CONDITIONAL LATENT BLOCK MODEL: A MULTIVARIATE TIME SERIES CLUSTERING APPROACH FOR AUTONOMOUS DRIVING VALIDATION

Etienne Goffinet, Anthony Coutant, Mustapha Lebbah, Hanane Azzag, Loïc Giraldi

CONDITIONAL LATENT BLOCK MODEL: A MULTIVARIATE TIME SERIES CLUSTERING APPROACH FOR AUTONOMOUS DRIVING VALIDATION

Etienne Goffinet
Laboratoire Informatique de Paris-Nord
Université Sorbonne Paris Nord
Villetaneuse, France
etienne.goffinet@lipn.univ-paris13.fr

Anthony Coutant
Laboratoire Informatique de Paris-Nord
Université Sorbonne Paris Nord
Villetaneuse, France

Mustapha Lebbah
Laboratoire Informatique de Paris-Nord
Université Sorbonne Paris Nord
Villetaneuse, France

Hanane Azzag
Laboratoire Informatique de Paris-Nord
Université Sorbonne Paris Nord
Villetaneuse, France

Loïc Giraldi
Groupe Renault SAS
Avenue du Golf
Guyancourt, France

August 4, 2020

ABSTRACT

Autonomous driving systems validation remains one of the biggest challenges car manufacturers must tackle in order to provide safe driverless cars. The high complexity stems from several factors: the multiplicity of vehicles, embedded systems, use cases, and the very high required level of reliability for the driving system to be at least as safe as a human driver. In order to circumvent these issues, large scale simulations reproducing this huge variety of physical conditions are intensively used to test driverless cars. Therefore, the validation step produces a massive amount of data, including many time-indexed ones, to be processed. In this context, building a structure in the feature space is mandatory to interpret the various scenarios. In this work, we propose a new co-clustering approach adapted to high-dimensional time series analysis, that extends the standard model-based co-clustering. The FunCLBM model extends the recently proposed Functional Latent Block Model and allows to create a dependency structure between row and column clusters. This structured partition acts as a feature selection method, that provides several clustering views of a dataset, while discriminating irrelevant features. In this workflow, times series are projected onto a common interpolated low-dimensional frequency space, which allows to optimize the projection basis. In addition, FunCLBM refines the definition of each latent block by performing block-wise dimension reduction and feature selection. We propose a SEM-Gibbs algorithm to infer this model, as well as a dedicated criterion to select the optimal nested partition. Experiments on both simulated and real-case Renault datasets shows the effectiveness of the proposed tools and the adequacy to our use case.

Keywords Model-Based Clustering · Coclustering · Time Series Analysis

1 Introduction

Autonomous car development remains a challenge for car manufacturers. Nowadays, advanced driving assistance systems are being introduced gradually into new car models, yielding more and more complex vehicles that must be proven to be safe. Given the high number of different vehicles, different models, embedded systems, drivers, and expected reliability, physical validation of cars has become prohibitive. *Groupe Renault* has made the technical choice to invest in massive driving simulation technology in order to circumvent this issue. The simulation tool chain mimics car driving conditions based on vehicle physics, driver behavior, and interaction with a configurable environment. The software produces a large quantity of data of excellent quality that needs to be mined. The simulation process outputs a large amount of information in the form of multivariate time series. Data size, complexity, and dimensions are considerable: the simulation of a validation test suite, the number of simulations can be as large as $\mathcal{O}(10^6)$, with $\mathcal{O}(10^3)$ signals, each recording $\mathcal{O}(10^4)$ time steps. Overall, this setting implies the production of more than $\mathcal{O}(10^{13})$ data points.

One issue driving system developers are facing with this large amount of data, is the ability to identify operational modes of the driving systems, in order to better understand and refine the control logic. Specific visualization methods are required for this purpose. Clustering is a first approach to tackle the problem, which consists in the automatic grouping of "similar" observations into homogeneous groups (clusters). The clustering of time series (also called *functional* data) already helps decision-making in many domains (Health, Finance, Industry. . .) and has been intensively studied (see Bagnall et al. (2017); Aghabozorgi et al. (2015) for reviews). In such methods, observation clusters construction is based on every functional features (see Fig. 1, left panel).

