Non-Parametric Multivariate Time Series Co-clustering Model Applied to Driving-Assistance Systems Validation

No Author Given

No Institute Given

Abstract. In this paper, we propose a new Bayesian co-clustering approach applied to Multivariate time series. Our methodology of Functional Non-Parametric Latent Block Model (FunNPLBM) simultaneously creates a partition of observation and a partition of temporal variables, using latent multivariate gaussian block distributions. We propose to use a bi-dimensional Dirichlet Process as a prior for the block distributions parameters and for block proportions, which natively provides model selection. This approach is benchmarked and studied on a simulated dataset and applied to an advanced driver-assistance system validation use-case.

Keywords: Bayesian Non Parametric · Co-clustering · Model-Based Clustering · Multivariate Time Series · Driving-Assistance Systems

1 Introduction

Unsupervised classification, or *clustering*, is a first approach to dataset exploration that consists in the automatic grouping of similar observations into homogeneous groups, without supervision, i.e., without labels. Time series clustering is crucial to decision-making in many domains (Industry, Health, Finance, Biology,...) and has been extensively studied in the literature [1, 2].

In a multivariate setting, the clustering methods deal with several variables simultaneously. The *Co-clustering* (also called Bi-clustering, or block clustering) simultaneously produces a partition of observations and a partition of variables (respectively *row-partition* and *column-partition*). This approach creates a structure that highlights the dependencies between observation groups and variables distributions. The co-clustering has been applied in various fields, (e.g., genetics [27], biological applications [38], text mining [45]) and has been addressed with a large numbers of methods: through spectral analysis [12], matrix factorization [31], information theory [13], and, more recently, optimal transport [30] and deep learning [44]. The Latent Block Model (LBM) [19] is a model-based co-clustering method, recently used in several domains [21, 25]. The model-based approach natively provides missing values inference and probabilistic outliers detection, while keeping a sparse parameter number, which helps interpretability.

In the standard LBM approach, datasets are composed of 1-d cells, i.e., the considered dataset is a matrix, and the co-cluster (or *block*) distributions are

univariate. LBM methods for time series, where each cell is a temporally-indexed vector, have only been introduced recently. The method FLBM from [9] uses a piecewise polynomial regression model as block distribution, which assumes that every time series admits a latent segmented structure in a common polynomial basis.

However, FLBM does not reduce the time series dimension as it directly models the time series in the observation space and is impractical for high-dimensional time series datasets. The method FunLBM [6], by contrast, includes a dimension reduction step, which relies on functional PCA projections [36] of the time series, and assumes a multivariate Normal model for the block distribution. In a different context, some works extend these approaches by assuming the presence of several independent [18] or hierarchically nested [10] partitions.

A limitation of FunLBM is that, as a parametric model, the number of blocks is assumed known a priori. In practice, it is rarely true, and this number must be estimated with an additional model selection step. This selection is usually performed either with a grid-search or by hierarchically exploring existing clusters with greedy optimization. These strategies have drawbacks: 1) with the grid search, the computation cost can be prohibitive as every combination of block number is tested, and the user is never certain that the true model is within the grid; 2) the greedy optimization heuristic is sub-optimal, by picking iteratively local optima and assuming a hierarchical structure of the mixture components; 3) whether with the greedy optimization or grid-search, choosing the model selection criterion [8, 15] is not an easy task and influences the final results.

The Dirichlet Process Mixture Model (DPMM) is a Bayesian non-parametric model-based clustering approach that can infer the number of latent clusters. As a *non-parametric* model, its parameter set dimension may increase indefinitely with the dataset size. This model is particularly well suited to massive dataset exploration, especially when it is possible to allocate additional resources to augment the dataset and discover new observation space areas.

Non-parametric approaches to LBM (NPLBM) have been studied in few works [32, 22], but, to the best of our knowledge, never applied to multivariate time series co-clustering. The non-parametric framework has also been used by [5] for the clustering of multivariate time series data based on a grid-based multivariate density estimation, but not for coclustering. This paper proposes to close the gap between FunLBM and NPLBM with *functional non-parametric Latent Block Model* (funNPLBM), the first non-parametric model-based method applied to multivariate time-series co-clustering.

In addition, our contribution includes a practical use case illustrating the method's capacities, a more compact definition of the NPLBM, experiments, hindsight on the inference settings, and Scala code provided for reproducibility.

This paper illustrates the funNPLBM application to advanced driver-assistance systems (ADAS) development, which remains a challenge for car manufacturers. These systems (e.g., emergency braking, lane centering, assisted parking, ...) are introduced gradually into new cars. Given the high number of different vehicles, driving conditions, traffic laws, and given the expected reliability, it is today im-

possible to validate ADAS rigorously with only physical "on-tracks" tests. *Groupe Renault* has invested in massive driving simulation technology to circumvent this issue. The simulation tool mimics car driving conditions based on vehicle physics, driver behavior, and interaction with a configurable environment. The simulation process outputs a large amount of information, mainly as multivariate time series with unequal length. Simulated datasets dimensions are considerable: for a given use case, the number of simulations can be as large as $\mathcal{O}(10^6)$, with $\mathcal{O}(10^3)$ variables, each recording $\mathcal{O}(10^4)$ time steps. Overall, more than $\mathcal{O}(10^{13})$ data points are produced.

In the following, Sect. 2 presents a review of the model-based clustering and co-clustering. Section 3 describes funNPLBM, its inference, and the time series preprocessing. Benchmark and experiments are studied in Sect. 4 and, finally, a real-case application on an industrial dataset is presented in Sect. 5.

2 Related work

In the next sections we use the following notations: $X = (x_{i,j,s})_{n\times p\times d}$ is the dataset, where n is the number of rows (in our cases, the simulation number), p the number of columns (in our case, the number of simulated variables) and d the observation space dimension (c.f. time series dimension reduction step in introduction of Sect.3.1).

