB-PPM with uniform prior over the cluster partition.
Both in case of a fixed covariance structure and covariance updating within clusters, respectively shown in Figure (5.2) and (5.3), we considered K = 2 as fixed number of clusters[1], since we noticed that in both cases increasing K results in a minimal decrease of the loss function associated to the partition (e.g. in case of fixed covariance structure from 48.1 to 47.8) due to the generation of single unit clusters among the clusters already identified with K = 2.
Moreover we get a cluster with two observations in case of a fixed covariance structure (cf. Figure (5.2)) and a single unit cluster in the case of updating cluster covariance (cf. Figure (5.3)).

[1]Recall that K must be fixed a priori

Between the two models we can see from Table (5.1) that the one that takes in consideration covariances updating within clusters has a slightly better performance in terms of the loss function evaluated with respect to the optimal partition with K = 2.

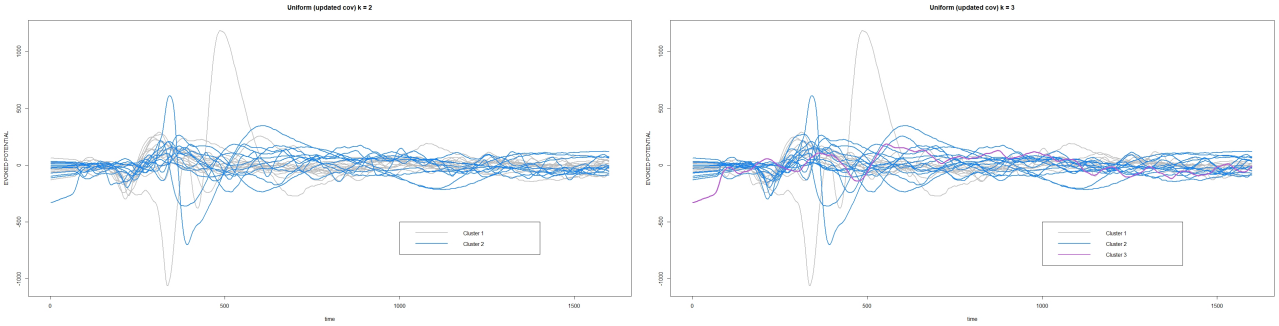
K=2 fixed	K=2 updated
48.1	46.3

Table 5.1: Loss values with uniform prior



(a) $K = 2$ (b) $K = 3$

Figure 5.2: Clustering with uniform prior and fixed covariance structure



(a) $K = 2$ (b) $K = 3$

Figure 5.3: Clustering with uniform prior and updating covariance within clusters

Gibbs Sampling

As in the previous data-example, we performed a Gibbs sampling starting from the optimal partition obtained with MAP. In this case we drew 5000 samples after a burn-in phase of 1000 iterations.
In Figure (5.5b) the co-clustering matrix is reported.
Using a greedy search algorithm, in the partition estimate found with Binder's loss the observations are divided into 11 clusters, whereas Variation of Information cannot detect any cluster (cf. Table (5.2)).
On the other hand, using the draw algorithm, which restricts the minimization problem to the MCMC output, we obtain a partition with K clusters, but we notice high misclassification probabilities.
In fact, even if the results seem to divide well data into clusters, as shown in Figure (5.4), we can clearly see how weak the estimate is since the co-clustering probabilities are almost indistinguishable one from another. Moreover, by plotting the data with coloring based on misclassification probabilities as shown in Figure (5.5a), we can see that they are all very similar (notice the scale), and close to the value of 0.5 for the case of $K = 2$ clusters, meaning probability of $1/K$ of choosing the correct cluster.