Co-clustering techniques produce joint clusters of observations and clusters of features. The Latent Block Model (LBM) is a model-based approach to co-clustering which has recently proved its effectiveness in various applications (Govaert and Nadif (2013); Jacques and Biernacki (2018)). It can be applied when every feature can be modeled with the same probability density function, for instance applied to the clustering of text based on word counts, or in the functional case like in Bouveyron et al. (2018) where features (also called *signals*) come from the day-by-day segmentation of electricity consumption curves. Latent Block Model applied to time series is a recent approach, and there exists only few works on the topic Chamroukhi and Biernacki (2017); Slimen et al. (2018); Schmutz et al. (2019). In particular, Bouveyron et al. (2018) presents an interesting Functional Latent Block Model (*FunLBM*) that relies on functional PCA projections of the series expressed in a Fourier basis. Co-clustering methods enable grouping of similar features with, in our applications, a limiting constraint: observation clusters and feature clusters are independent and observation partition is common to every features (see Fig. 1, middle panel).

In real cases, chances are that for every feature cluster there is a different set of observation clusters. The main advantage of this new model, called Functional Conditional Latent Block Model (*FunCLBM*), is that a joint structure is introduced in the clusters dependency: clusters are not assumed to be independent anymore and observation clusters depend on feature ones. Consequently, users have at disposal a clustering made of multiple views, from which it is easy to discard groups of useless features. This construction grants a valuable tool to the expert: feature selection and discrimination of uninformative features. Fig. 1 shows the differences between the clustering, co-clustering, and the proposed conditional co-clustering approaches.

In the most recent FunLBM works, time series are expressed in a common polynomial basis and block-wise functional PCAs (Ramsay and Silverman (2005)) are applied on the regression coefficients. FunCLBM builds on this construction and improves the first part: the time series are first transformed with a Discrete Fourier Transform procedure, then obtained periodograms are interpolated in order to construct a finely-tuned expression basis. The rest of this paper is organized as follows: Section 2 presents the work related to both model-based clustering and co-clustering. Both data processing aspects and the FunCLBM workflow are detailed in Section 3. Section 4 describes the inference and implementation details. Experiments on both a simulated dataset and a real case data are presented in Section 5. Finally, the paper ends on Section 6 with future work perspectives.

2 Related work

2.1 Model-based clustering

Mixture modeling (*MM* is a standard clustering approach first introduced in Dempster et al. (1977) and based on the assumption of latent clusters. The cluster membership probabilities are jointly estimated with the mixture parameters: the proportions and the distribution parameters of each component. In opposition to non-model-based methods, this approach enables the construction of confidence intervals and probabilistic outliers detection. The inference is performed by optimizing the likelihood of the model, with a dedicated algorithm, the Expectation-Maximization (EM) algorithm