We denote $\mathbf{x}_{i,.} = (x_{i,j})_{1\leq j\leq p}$ the i-th row of X, $\mathbf{x}_{.,j} = (x_{i,j})_{1\leq i\leq n}$ the j-th column. $X_{-i,.}$ and $X_{.,-j}$ designates the dataset without the corresponding row or column. The row-clusters memberships vector is noted $\mathbf{z} = (z_i)_n$ and the column-clusters memberships $\mathbf{w} = (w_j)_p$, such that $(z_i, w_j) = (k, l)$ indicates that element $x_{i,j}$ belongs to row-cluster k and column-cluster l.

2.1 Model-based clustering and Dirichlet Process Mixture Model

Mixture Model (MM) [11] is a probabilistic clustering approach which assumes that the overall density on the $(p.d)$-dimensional space is a convex combination of densities: $\mathrm{p}(\mathbf{x}_{i,.}) = \sum_{k=1}^{K} \pi_k F(\theta_k)$, with $\pi_k = \mathrm{p}(z_i = k)$, and $F(\theta_k) = \mathrm{p}(\mathbf{x}_{i,.}|z_i = k)$ is the distribution of $\mathbf{x}_{i,.}$ in component k, with density family F. With this definition, sampling $\mathbf{x}_{i,.}$ is performed by first drawing a membership $z_i \sim Mult(\pi)$ then drawing from $F(\theta_{z_i})$. Model inference is performed by likelihood optimization using the Expectation-Maximization (EM) algorithm [11]. The MM admits the alternative representation:

$$\forall i \in \{1, .., n\},\ \mathbf{x}_{i,.} \mid \theta_i \sim F\left(\theta_i\right),\ \theta_i \sim G,\ G = \sum_{k=1}^{K} \pi_k \delta_{\theta_k},$$

with δ_θ the Dirac delta function, In this definition, each observation x_i is associated to a parameter θ_i. Because G is finite and $K < n$, several θ_i are similar, which creates groups of elements with common distribution. The Dirichlet Process Mixture Model (DPMM) can be seen as a Bayesian non-parametric extension of

the MM where G is now an infinite random distribution with a Dirichlet Process (DP) prior distribution. This prior is a distribution over distribution that takes two parameters: a concentration α and a base distribution G_0. The distribution G admits [40] the stick-breaking representation $G = \sum_{k=1}^{\infty} \pi_k(\mathbf{v})\delta_{\theta_k}$, with

$$\pi_i(\mathbf{v}) = v_i \prod_{j=1}^{i-1} (1 - v_j), \quad \mathbf{v} = (v_i)_{\{1,\dots,n\}}, \quad v_j \overset{\text{i.i.d.}}{\sim} \text{Beta}(1, \alpha), \quad \theta_j \overset{\text{i.i.d.}}{\sim} G_0.$$

Several methods have been developed to infer DPMM's parameters, either based on variational inference [4], or Markov chain Monte Carlo (MCMC) [14, 34]. For large datasets applications, variational inference methods have often been preferred over MCMC for their speed, at the cost of hypothesis on the posterior distribution structure. However, recent works [42, 33] have made MCMC processes scalable and rehabilitate their use for industrial purposes. In particular, the collapsed Gibbs sampler is a natively fast MCMC's method that assumes the prior distribution G_0 conjugate to the density family F. This assumption enables close-form computations of the prior and posterior predictive distributions, that are used to estimate posterior cluster membership probabilities.

2.2 Latent Block Model

The LBM [21] is a bi-dimensional MM, that assumes the presence of a finite number of latent column-clusters in addition to the observation partition. Inside a block $X_{k,l}$, each cell follows the component distribution $F(\theta_{k,l})$, with F a density family. The LBM likelihood of X is given by:

$$\mathrm{p}(X) = \sum_{\mathcal{Z}\times\mathcal{W}} \mathrm{p}(\mathbf{z}, \mathbf{w})\mathrm{p}(X|\ \mathbf{z}, \mathbf{w}) = \sum_{\mathcal{Z}\times\mathcal{W}} \mathrm{p}(\mathbf{z})\mathrm{p}(\mathbf{w})\mathrm{p}(X|\ \mathbf{z}, \mathbf{w}),$$

where $\mathcal{Z}$ and $\mathcal{W}$ respectively denote the sets of all possible row and column partitions. The row-membership distribution $\mathrm{p}(\mathbf{z})$ is defined as $\prod_i \mathrm{p}(z_i) = \prod_i \pi_{z_i}$, with $\pi = (\pi_k)_K$ the mixing proportions, and $\mathrm{p}(\mathbf{w}) = \prod_j \mathrm{p}(w_j) = \prod_j \rho_{w_j}$. The density of X is $\mathrm{p}(X|\ \mathbf{z}, \mathbf{w}) = \prod_{k,l} \prod_{x\in X_{k,l}} p(x \mid \theta_{k,l})$, with $x \sim F(\theta_{k,l})$. The inference process is usually performed in an Expectation-Maximization fashion, e.g. with a Stochastic-Gibbs [26] approach, or variational inference [20]. In the following, we define the funNPLBM and the stochastic inference process.

3 Functional Non-Parametric Latent Block Model

This section introduces the novel *Functional Non-Parametric Latent Block Model* and the inference. Because multivariate time series are observed in high-dimensional spaces, in which models are known to suffer from the *curse of dimensionality*, it is essential to preprocess the dataset with a dimension reduction method. In addition, this step greatly reduces the computation cost. The functional PCA (fPCA) [36] is a two-step dimensionality reduction method, popular in the parametric model-based setting [41, 6]. This method handles time series

with unequal lengths, as is the case in our applications. During fPCA, the time series are first projected in a common polynomial basis. Among the many candidate representations available in the literature (e.g., Fourier, Legendre, Chebyshev, ...), we use an interpolated log-scaled Fourier periodogram, as advocated in [7]. This transformation consists in projecting each time series individually in the frequency domain, then interpolating the obtained log-periodograms in a common frequency basis. After this transformation, the second step of the fPCA consists in projecting the obtained coefficients in a lower-dimensional space using PCA. As a result, the unequal-length time series, described by $\mathcal{O}(10^3)$ points at the origin, are transformed into equal-length d-dimensional vectors, with d the number of PCA axes kept (usually less than 10).