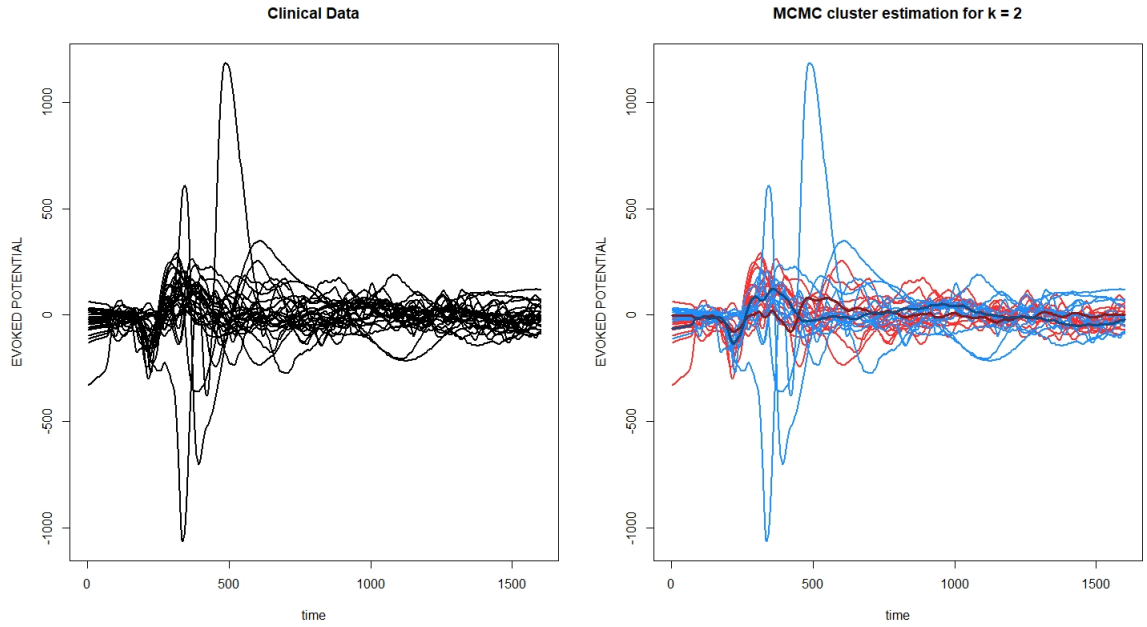
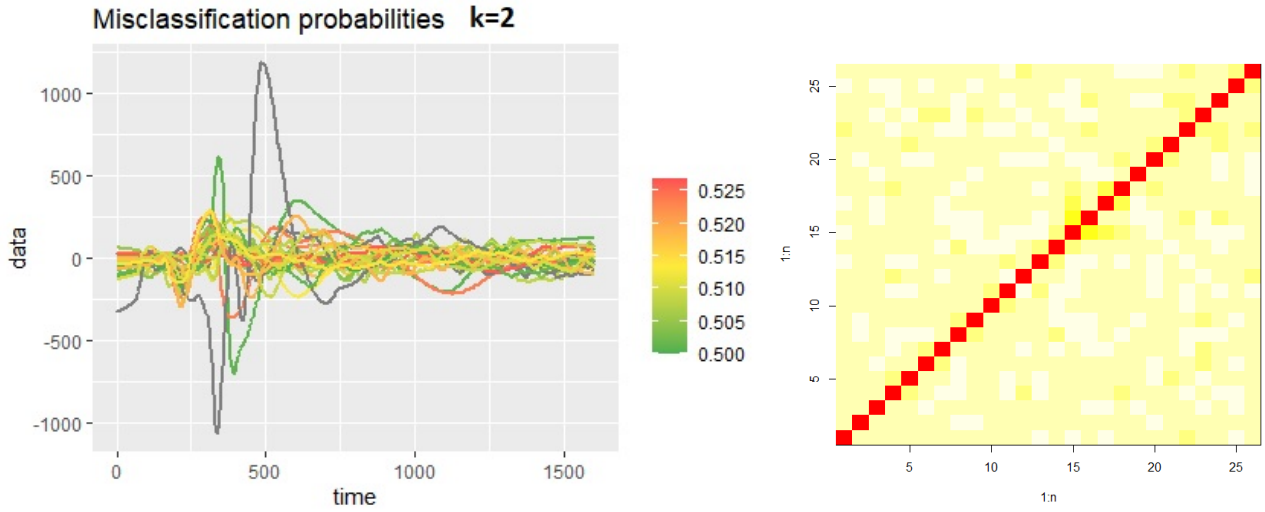


Figure 5.4: MCMC clustering with uniform prior and covariance updating

Loss	$\hat{K}$	$E[Loss \mid D]$
B	11	0.469
VI	1	0.970

Table 5.2: Greedy search algorithm with B and VI



(a) Misclassification probability for $\lambda = 1$

(b) Similarity matrix

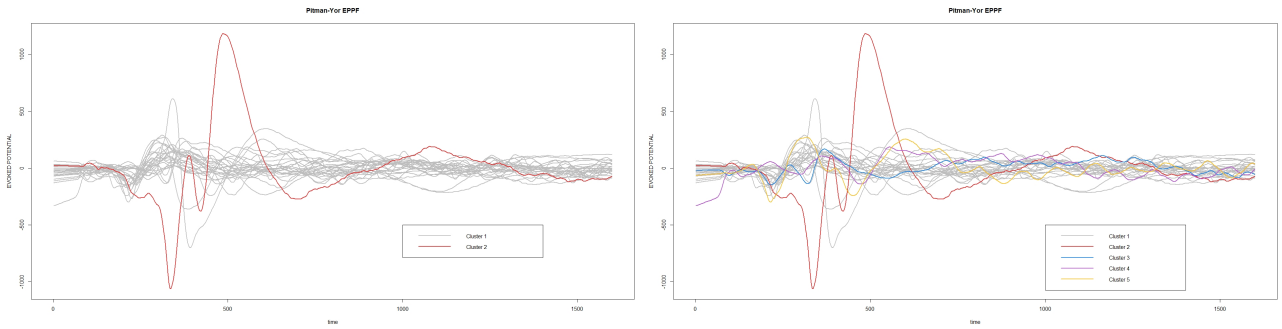
Figure 5.5: Clinical data

5.2.2 Clustering with PY-EPPF prior

MAP

The next step in our analysis is to consider the GB-PPM model with the Pitman-Yor EPPF prior. We considered different values of the parameters' model (λ, σ and θ) and we focused a lot on trying the algorithm with different values of σ, since we know that it should penalize small clusters. Finally, we set $\lambda = 0.65$, $\theta = 3.66$ and the maximum number of clusters $\bar{K} = 7$. Then we ran twice Algorithm (4) with $n_{simul} = 1000$, changing the value of σ from 0.75 to 0.5.