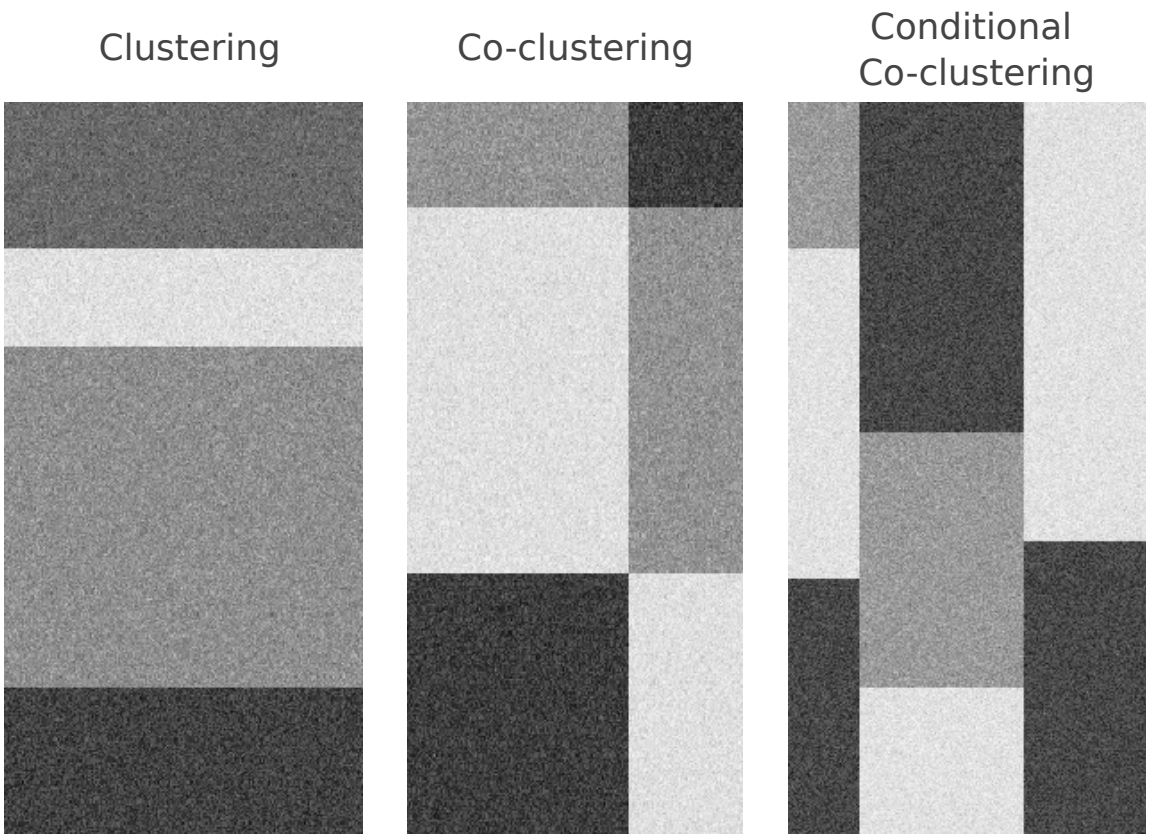

Figure 1: Differences between Mixture Model clustering, Latent Block Model Co-clustering and Conditional Latent Block Model Co-clustering

Dempster et al. (1977). This algorithm is iterative and composed of two steps: the E step computes the posterior cluster membership probabilities while the M step updates the model parameters based on these probabilities.

Several variants of this algorithm exist, which mainly consist in variations of the M step. The Classification-EM version updates the parameters based on the observations that are "most likely" to belong to each cluster (this version is the closest to the popular K-Means, of which it can be considered a probabilistic generalisation). In the Stochatisc-EM (SEM) approach, the cluster belongings are drawn at random according to the membership probabilities. FunCLBM uses this last EM extension, in an adapted version detailed in Section 4. Model-based clustering has been subject to many works, improving several aspects (e.g. the initialization Biernacki et al. (2003) or the model selection strategy Vlassis and Likas (2002); Biernacki et al. (2000)). It has also been extended to the modeling of various data types, including time series Bouveyron and Jacques (2011); Chamroukhi et al. (2010) The Latent Block Model is an extension of the MM model addressing the co-clustering problem.

2.2 Model-based co-clustering

The Latent Block Model, proposed in Govaert and Nadif (2013), assumes the presence of latent feature clusters (*column-cluster*) in addition to the observation clusters (*row-cluster*). In the standard multivariate MM, each component of the mixture defines a density over multivariate observations (an observation corresponding to a row). It is no longer the case in the LBM framework: the modeled object is the *cell*, i.e the intersection of a row and a column, that is an observation for a given feature.

Inside the block component created by crossing the column and row partition, LBM assumes the independence conditional to the block, which means that, given a block partition, every cell composing a block is independent of each other. From this perspective, the density of an LBM component is univariate.