3.1 Functional Bayesian Non-Parametric Latent Block Model

In another context than the multivariate time series analysis, [32] proposed a definition of the NPLBM. This work assumes Pitman-Yor Process (PYP) priors for the row-memberships and column-membership. However, PYP priors (as DP priors) are distributions over parameters and not on memberships. Using PYP, the authors are in fact implicitly defining sets of parameter distributions that are not linked to the block distributions. In the following, we propose a comprehensive definition that links block distributions and memberships intuitively.

This definition is based on a bi-dimensional extension of the DP. Each dataset cell $x_{i,j}$ is associated with a parameter $\theta_{i,j}$, grouped in the $n \times p$ matrix Θ. Two Dirichlet Process priors are used: one on Θ's rows and one on Θ's columns. This double definition ensures that every cells of a row belongs to the same row-cluster and every column element to the same column-cluster. This process is defined by:

$$
\begin{aligned}
x_{i,j} \mid \theta_{i,j} &\sim F(\theta_{i,j}) \\
\theta_{i,\cdot} \mid G \sim G, G &= \sum_{k=1}^{\infty} \pi_k(\mathbf{s})\ \delta_{\theta_{k,\cdot}}, \\
\theta_{\cdot,j} \mid H \sim H, H &= \sum_{l=1}^{\infty} \rho_l(\mathbf{t})\ \delta_{\theta_{\cdot,l}}, \\
\pi_k(\mathbf{s}) = s_k \prod_{k'=1}^{k-1} (1 - s_{k'}), \quad \mathbf{s} &= (s_k)_{\{1,\dots,n\}}, \quad s_k \overset{\text{i.i.d.}}{\sim} \text{Beta}(1, \alpha), \\
\rho_l(\mathbf{t}) = t_l \prod_{l'=1}^{l-1} (1 - t_{l'}), \quad \mathbf{t} &= (t_l)_{\{1,\dots,p\}}, \quad t_l \overset{\text{i.i.d.}}{\sim} \text{Beta}(1, \beta).
\end{aligned}
$$

With this definition, generating a dataset X is done by drawing π and ρ, sampling $\mathbf{z}$ and $\mathbf{w}$ separately, then sampling Θ given $\mathbf{z}$ and $\mathbf{w}$, and finally drawing the cells value $x_{i,j}$ from $F(\theta_{i,j})$. The likelihood of X is given by $\mathrm{p}(X \mid \mathbf{z}, \mathbf{w}, \Theta) = \prod_{i,j} \mathrm{p}(x_{i,j} \mid \theta_{i,j})$, and the joint prior distribution of the $\mathbf{z}$ and $\mathbf{w}$ is given by:

$$
\mathrm{p}(\mathbf{z}, \mathbf{w}, \mathbf{t}, \mathbf{s}, \Theta \mid G_0, \alpha, \beta) = \mathrm{p}(\mathbf{z} \mid \mathbf{s})\ \mathrm{p}(\mathbf{s} \mid \alpha)\ \mathrm{p}(\mathbf{w} \mid \mathbf{t})\ \mathrm{p}(\mathbf{t} \mid \beta)\ \mathrm{p}(\Theta \mid G_0).
$$

In the next section we describe the bi-dimensional stochastic inference process.

3.2 Model Inference

The inference is performed with a collapsed Gibbs sampler that simulates draws from the posterior distribution $\mathrm{p}(\mathbf{z}, \mathbf{w} \mid X, G_0, \alpha)$. This approach directly uses the predictive distributions closed form and therefore does not require sampling block parameters. At each iteration m, the sampler alternates the following two-steps:

1. Draw $\mathbf{z}^{(m+1)} \mid \mathbf{w}^{(m)}, X, \alpha, G_0$,
2. Draw $\mathbf{w}^{(m+1)} \mid \mathbf{z}^{(m+1)}, X, \beta, G_0$.

During the first step, the row memberships update is performed sequentially: each row-cluster membership z_i is updated with the other row-memberships $\mathbf{z}_{-i}$ and column-partition $\mathbf{w}^{(m)}$ fixed, following $\mathrm{p}(z_i = k \mid \mathbf{z}_{-i}, X, \mathbf{w}^{(m)}, \alpha, G_0) \propto$

$$\begin{cases} \dfrac{n_k}{n-1+\alpha}\ \mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}^{(m)}, X_{-i}, \mathbf{z}_{-i}, G_0), & \text{existing cluster } k, \quad (1) \\ \dfrac{\alpha}{n-1+\alpha}\ \mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}^{(m)}, G_0), & \text{new row-cluster}, \quad (2) \end{cases}$$

where n_k is row-cluster k size. We emphasize that the parameters Θ do not appear in these formulas, as they are integrated over in the predictive distributions.

In eq. (2), $\mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}^{(m)}, G_0) = \prod_l \mathrm{p}(\mathbf{x}_{i,l}^{(m)} \mid G_0)$, with $\mathbf{x}_{i,l}^{(m)}$ the elements of row i in column-cluster l at iteration m. For each column-cluster l, the prior predictive distribution of $\mathbf{x}_{i,l}^{(m)}$ is obtained by integrating over the component's parameter: $\mathrm{p}(\mathbf{x}_{i,l}^{(m)} \mid G_0) = \int_\theta p(\mathbf{x}_{i,l}^{(m)} \mid \theta)\ \mathrm{p}(\theta \mid G_0)\ d\theta$. Because G_0 is a prior conjugate to F, this integral is analytically tractable (see Sect. 3.3 for the detail in the multivariate gaussian case). The joint posterior predictive distribution $\mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}^{(m)}, X_{-i}, G_0)$ from eq. (1) has the same definition, with G_0 updated with the observations inside the blocks.