As we can see from Figure (5.6), in both cases we obtain one cluster containing most of the sig-



(a) $\sigma = 0.75$ (b) $\sigma = 0.5$

Figure 5.6: Clustering with PY-EPPF prior

nals and, depending on the value of σ, we get the number of single unit clusters decreasing along side the value of the discount parameter σ.
Overall this algorithm has not been able to perform an effective clustering of clinical data. In particular, for all the electrodes no clustering structure is detected, except for the isolation of very dissimilar functions in shape and amplitude.

Gibbs Sampling
We performed 10000 MCMC iterations after a burn-in phase of 1000 samples. The Similarity matrix is reported in Figure (5.7). We tried various combinations of the hyperparameters but the results never reported clearly interpretable partitions. In particular, as a known tendency of the 2 estimators, depending on the different configurations of λ, θ, σ, the Binder estimate reported a number of clusters between 11 and 15 and VI constantly reported no cluster distinction (i.e. just one cluster for all the data). Given these results we deemed the computation of the misclassification probabilities as not necessary.

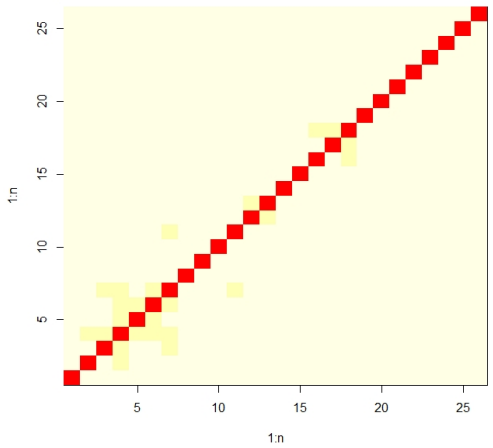


Figure 5.7: Similarity matrix for clinical data with PY-EPPF prior

5.3 Conclusions

Among the three different tested algorithms, the GB-PPM with uniform prior and updating covariance within clusters seems to be the best one for clustering the clinical data at our disposal.

None of the models presented above proved correlation with respect to the clinical evaluations of the patients in all of the four components: we argue that this may be caused by the extremely high variability of the functional clinical data which plays an overpowering role in our models.

For a better understanding of this clinical data, a possible research path may be in the choice of a different functional distance that isn't as highly influenced by the covariance structure of the data and/or a more flexible variation of the likelihood term.

Bibliography

[1] José R. Berrendero, Beatriz Bueno-Larraz, and Antonio Cuevas. "On Mahalanobis Distance in Functional Settings". In: *Journal of Machine Learning Research* 21 (2020), pp. 1–33.

[2] Antonio Canale et al. "On the Pitman–Yor process with spike and slab prior specification". In: (2017).

[3] Markus Herdin et al. "Correlation Matrix Distance, a Meaningful Measure for Evaluation of Non-Stationary MIMO Channels". In: (2005), p. 2.

[4] J. Pitman. "Exchangeable and partially exchangeable random partitions". In: *Probab. Theory and Relat. Fields* 102 (1995), pp. 145–158.

[5] Tommaso Rigon, Amy H. Herring, and David B. Dunson. "A generalized Bayes framework for probabilistic clustering". In: *arXiv:2006.05451* (2020).

[6] Sara Wade and Zoubin Ghahramani. "Bayesian Cluster Analysis: Point Estimation and Credible Balls (with Discussion)". In: *Bayesian Anal.* 13 (2) (2018), pp. 559–626.

Codes

All the analysis were implemented in
`R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.` https://www.R-project.org/.

Codes are publicly available at this GitHub Repository:
https://github.com/PietroSpina96/BayesianProject

Appendix A

Useful theoretical concepts

A.1 Reproducing Kernel Hilbert Space

Introduce the *Reproducing Kernel Hilbert Space* (RKHS) associated with the covariance function K:

$$\mathcal{H}(K) := \mathcal{K}^{1/2}(L^2[0,1]) = \left\{ f \in L^2[0,1] : \sum_{j=1}^{\infty} \frac{\langle f, e_j \rangle^2}{\lambda_j} < \infty \right\} \tag{A.1}$$

and the corresponding inner product and norm, given two functions f,g:

$$\langle f, g \rangle_K = \sum_{j=1}^{\infty} \frac{\langle f, e_j \rangle \langle g, e_j \rangle}{\lambda_j} \qquad\qquad \|f\|_K^2 = \sum_{j=1}^{\infty} \frac{\langle f, e_j \rangle^2}{\lambda_j} \tag{A.2}$$

where (e_j, λ_j) are the eigenfunctions and eigenvalues of the covariance operator $\mathcal{K}$.

First of all, we highlight that in the finite dimensional case, the RKHS associated with a non-singular covariance matrix coincides with R^d, and the square norm of a vector $x \in R^d$ is $\|x\|_\Sigma^2 = \sum_{j=1}^{p} \frac{(x' e_j)^2}{\lambda_j}$. Therefore, we can reformulate the Mahalanobis distance (2.6) in the discrete case using the square norm of the RKHS, that is:

$$M(x,y)^2 = \|x - y\|_\Sigma^2 \tag{A.3}$$

A.2 Properties of the α-Mahalanobis distance

Here we report the properties of the α-Mahalanobis distance explained in section 2.2:

- It defines a metric in $L^2[0,1]$, that is, M_α aims at measuring either the statistical distance between two observations $x = x(t)$ and $y = y(t)$ of a process with covariance operator $\mathcal{K}$, or the distance between a trajectory $x = x(t)$ and the mean function $m = m(t)$.