Given an observed dataset $x = (x_{ij})_{n\times p}$ of n observations of p features, let denote $z = (z_{ik})_{n\times K}$ and $w = (w_{j\ell})_{p\times L}$ the random binary matrices indicating respectively the row and cluster partitions. The standard LBM is defined by:

$$\mathrm{p}(x;\theta) = \sum_{\mathcal{Z}\times\mathcal{W}} \mathrm{p}(z;\theta)\mathrm{p}(w;\theta)\mathrm{p}(x|z,w;\theta),$$

where $\mathcal{Z}$ and $\mathcal{W}$ respectively denote the sets of all possible row and column partitions. The quantity $\mathrm{p}(z;\theta)$ is defined as $\prod_{ik}\pi_k^{z_{ik}}$, with $\pi = (\pi_k)_K$ the membership probabilities prior (respectively for w with the membership probabilities $\rho = (\rho_\ell)_L$). The set of parameters θ is composed of the mixing proportions and of the component density parameters. These densities $\mathrm{p}(x|z,w;\theta)$ are part of a common model family suited to clustering interpretation.

We emphasize that our goal is to provide a cluster belonging probability that is not independent anymore, i.e. the approximation $\mathrm{p}(z,w;\theta) \approx \mathrm{p}(z;\theta)\mathrm{p}(w;\theta)$ does not hold.

2.3 Functional co-clustering

In the case of time series co-clustering, the dataset is composed of sequences: $x = (x_{ij})_{n\times p}$, with $x_{ij} = (x_{ij}(t))_T$ and T the time support. Each time series model-based co-clustering method defines and makes use of a specific representation of the time series and probability density functions for the mixture components. While one article from Chamroukhi and Biernacki (2017) uses a density based on a piecewise regression model, the majority of them Slimen et al. (2018); Bouveyron et al. (2018); Schmutz et al. (2019) are based on modeling the time series using a functional PCA (fPCA) projection Ramsay and Silverman (2005). This process assumes the dataset time series can be adequately represented in a common low-space expansion basis, i.e. each x_{ij} can be expressed as $x_{ij}(t) = \sum_{s=1}^{S} c_{ijs} f_s(t)$. This projection allows to reconstruct the functional form from the discrete time series representation. Fourier basis is a common choice in the domain.

The LBM is then applied to the coefficients dataset. Applying the fPCA block-per-block, as presented in Bouveyron et al. (2018), allows to detect even the smallest signal change. This work shows good performances but cannot deal with datasets that contain irrelevant, uninformative features or in the case column clusters define different row clusters. In this situation, using the LBM forces to make compromises in the block partitioning, resulting in sub-optimal block clustering solutions. Fig. 2 illustrates this behavior with an example of univariate Gaussian Latent Block Model co-clustering. FunCLBM allows to extend the FunLBM in order to overcome this limitation.