The second step of the Gibbs-sampler is performed symmetrically on column clusters. Once the maximum number of iterations reached, the row and column final partitions are estimated with the mean of the partitions sampled after burn-in (c.f. Sect. 3.4 - §3 for computation detail). The algorithm global complexity is linear in n, p, in the current blocks number and in the iterations number, but also depends on the complexity of the sufficient statistics update and predictive distribution computation. The inference is summarized in algorithm 1. In the next subsection we detail how eq. (1) and (2) simplify with our choice of G_0.

3.3 Multivariate Gaussian case

After the time series preprocessing step, each dataset cell $x_{i,j}$ is a d-dimensional numeric vector produced by the fPCA, that we model with a multivariate Gaussian density. As as conjugate prior, we choose G_0 to be the Normal Inverse Wishart (NIW) distribution with hyper-parameters $(\mu_0, \lambda_0, \Psi_0, \nu_0)$. Given $X_{k,l}$,

the observations in block (k,l), the block parameters posterior distribution is formally defined by $\mathrm{p}(\mu, \Sigma \mid X_{k,l}) = NIW(\mu, \Sigma \mid \mu_{k,l}, \kappa_{k,l}, \Psi_{k,l}, \nu_{k,l})$, with:

$$\mu_{k,l} = \frac{\kappa_0 \mu_0 + n_{k,l} \overline{x}_{k,l}}{\kappa_{k,l}}, \quad \kappa_{k,l} = \kappa_0 + n_{k,l}, \quad \nu_{k,l} = \nu_0 + n_{k,l},$$
$$\Psi_{k,l} = \Psi_0 + C + \frac{\kappa_0 n_{k,l}}{\kappa_{k,l}} (\mu_0 - \overline{x}_{k,l})(\mu_0 - \overline{x}_{k,l})^T,\ C = \sum_{x \in X_{k,l}} (x - \overline{x}_{k,l})(x - \overline{x}_{k,l})^T.$$

With these parameters and following [16], the joint posterior predictive distribution needed in eq. (1) is the multivariate t-student distribution:

$$t_{\nu_{k,l}-p+1} \left(\mu_{k,l}, \frac{(\kappa_{k,l}+1)\, \Psi_{k,l}}{\kappa_{k,l}(\nu_{k,l}-p+1} \right).$$

This definition outlines the cubic complexity in d, due to the t-student density computation cost. The next section details the inference process implementation.

3.4 Implementation

G_0 hyperparameters specification The clustered objects are PCA coefficients, which are centered. Therefore we set μ_0 to be the d-dimensional zero vector. The precision matrix Ψ_0 specification is a bit trickier and depends on assumptions on the dataset. For non-parametric autoregressive models, [39] compares several prior specifications for Ψ_0 and concludes that the dataset precision obtained with maximum likelihood estimation is a good standard, which we keep in our application. κ_0 and ν_0, which represent the user's confidence in μ_0 and Ψ_0, are set to their lowest value, as we want them as uninformative as possible.

Initialization strategy Initializing a bayesian non-parametric MCMC inference algorithm is often done with single-component partition [37, 35]. However, [23] shows that this strategy may be suboptimal when dealing with high-dimensional datasets and high component numbers, and recommends to use random partition as initial state, with more components than the actual component number. However, this number is unknown. A tempting solution is to initialize the partitions with one component per observation, but this choice is computationally expensive because the membership update has linear complexity in the block number. We propose a heuristic, consisting of running the inference process twice. In a first run, the inference is initialized with a one-cluster partition; after this first run completion, the maximum block number sampled during the inference is kept and used as the initial number of components for the second run.

Infering the final partitions In output of the Gibbs sampler algorithm, the user gets a set of sampled row-partitions $(\mathbf{z}^{(m)})_m$ and column-partitions $(\mathbf{w}^{(m)})_m$, that must be aggregated to obtain the final partition modes: $\hat{\mathbf{z}}$ and $\hat{\mathbf{w}}$. This

consensus partition estimation is an NP-complete problem [29], that several works have addressed (c.f. [43] for an extensive review). We use the recent method [17], which proposes an efficient extension of a combinatorial optimization method [24] that construct the partition with the minimal distance [28] to the samples, without assumptions on the final number of clusters or on the clustering structure.

Algorithm 1: FunNPLBM Inference

input : Dataset X, $n \times p \times d$ tensor
α, β, G_0 (c.f. specification strategies in Sect. 3.4 - §1)
Iteration number M

output : Estimated row-partition $\hat{\mathbf{z}}$ and column-partition $\hat{\mathbf{w}}$

Initialize $\mathbf{z}^{(0)}$ and $\mathbf{w}^{(0)}$ (c.f. initialization methods in Sect. 3.4 - §2)
for $m \leftarrow 1$ **to** M **do**
  for $i \leftarrow 1$ **to** n **do**
    Compute $p(z_i \mid \mathbf{z}_{-i}, X, \mathbf{w}^{(m)}, \alpha, G_0)$ as defined by eq. (1) and (2)
    Sample $z_i^{(m+1)}$
  for $j \leftarrow 1$ **to** p **do**
    Compute $p(w_j \mid \mathbf{w}_{-j}, X, \mathbf{z}^{(m+1)}, \beta, G_0)$ as defined by eq. (1) and (2)
    Sample $w_j^{(m+1)}$
Average the partitions (c.f. Sect. 3.4 - §3) to obtain the final partitions $\hat{\mathbf{z}}$ and $\hat{\mathbf{w}}$.