- It is continuous and decreasing with respect to α.
  The idea is that: when considering two different function x, y and the value of α is very large, the approximation x_α, y_α are very smoothed and far away from the original functions: hence, their distance is small.

- It is invariant with respect to operator preserving the norm (isometries).
  We recall that a bounded and linear operator L is an isometry if it maps $L^2[0,1]$ into $L^2[0,1]$ and $\|f\| = \|Lf\|$. Thus, $M_\alpha(x,y) = M_\alpha(Lx, Ly)$.

One really useful property from a practical point of view is the following: there exist an estimator for the α-Mahalanobis distance. In particular, given a stochastic process $X(t)$ with mean m and given a sample of n observations $X_1(t), ..., X_n(t)$ drawn from $X(t)$, define:

- The sample mean $\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i(t)$

- The sample covariance function $\widehat{K} = \frac{1}{n}\sum_{i=1}^{n}(X_i(s) - \bar{X}(s))(X_i(t) - \bar{X}(t))$
- The sample covariance operator $\widehat{\mathcal{K}}$ such that $\widehat{\mathcal{K}}f(\cdot) = \int_0^1 \widehat{K}(\cdot, t)f(t)dt$

the estimator for $M_\alpha(X, m)$ has the form:

$$\widehat{M}_{\alpha,n}(X, \bar{X}) := \|\widehat{X}_\alpha - \bar{X}_\alpha\|_{\hat{K}} \tag{A.4}$$

where $\widehat{X}_\alpha$ and $\bar{X}_\alpha$ are approximations of X and $\bar{X}$ in $\mathcal{H}(\hat{K})$. They can be easily obtained from (2.9), substituting $\mathcal{K}$ and (e_j, λ_j) with the empirical ones, for instance: $\widehat{X}_\alpha = (\widehat{\mathcal{K}} + \alpha\mathbb{I})^{-1}\widehat{\mathcal{K}}X$.

We have some convergence results:

- $\|\bar{X} - m\| \to 0 \quad$ as $n \to \infty$
- $\|\widehat{K} - K\|_{L^2([0,1]x[0,1])} \to 0 \quad$ as $n \to \infty$
- $\|\widehat{\mathcal{K}} - \mathcal{K}\|_{op} \to 0 \quad$ as $n \to \infty$
- $\widehat{M}_{\alpha,n}(f, \bar{X}) \xrightarrow{\text{n}} M_\alpha(f, m) \quad$ as $n \to \infty \quad \forall f \in \mathbb{L}^2[0, 1]$
  with order of convergence $O_P(n^{-1/4})$, where O_P stands for convergence order in probability.

A.3 Pitman-Yor process

The Pitman-Yor process is the most popular generalization of the Dirichlet Process (DP).

Consider the following model for the data: $X_1, X_2, ..., X_n|P \sim P$ with $P \sim \text{DP}(a, \alpha_0)$. Due to the discreteness of P the sample $X_1, X_2, ..., X_n$ induces a partition Ψ_n on $\{1, .., n\}$ such that i, j belong to the same cluster C_l, $l \leq n$ if and only if $X_i = X_j$. The corresponding probability distribution $\mathbb{P}(\Psi_n = \{C_1, .., C_K\})$ is called "exchangeable partition probability function" (EPPF).

P is a Pitman-Yor process, $P \sim \text{PY}(\sigma, \theta; \alpha_0)$ with $\sigma \in [0, 1)$ and $\theta > -\sigma$ if P is built according to the "Stick-Breaking Construction" with $V_i \sim \text{Beta}(1 - \sigma, \theta + i\sigma)$[1].

[1] If we consider $\sigma = 0$ we recover the DP.

Appendix B

Simulated Data

Simulated Data 1
Here we report in more details the representation of the simulated data with a brief analysis. Simulated Data 1 are generated as follows:

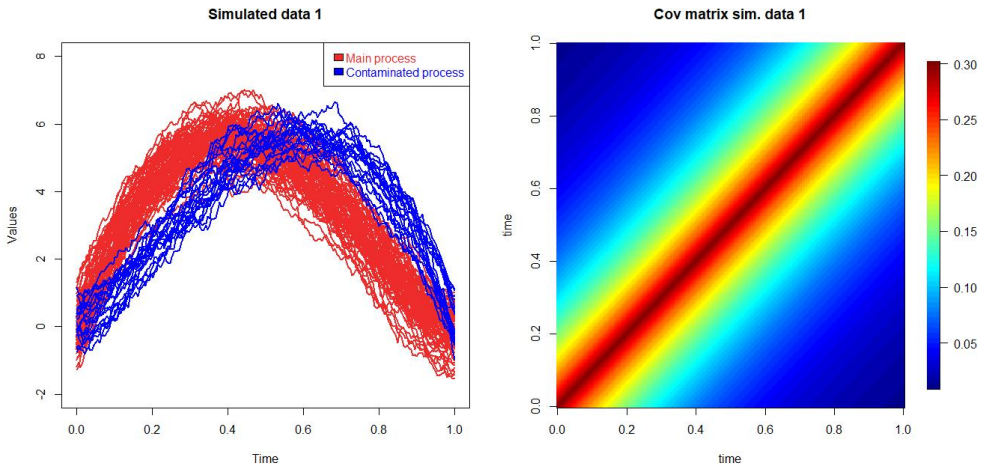
- Main process: $X(t) = 30t(1-t)^{3/2} + \epsilon(t) \quad t \in [0,1]$
- Contaminated process: $X(t) = 30t^{3/2}(1-t) + \epsilon(t) \quad t \in [0,1]$

with $\epsilon(t) \sim \text{GP}(0, \mathcal{C})$ and $\mathcal{C}(s,t) = 0.3 \cdot exp(-|s-t|/0.3)$