4 Experiments on Synthetic Data

4.1 Experimental setup

Benchmark and experiments are conducted on a dataset sampled from a known generative model. The dataset is generated by sampling from the distributions $\mathcal{N}\left(f_{k,l}(t), s^2\right)$ where $f_{k,l}$ is a given *prototype* function and $s = 0.02$. The estimated block partition quality is compared to the known generative partition, based on several scores: the Rand Index (RI), Adjusted Rand Index (ARI) and the Normalized Mutual Information (NMI). The RI is a popular criterion choice in the clustering domain, which represents the proportion of correctly grouped and separated observations with respect to the observed classes. The ARI is a corrected-for-chance version of the RI that takes into account the probability of getting good RI at random. The NMI is an entropy-based criterion from the information theory literature estimating the quantity of knowledge a partition gives on another. In the following benchmark and experiments, we work on a dataset of dimension 140×140, with unbalanced row cluster sizes $(20, 30, 40, 30, 20)$ and column cluster sizes $(40, 20, 30, 20, 30)$, which amounts to 19600 time series.

4.2 Baselines and compared methods

As detailed in the introduction, co-clustering has been the subject of numerous works on various data types, but most of the time on univariate observations matrices (i.e., $d = 1$). To the best of our knowledge, the co-clustering on datasets containing multidimensional cells ($d > 1$) has only been dealt with very recently in the literature, and only a few methods currently exist. Apart from the model-based method from [9], which does not include a dimension reduction aspect and is, therefore, impractical on large datasets, FunLBM [6] is the only existing method dealing with this use case. In addition, we consider two decoupled methods that perform co-clustering by inferring row-partitions and column-partitions independently: a bi-dimensional Gaussian Mixture Model (BGMM) and a bi-dimensional DPMM (BDPMM). This benchmark compares the block partitions quality, but also the results of the associated model selection step, described by the selected number of blocks, compared to the true number (5×5). For BGMM and FunLBM, which are parametric approaches, this selection step is performed by a grid-search with the ICL selection [3, 26] and with a maximum of seven row clusters and seven column clusters. In the non-parametric cases, this selection is natively performed with the inference, with hyper-parameters $\alpha = 0.1, \beta = 0.1$, and G_0 specified as described in 3.4. For each methods, the results are the performances mean on 10 runs - or the median in the case of the number of clusters. These performances are shown in Table 1.

FunNPLBM has the upper-hand in this benchmark, and estimates the correct structure, while its parametric counterpart FunLBM slightly underestimates the correct number of blocks. In this setup, FunLBM's performances comes from locally optimal estimations of the candidate models during the grid-search model selection. The same effect explains the performance's gap between BGMM and BDPMM. Overall, the two-steps methods BGMM and BPDMM show the worse results, presumably because they both infer the row and column partitions independently and therefore cannot use the row-clusters informations to help finding the best column-clusters partition, and reciprocally. In conclusion, FunNPLBM is able to simultaneously select the right model and to infer the right partition. On an Ubuntu 18.04 64-bit with Intel® Core™ i7-8850H CPU @ 2.60GHz × 12 CPU 32Gib RAM, the computing time of FunNPLBM was <= 30 seconds.

Table 1. Benchmark with Bi-dimensional GMM (BGMM), Bi-dimensional DPMM (BDPMM), functional Latent Block Model (FunLBM), and our proposal FunNPLBM

Score	BGMM	BDPMM	FunLBM	FunNPLBM
ARI	0.558	0.667	0.897	1
RI	0.927	0.958	0.990	1
NMI	0.837	0.905	0.968	1
# Blocks	12	16	22	25

4.3 Hyperparameters specification study

The hyperparameter set is composed of the base NIW distribution G_0, the concentration parameters (α, β), the iterations number M, and the preprocessing parameters: the Fourier basis dimension and the number of PCA axes. In an unsupervised context, hyperparameters specification remains an active research topic as there is no label to support hyperparameter inference. Consequently, it is not possible to give definitive and general good specifications choices, which depend on the dataset contents and must be hand-tuned by the experts. These specifications, however, can be based on the knowledge of each hyperparameters impact on funNPLBM's behavior, which we illustrate in the following. In each case, we compare funNPLBM's performances when one hyperparameter varies while keeping the others equal to given default values: $\alpha = \beta = 0.1$, $M = 10$, a Fourier expression basis of dimension 30 and 3 PCA axes.

Concentration parameters and number of iterations In the funNPLBM setting, as in the DPM, the prior distribution of the number of components is an increasing function of the concentration parameters: without knowledge of the data, the higher the concentration parameters, the higher the probability of producing high numbers of components. Because the whole method is symmetric on the dataset rows and columns, and because the experiment dataset dimensions n and p are equal, we consider the case $\alpha = \beta$. As shown in Fig. 1, the concentration parameters effects are negligible for this dataset and only have an impact when extremely high (>1E12) - in which case the number of components is highly overestimated - or extremely small (<1E-10) - in which case only one-cluster partitions are inferred. This small impact of α comes presumably from the high separation of the components in the time series high-dimensional observation space. This separability is simulated for this experiment but is consistent with what we observe in practice. This separation also explains the small number of iterations needed for convergence (here, less than 4 for $1e{-}6 \leq \alpha \leq 1e10$), and the high stability of the MCMC. For a specific dataset, if α and β's values are complex to specify, we advise, as a simple workaround, to add a Gamma hyper-prior assumption for α and β and estimate their values during the inference algorithm, which is a common practice in the DPM setting. It implies, however, to study specification strategies for the Gamma distribution parameters.

Preprocessing parameters The Fourier Basis dimension and the PCA axes numbers both influence funNPLBM's performances and affect the trade-off between sparsity and quality of time series representation. Fig. 2(a) shows the effects of under-estimating or over-estimating the number of PCA axes. Low numbers of PCA axes (<3) are associated with poor scores and few block components due to poor representations of the time series, which are too close in the projection space. On the contrary, if the number of axes is too high (>=6), the high dimensionality exaggerates the time series separation, and the component number is overestimated, which explains the sharp decrease of the ARI and NMI.