These data are characterized by:

◇ Two different functions $X(t)$ that lead to two clusters (cf. Figure (B.1,a)) and a single covariance matrix corresponding to $\mathcal{C}(s,t)$, that does not depend on the clusters (cf. Figure (B.1,b))

◇ A Gaussian Process that adds some irregularity to the data, since real data often contains some kind of noise. To deal with this problem, we recall the α-Mahalanobis approximation of a generic function using the smoothing parameter α (cf. section 2.2)



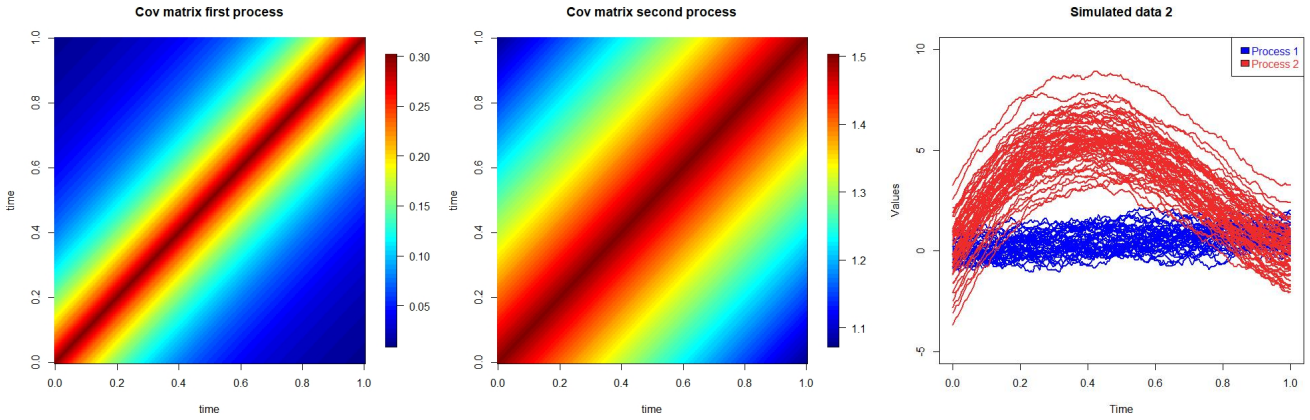
(a) *Data clustered in two groups* (b) *Covariance matrix* $\mathcal{C}$

Figure B.1: Simulated data 1

Simulated Data 2
Simulated data 2 are generated as follows:

- First process: $X(t) = sin(t) + \epsilon_1(t)$
  with $\epsilon_1(t) \sim \text{GP}(0, \mathcal{C}_1) \quad \mathcal{C}_1(s,t) = 0.3 \cdot exp(-|s-t|/0.3) \quad t \in [0,1]$
- Second process: $X(t) = 30t(1-t)^{3/2} + \epsilon_2(t)$
  with $\epsilon_2(t) \sim \text{GP}(0, \mathcal{C}_2) \quad \mathcal{C}_2(s,t) = 1.5 \cdot exp(-|s-t|/3) \quad t \in [0,1]$

They are characterized by two Gaussian Processes that add some irregularity to the data, but now are such that the covariance matrices are different. In particular, the covariance of the first process is different from zero in a narrow diagonal band and the one of the second process is different from zero in a larger diagonal band, as one can see in the Figures (B.2,a) and (B.2,b).



(a) *Covariance matrix* $\mathcal{C}_1$ (b) *Covariance matrix* $\mathcal{C}_2$ (c) *Data clustered in two groups*