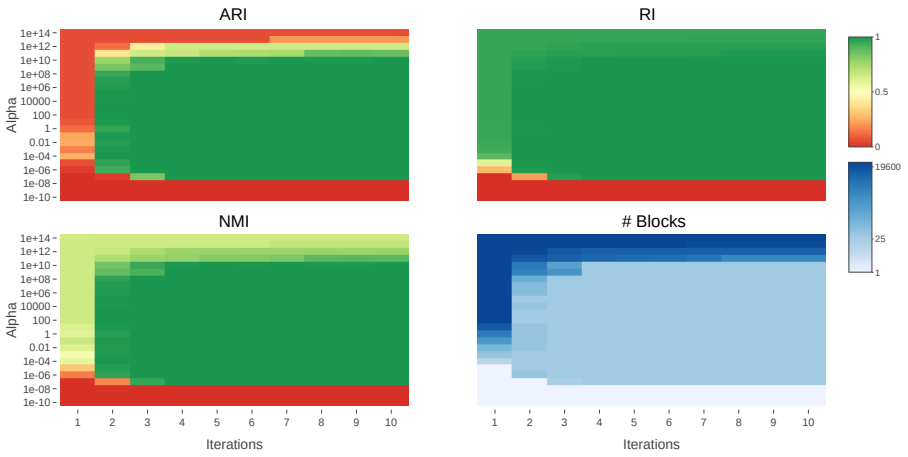

Fig. 1. Scores versus Alpha and Iterations - with # Block in log scale

In our use cases, we observed that 3 PCA axes lead to the most interesting results. In Fig. 2(b), we observe that, with a fixed number of 3 PCA axes, high polynomial basis dimensions (> 50) are correlated with poor scores, presumably because of lower time series representation quality (reflected by the low variance explained score). On the contrary, when this basis dimension is low (<10), the 3-axes PCA adequately represents the information and the variance explained is high ($>= 0.97$). However, in this case, the Fourier basis dimension is too low to adequately represent the time series, which explains the poor scores. The studied

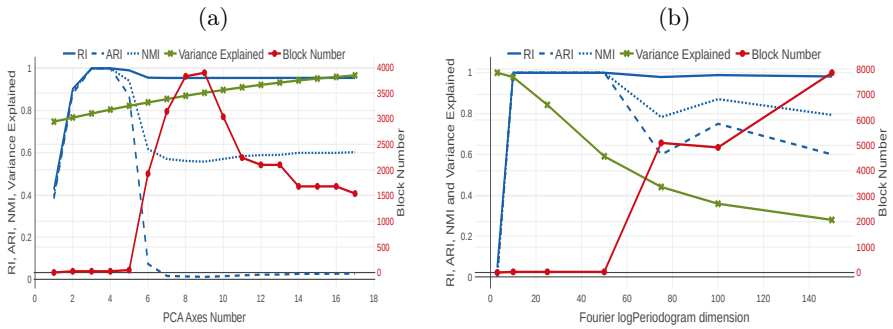

Fig. 2. (a) Scores versus number of PCA axes. (b) Scores versus Fourier logPeriodogram dimension.

datasets, data generation scripts, and the Scala code used for the benchmark and the experiments are available at the following (anonymized) github repository https://tinyurl.com/4k9jze45, along with the data simulation method. In the

next section, a real-case situation is studied and illustrates the method's interest for Advanced Driving-Assistance Systems validation.

5 Application to Advanced Driver-Assistance System Validation

5.1 Use case description

This section illustrates the use of the co-clustering approach to the Emergency Lane Keeping (ELK) assistance system validation. In a straight lane scenario, the vehicle under test (called *ego*) is drifting towards an oncoming car on the other lane (c.f. Fig. 3). The ELK system is expected to put the vehicle back to its lane center with an emergency maneuver. With different settings (ego speed, drift angle, ...), the simulation system has produced a set of 400 simulations, described by 22 features. We emphasize that these simulations are generated with a simulation black-box that faithfully recreates the real-life driving conditions, and are not simply produced by the generative model proposed in this article.

The time series are expressed in a common log-periodogram with dimension 40, then reduced to 3 PCA axes. The concentration parameters are set to 1e-2, and the NIW parameters to the default values discussed in Sect. 3.4. The objective is to discriminate simultaneously driving patterns and correlated variables.

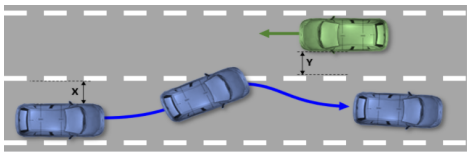

Fig. 3. Use case illustration: ego drifts from its lane, crosses the center line and heads toward an oncoming vehicle. The system detects the target and change ego's direction.

5.2 Results

The final co-clustering is the block partitions mean (c.f. averaging methods in 3.4 - §3) of 10 samples obtained after a burnin of 10 iterations and is composed of 6 row-clusters and 13 column-clusters. With color indicating block membership, Fig. 4 shows the global co-clustering structure, and Fig. 5 an extract of the block contents. The first column-cluster discriminates uninformative signals (car width, road bend radius, constant inactive system, ...). The other column-clusters relevantly regroup variables of interest: the 6th, 7th, and 8th column clusters respectively regroup ego direction variables, ego lateral position variables, and ego speed variables. Their content is shown in Fig. 5 top-left, top-right and bottom-left respectively. The row-clustering is also insightful: each row-cluster

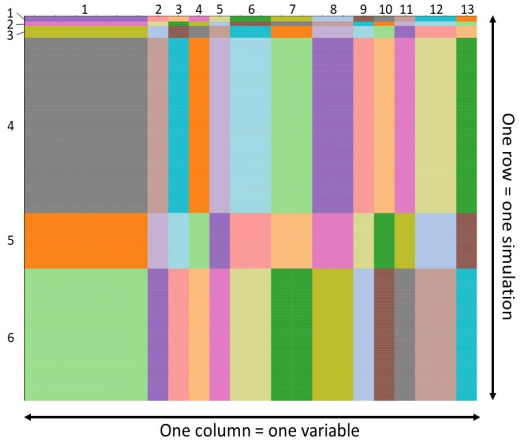

Fig. 4. Resulting co-clustering on Emergency Lane Keeping (ELK) dataset. The result consists of 6 row-clusters and 13 column-clusters

correspond to well-separated driving behaviors. This separation is best seen in Fig. 5 (top-right) that shows the following driving behaviors: 1) ego drifting left and the ELK system failing (light green); 2) the symmetric behavior on the right (dark green); 3) the ELK system correcting the car trajectory (light orange).