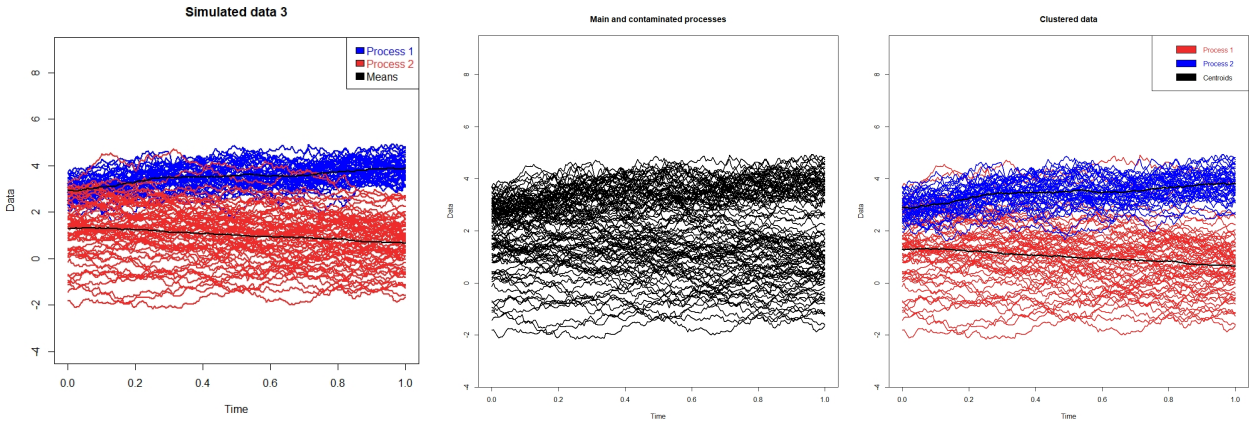
Figure B.2: Simulated data 2

Simulated data 3

In order to test the effectiveness of Algorithm (2) (i.e. the capacity to detect clusters with respect to their covariance matrices), we created the simulated data reported in Figure (B.3,a)) with the requirement that: the cluster means were close and the covariances were different[1].
In detail:

- First process: $X(t) = sin(t) + 3 + \epsilon_1(t)$
  with $\epsilon_1(t) \sim \text{GP}(0, \mathcal{C}_1) \quad \mathcal{C}_1(s,t) = 0.3 \cdot exp(-|s-t|/0.3) \quad t \in [0,1]$
- Second process: $X(t) = cos(t) + \epsilon_2(t)$
  with $\epsilon_2(t) \sim \text{GP}(0, \mathcal{C}_1) \quad \mathcal{C}_1(s,t) = 1.5 \cdot exp(-|s-t|/3) \quad t \in [0,1]$

Then we applied the algorithm and we get the optimal partition shown in Figure (B.3,b). We can observe that some observations (4 up to 100) have been wrongly classified, but this is the best result that we get, in terms of minimum loss, after lots of simulations. Indeed, since the clusters are not well separated, our algorithm often does not converge to a result.



(a) *Two overlapped clusters* (b) *Data clustered in two groups*

Figure B.3: Simulated data 3

[1]The covariance matrices are the same of Simulated Data 2.

We have also applied Algorithm (10) on these data and we get the result shown in Figure (B.4): only 5 observations (up to 100) that have been misclassified.

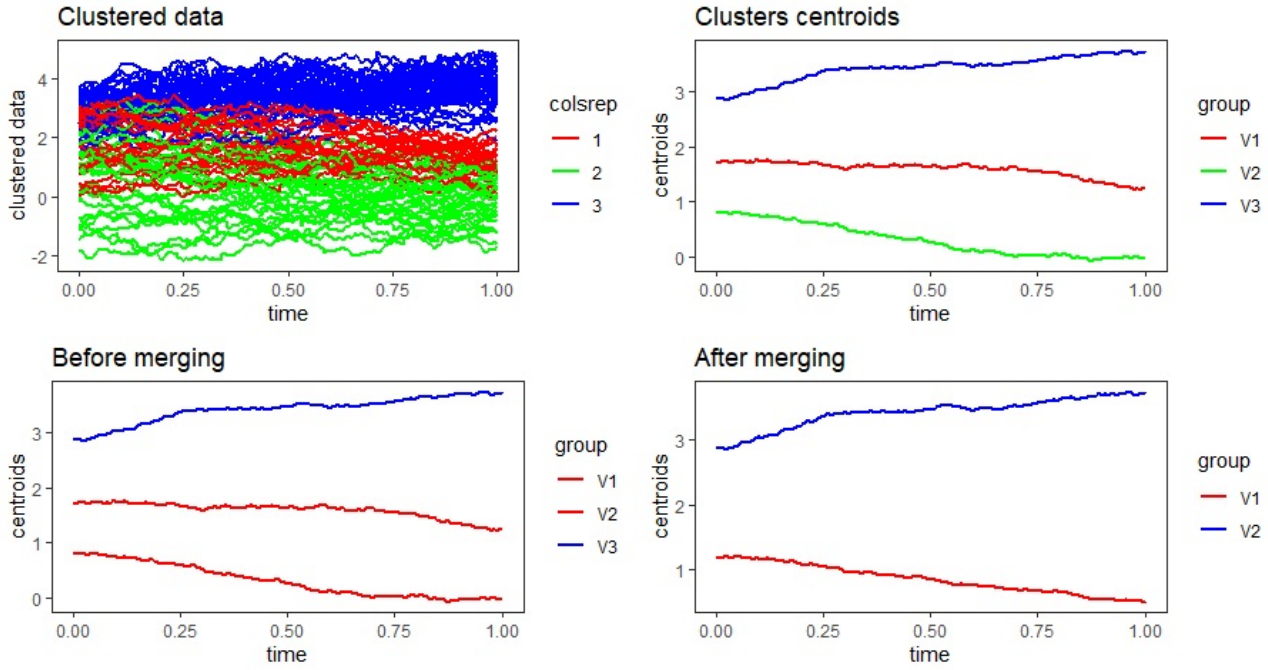


Figure B.4: Union of similar clusters on Simulated data 3

Appendix C

Additional algorithms

C.1 α-Mahalanobis approximation and distance function

Let $x \in L^2[0,1]$ be a generic function, $\alpha > 0$ the smoothing parameter, (λ_j,e_j) the j-th eigenvalue and eigenfunction of the data covariance matrix ($\forall j$).