Finally, the three other row-clusters (regrouping the remaining 5% of the observations) are composed of outliers simulations, with driving behavior displayed on Fig. 5 (bottom-right). In this situation, the oncoming car is correctly detected, and ego heading is changed accordingly, but not enough to prevent the collision. In conclusion, funNPLBM has correctly discriminated uninformative signals while creating meaningful clusters of features and clusters of simulations. From this information, it is easy to visualize the variety of driving behaviors that compose our datasets and understand them from the variable perspectives, which was the original objective of the application. The next step is to link the driving behavior to the control logic parameters and, if need be, refine them to reach the performance objectives. With the same computer specifications than for the experiments (c.f. Sect.4.2), the computing time was < 20 seconds.

6 Conclusion and future work

This paper describes FunNPLBM, a Bayesian non-parametric based method that addresses the problem of co-clustering multivariate time series. This work proposes the first Bayesian non-parametric co-clustering method dedicated to functional data analysis, the description of an adapted collapsed Gibbs sampling, and a more compact definition of the NPLBM model.

This method regroups redundant features, discriminates uninformative ones, and provides the user with a two-dimensional analysis of a multivariate time series dataset. The hyperparameters (concentrations parameters, components

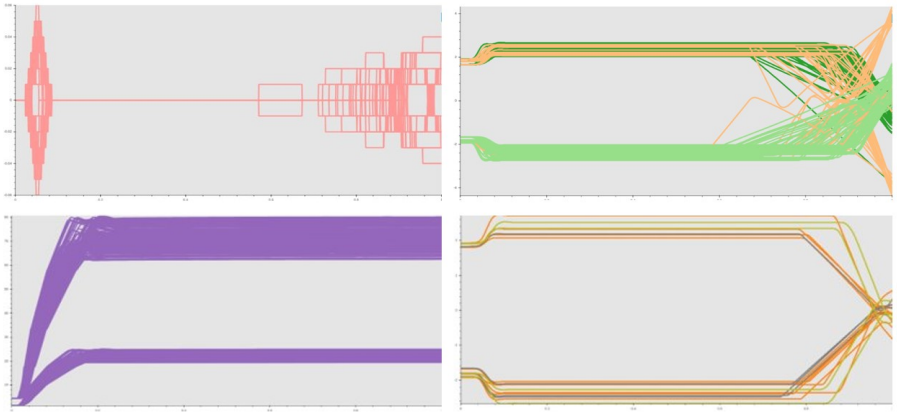

Fig. 5. top-left: two highly negatively correlated direction change signals; top-right: ego lateral position in the 3 biggest observation clusters; bottom-left: 2 correlated speed variables; bottom-right: 3 outlier driving pattern in the 3 smallest observation clusters.

parameters) specifications are discussed and experimented on a simulated dataset, and a benchmark is presented that shows FunNPLBM adequacy in a context that matches our assumptions. Finally, the method is applied to a real-case dataset from the autonomous driving system validation domain. In this application, FunNPLBM proves its ability to create meaningful clusters of driving behavior and correlated variables simultaneously.

We are confident that FunNPLBM can be useful in other domains dealing with correlated temporal variables. For instance in industrial contexts for sensor anomaly detection or predictive maintenance, in health for ECG and biological signals data analysis, in finance for stock trade data analysis. In addition, The method can also be applied to anomaly detection thanks to the native production of probabilistic predictive intervals and supervised classification by simply constraining the row and column partitions values. We consider two extensions: a) higher-order tensor co-clustering; b) relaxing the model to multi-clustering. These perspectives will be addressed in future work.

References

1. Aghabozorgi, S., Shirkhorshidi, A.S., Wah, T.Y.: Time-series clustering–a decade review. Information Systems **53**, 16–38 (2015)
2. Bagnall, A., Lines, J., Bostrom, A., Large, J., Keogh, E.: The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery **31**(3), 606–660 (2017)
3. Biernacki, C., Celeux, G., Govaert, G.: Assessing a mixture model for clustering with the integrated completed likelihood. IEEE transactions on pattern analysis and machine intelligence **22**(7), 719–725 (2000)

4. Blei, D.M., Jordan, M.I., et al.: Variational inference for dirichlet process mixtures. Bayesian analysis **1**(1), 121–143 (2006)
5. Boullé, M.: Functional data clustering via piecewise constant nonparametric density estimation. Pattern Recognition **45**(12), 4389–4401 (2012)
6. Bouveyron, C., Bozzi, L., Jacques, J., Jollois, F.X.: The functional latent block model for the co-clustering of electricity consumption curves. Journal of the Royal Statistical Society: Series C (Applied Statistics) **67**(4), 897–915 (2018)
7. Caiado, J., Crato, N., Peña, D.: Comparison of times series with unequal length in the frequency domain. Communications in Statistics—Simulation and Computation® **38**(3), 527–540 (2009)
8. Celeux, G., Frühwirth-Schnatter, S., Robert, C.P.: Model selection for mixture models—perspectives and strategies. Handbook of mixture analysis (2018)
9. Chamroukhi, F., Biernacki, C.: Model-based co-clustering of multivariate functional data. In: Proceedings of the 61st World Statistics Congress (2017)
10. Côme, E., Jouvin, N., Latouche, P., Bouveyron, C.: Hierarchical clustering with discrete latent variable models and the integrated classification likelihood. Advances in Data Analysis and Classification pp. 1–30 (2021)
11. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society: Series B (Methodological) **39**(1), 1–22 (1977)
12. Dhillon, I.S.: Co-clustering documents and words using bipartite spectral graph partitioning. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 269–274 (2001)
13. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 89–98 (2003)
14. Escobar, M.D.: Estimating normal means with a dirichlet process prior. Journal of the American Statistical Association **89**(425), 268–277 (1994)
15. Forest, F., Mourer, A., Lebbah, M., Azzag, H., Lacaille, J.: An invariance-guided stability criterion for time series clustering validation. In: International Conference on Pattern Recognition (ICPR) (2020)
16. Gelman, A., Carlin, J.B., Stern, H.S., Dunson, D.B., Vehtari, A., Rubin, D.B.: Bayesian data analysis. CRC press (2013)
17. Glassen, T.J., von Oertzen, T., Konovalov, D.A.: Finding the mean in a partition distribution. BMC bioinformatics **19**(1), 1–10 (2018)
18. Goffinet, E., Coutant, A., Lebbah, M., Azzag, H., Giraldi, L.: Conditional latent block model: a multivariate time series clustering approach for autonomous driving validation. arXiv preprint arXiv:2008.00946 (2020)
19. Govaert, G., Nadif, M.: Clustering with block mixture models. Pattern Recognition **36**(2), 463–473 (2003)
20. Govaert, G., Nadif, M.: Block clustering with bernoulli mixture models: Comparison of different approaches. Computational Statistics & Data Analysis **52**(6) (2008)
21. Govaert, G., Nadif, M.: Co-clustering: models, algorithms and applications. John Wiley & Sons (2013)
22. Görür, D.: Nonparametric Bayesian Discrete Latent Variable Models for Unsupervised Learning. Doctoral thesis, Technische Universität Berlin, Fakultät IV - Elektrotechnik und Informatik, Berlin (2007)
23. Hastie, D.I., Liverani, S., Richardson, S.: Sampling from dirichlet process mixture models with unknown concentration parameter: mixing issues in large data implementations. Statistics and computing **25**(5), 1023–1037 (2015)

24. Huelsenbeck, J.P., Andolfatto, P.: Inference of population structure under a dirichlet process model. Genetics **175**(4), 1787–1802 (2007)
25. Jacques, J., Biernacki, C.: Model-based co-clustering for ordinal data. Computational Statistics & Data Analysis **123**, 101–115 (2018)
26. Keribin, C., Brault, V., Celeux, G., Govaert, G.: Estimation and selection for the latent block model on categorical data. Statistics and Computing **25**(6) (2015)
27. Kluger, Y., Basri, R., Chang, J.T., Gerstein, M.: Spectral biclustering of microarray data: coclustering genes and conditions. Genome research **13**(4), 703–716 (2003)
28. Konovalov, D.A., Litow, B., Bajema, N.: Partition-distance via the assignment problem. Bioinformatics **21**(10), 2463–2468 (2005)
29. Křivánek, M., Morávek, J.: Np-hard problems in hierarchical-tree clustering. Acta informatica **23**(3), 311–323 (1986)
30. Laclau, C., Redko, I., Matei, B., Bennani, Y., Brault, V.: Co-clustering through optimal transport. In: International Conference on Machine Learning. PMLR (2017)
31. Long, B., Zhang, Z., Yu, P.S.: Co-clustering by block value decomposition. In: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. pp. 635–640 (2005)
32. Meeds, E., Roweis, S.: Nonparametric bayesian biclustering. Tech. rep., Citeseer (2007)
33. Meguelati, K., Fontez, B., Hilgert, N., Masseglia, F.: Dirichlet process mixture models made scalable and effective by means of massive distribution. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 502–509 (2019)
34. Neal, R.M.: Markov chain sampling methods for dirichlet process mixture models. Journal of computational and graphical statistics **9**(2), 249–265 (2000)
35. Nguyen, V.A., Boyd-Graber, J., Altschul, S.: Dirichlet mixtures, the dirichlet process, and the structure of protein space. Journal of Computational Biology **20** (2013)
36. Ramsay, J., Silverman, B.: Principal components analysis for functional data. Functional data analysis pp. 147–172 (2005)
37. Ross, G.J., Markwick, D.: dirichletprocess: An r package for fitting complex bayesian nonparametric models (2018)
38. Schlüter, K., Drenckhahn, D.: Co-clustering of denatured hemoglobin with band 3: its role in binding of autoantibodies against band 3 to abnormal and aged erythrocytes. Proceedings of the National Academy of Sciences **83**(16) (1986)
39. Schuurman, N., Grasman, R., Hamaker, E.: A comparison of inverse-wishart prior specifications for covariance matrices in multilevel autoregressive models. Multivariate Behavioral Research **51**(2-3), 185–206 (2016)
40. Sethuraman, J.: A constructive definition of dirichlet priors. Statistica sinica (1994)
41. Slimen, Y.B., Allio, S., Jacques, J.: Model-based co-clustering for functional data. Neurocomputing **291**, 97–108 (2018)
42. Williamson, S., Dubey, A., Xing, E.: Parallel markov chain monte carlo for nonparametric mixture models. In: International Conference on Machine Learning (2013)
43. Xanthopoulos, P.: A review on consensus clustering methods. In: Optimization in science and engineering, pp. 553–566. Springer (2014)
44. Xu, D., Cheng, W., Zong, B., Ni, J., Song, D., Yu, W., Chen, Y., Chen, H., Zhang, X.: Deep co-clustering. In: Proceedings of the 2019 SIAM International Conference on Data Mining. pp. 414–422. SIAM (2019)
45. Yan, Y., Chen, L., Tjhi, W.C.: Fuzzy semi-supervised co-clustering for text documents. Fuzzy Sets and Systems **215**, 74–89 (2013)