From a practical point of view, since our functions are defined by a finite number of points, the cardinality of the set of eigenvalues-eigenfunctions is finite as well and equal to the number of points in which the domain of the function has been discretized and that we call T.
From now on, for simplicity of notation, every time we write $\{(\lambda_j,\, e_j)\}_j$ we simply mean the list of all (λ_j,e_j) $\forall j = 1, ..., T$

The algorithm for the "α-Mahalanobis approximation" of x just computes this functional approximation as explained in section 2.2 and it works as shown in Algorithm (6) and the "α-Mahalanobis distance" between two generic functions x and y with a discretized domain works as in Algorithm (7).

Algorithm 6: α-Mahalanobis approximation function

Input : x, α, $\{(\lambda_j,\, e_j)\}_j$
Output: α-approximation x_α of x
Compute: $x_\alpha = \sum_{j=1}^{T} \frac{\lambda_j}{(\lambda_j+\alpha)} \langle x, e_j \rangle e_j$

Algorithm 7: α-Mahalanobis distance function

Input : x, y, α, $\{\lambda_j,\, e_j\}_j$
Output: Distance $M_\alpha(x, y)$
Compute: $M_\alpha(x,y) = \sqrt{\sum_{j=1}^{T} \frac{\lambda_j}{(\lambda_j+\alpha)^2} \langle x - y, e_j \rangle^2}$

C.2 Gibbs Loss function

In order to obtain the optimal partition c_{opt}, a key implementation is the "loss function" that computes the loss as $\ell(c; X) = \sum_{k=1}^{K} \sum_{i \in C_K} M_\alpha(x_i; m_k)$. There are two different versions of this algorithm according to the choice of keeping fixed the covariance matrix of the data or to change it depending on the clusters.

With fixed covariance
Since there is no updating of the covariance matrix, the eigen-decomposition is passed as input in Algorithm (8) .

Algorithm 8: Gibbs loss function with fixed covariance

Input : K, $\{m_k\}_k$, c, $\{(\lambda_j, e_j)\}_j$, X
Output: Loss value ℓ
Compute: $\ell(c; X) = \sum_{k=1}^{K} \sum_{i \in C_K} M_\alpha(x_i; m_k)$

With covariance updating
On the other hand, after the computation of every cluster covariance matrix cov_k $\forall k$, we build two new matrices that are passed as input in Algorithm (9):

- Λ that contains the eigenvalues of cov_k, $\forall k$
- E that contains the eigenfunctions of cov_k, $\forall k$

Algorithm 9: Gibbs loss function with covariance updating

Input : K, $\{m_k\}_k$, c, Λ, E, X
Output: Loss value ℓ

for *k=1,...,K* **do**
  Extract $\{(\lambda_{j,k}, e_{j,k})\}_j$ of the k-th cluster from Λ, E
  for $i \in C_k$ **do**
    Compute $M_\alpha(x_i; m_k)$ using the k-th eigen-decomposition
Set $\ell(c; X)$ as the sum of all the distances $M_\alpha(x_i; m_k)$

C.3 Mahalanobis distance clustering algorithm with uniform prior

With fixed covariance
First of all, we define the centroid of each cluster as the sample mean of the observations that belong to that cluster. Of course, if there is just one unit in a cluster the centroid corresponds to that unit and if it is empty we decide to fix it equal to the mean of the whole set of data.

Due to the structure of the loss function, the main idea behind its minimization is to set the cluster label c_i of the i-th unit equal to k (k-th cluster) so that the distance between the unit x_i and the centroid m_k is minimized. This procedure is then repeated for all the units until the centroids stabilize, that means until the value of the loss function is stable and doesn't change any more.

To get the optimal partition, we continue to update the allocation of the units to the clusters and, consequently, the value of the loss function until the absolute value of the difference between its actual value and the previous one is smaller than a fixed tolerance (i.e. until the centroids stabilize). Finally, we return the last partition found and the corresponding loss.

With covariance updating
With respect to the previous version, here we pass the covariance matrix of the data and once we assign each unit to a cluster (according to the minimum distance concept), we compute the covariance matrix of the k-th cluster of data ($\forall$ cluster), so that when we have to calculate the distance between the i-th unit and the k-th centroid, we will use the eigen-decomposition of the k-th cluster ignoring all the others.

C.4 Union of similar clusters

The Algorithm (10) checks the cluster centroids, and if it founds them to be close to each other, it then checks their within-cluster covariances to see if they're also similar. The centroids distances are computed using Euclidean distance, while the cluster covariances are compared through a kind of cosine similarity for matrices as inspired from [3]. The thresholds to determine closeness are automatically set to specific values based on various tests performed on different datasets.

Algorithm 10: Union of similar clusters

Input : Optimal centroids $\mathbf{m}_1, ..., \mathbf{m}_k$, X
Output: United centroids where needed

Compute centroid distances matrix d : $d_{ij} = \|\mathbf{m}_i - \mathbf{m}_j\|_2^2 \quad \forall i, j$;
Set $\varepsilon = 0.5 \cdot (\text{median}(d) + \text{mean}(d))$;
if $d_{ij} < \varepsilon$ **then**
  Compute cluster covariances and their distance: $d_{cov} = 1 - \frac{\text{tr}\{C_i C_j\}}{||C_i||_f ||C_j||_f}$;
  if $d_{cov} < 0.05$ **then**
    Merge clusters i, j;
Recompute the centroids and repeat until $d_{i_{new} j_{new}} > \varepsilon_{new} \quad \forall i_{new}, j_{new}$

C.5 Mahalanobis distance clustering algorithm with PY-EPPF prior

Due to the change of prior distribution, we adapt our Mahalanobis distance clustering algorithm in order to find the optimal partition c_{opt}. In detail, we need to build a new function (cf. Algorithm (11)) that computes the value of the posterior distribution according to the formula reported in subsection 2.4.

Algorithm 11: Pitman-Yor posterior

Input : K, σ, θ, λ, c, $loss$, X
Output: Pitman-Yor posterior

Count the observations in each cluster: $\{n_j\}_j$
Compute: $post = \frac{1}{(\theta + K\sigma)} \prod_{j=1}^{K} (\theta + j\sigma)\Gamma(n_j - \sigma) \cdot \exp\{-\lambda \sum_{k=1}^{K} \sum_{i \in C_K} M_\alpha(x_i; X_k)\}$

Since we want to maximize the value of this posterior, every time we update the vector of labels we check if the associated posterior is higher than the previous one and, if not, we return the highest one just found (cf. while loop condition).

With reference to Algorithm (4), we also thought to implement this costly procedure in such a way that Algorithm (3) could have the number of clusters not fixed a priori and could update it at each iteration. The problem here observed is that, after few iterations, all the units tend to be clustered in a single group leaving the others empty. Of course, this situation is not acceptable and to forbid it we impose the non-emptiness of each cluster: once all the units have been assigned to the groups we check that these are non-empty: if not, we sample one observation from the data and impose that it belongs to the empty cluster. We repeat until all the clusters have at least one unit.

A possible development to solve the degenerate case of all the units in a single cluster is to introduce a penalization term that decreases the value of the posterior distribution once we note that the majority of the units belongs to just one cluster. In this way the posterior has a smaller value and the algorithm will continue to look for the highest posterior, ignoring the one just found.

With covariance updating
As we did with the uniform prior, we tried to implement a further version of Algorithm (3) where at each iteration we compute the covariance matrix of the k-th cluster of data (∀ cluster).

Unfortunately, we had lots of problems caused by the oscillating and non-monotone behaviour of the value of the posterior at each iteration. To solve this problem we tried to re-use the while loop condition of Algorithm (2), but none of the results was the correct one. This forced us to put aside the implementation of the Clustering function with covariance updating.

Appendix D

Parameters calibration

With uniform prior
As mentioned in section 2.4, to get c_{opt} we can equivalently minimize the loss function instead of maximize the posterior distribution that depends on the parameter λ. This equivalence implies that we are not obliged to tune λ since it does not appear in our code.

With PY-EPPF prior
On the other hand, with the change of prior distribution we have to tune not only λ but also the other two parameters of the PY-EPPF, that are σ and θ. We do not assume for them a prior distribution but we just considered them fixed.
To find their best values or a range of optimal values, we execute Algorithm (4) several times on Simulated data 1 (with $n_{simul} = 100$), changing one at the time the value of λ, σ and θ. At each repetition, we adopt the following criterion: we accept a value if the resulting partition has two clusters (since we know that simulated data have two).

The list of the simulations with the results (values of the posterior with different clusters) is reported below. We can deduce that even if we impose a specific rule to accept/reject a result, it seems that there not exist ranges of parameters that respect this rule, in the sense that we just found single and different values. To conclude, the calibration of the parameters is data-dependent and it does not seem to follow a precise rule so that we need to tune them manually.

In the first table, λ varies with: $\sigma = 0.25$, $\alpha = 0.1$ and $\theta = 3.66$.
In the second one, σ varies with: $\lambda = 0.75$, $\alpha = 0.10$ and $\theta = 3.66$
In the last one, θ varies with: $\sigma = 0.50$, $\alpha = 0.10$ and $\lambda = 0.75$

λ	K=1	K=2	K=3
0.10	7.14e+134	4.34e+131	5.03e+115
0.25	7.27e-107	2.44e-40	6.53e-39
0.70	1.42e+11	4.17e+13	5.04e+11
0.73	9.27e+04	2.94e+08	7.87e+8
0.75	6.98	1.08e+05	2.63e+04
0.80	3.43e-10	2.80e-04	4.58e-07

σ	K=1	K=2	K=3
0.45	2.78	2.63e+04	5.92e+02
0.50	2.21	1.84e+04	3.83e+05
0.75	7.01e-01	3.15e+03	3.00
0.80	5.57e-01	2.21e+03	7.00e+02
0.90	3.52e-01	1.09e+03	3.93e+04

θ	K=1	K=2	K=3
-0.4	2.21	4.43e+02	2.37e+02
0	2.21	2.22e+03	2.14e+03
3.66	2.21	1.84e+04	3.83e+05
4	2.21	2.00e+04	8.32e+04
8	2.21	3.77e+04	1.90e+04