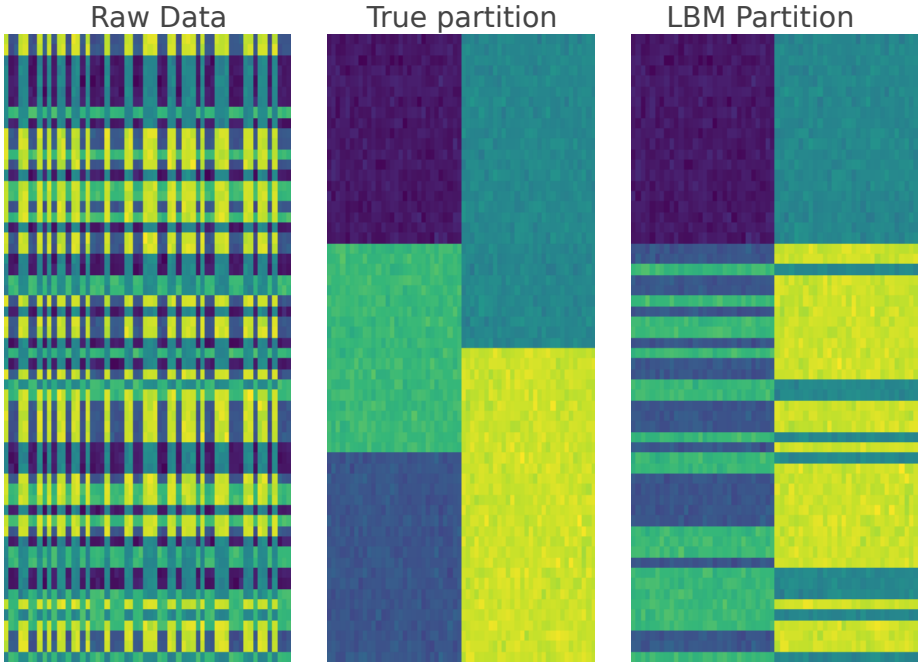

Figure 2: Several reordered views of a conditionally partitioned univariate Gaussian dataset: the original "random" view, the optimal "true" partition and the best partition that can be produced with a standard Gaussian LBM approach

3 Functional Conditional Latent Block Model

This section presents the FunCLBM model, as well as its inference and model selection strategies. The proposed approach relies on the projection of the time series in a specific space. It consists in applying a PCA on the representation of the series in the frequency domain.

3.1 Representation with interpolated Fourier transform

This paper focuses on working with high-dimensional time series. Using such series as is makes learning hard due to both the *curse of dimension*, tending to make all individuals equally distant from each other, and the huge quantity of noise involved. It is required to choose a more compact representation for learning. Many have been studied in the literature (Fourier, wavelets, Chebyshev, ...) each with its pros and cons. In this paper, an interpolated log-scaled Fourier periodogram representation is chosen, following what is advocated in Caiado et al. (2009). Formally, given a family of series $X = (x_{ij})_{n\times p}$, each x_{ij} with its own time length $l_{ij} = |x_{ij}|$ (its number of discrete time points), the first step is to compute the Fourier periodograms $P = (p_{ij})_{n\times p}$ of same length. These periodograms are not indexed over time though but over frequencies $F = (f_{ij})_{n\times p} = (\langle f_{ij}^1, \ldots, f_{ij}^{l_{ij}}\rangle)_{n\times p}$.

However, the different length of each time series makes the discrete periodogram frequencies unaligned, so that the f_{ij} values for a dataset of series are likely to be significantly different between series. Thus, in order to make the representation comparable, one needs to find a common sequence of frequencies $\widehat{f}$ from F which is not obtainable by selecting a subset of each series frequencies. A possible solution chosen for this paper is to obtain $\widehat{f}$ by computing the sample average gap $\widehat{\Delta f}$ between two consecutive frequencies over all time series. Then, choosing a desired representation length $\widehat{l}$, it is possible to build $\widehat{f} = \langle 0, \widehat{\Delta f}, 2\widehat{\Delta f}, \ldots, (\widehat{l}-1)\widehat{\Delta f}\rangle$.

The final step is to obtain the periodogram values $\widehat{P} = (\widehat{p}_{ij})_{n\times p}$ of all series of X for $\widehat{f}$ frequencies, which can only be estimated, e.g. using linear or cubic interpolation techniques from P. One could be tempted to use $\widehat{P}$ directly as the compact representation for model selection. However, as advised in Caiado et al. (2009), $\log\left(z\left(\widehat{P}\right)\right) = \left(\log\left(z\left(\hat{p}_{ij}\right)\right)\right)_{n\times p}$, where z is the z-normalization function substracting the mean value and dividing by the standard deviation, is used instead to compare relative periodogram values and limit the bias towards low frequencies encountered in practice.

3.2 Model definition

Considering independent row-clusters partition for each column-cluster, it is mandatory to adapt the previous notation. Let denote K_ℓ the number of row-clusters associated to column-cluster ℓ, $1 \leq \ell \leq L$. We still denote by $w = (w_{j\ell})_{p\times L}$ the binary vector that indicates the column-cluster belonging. Given a column-cluster ℓ, the associated row-clusters partition is denoted as $z^\ell = (z_{ik}^\ell)_{n\times K_\ell}$. For convenience, we note $z_i^\ell = (z_{ik}^\ell)_{K_\ell}$ the row-cluster membership of observation i in the column cluster ℓ. The global row-partition is denoted as $z = (z_\ell)_L$. While the column mixing proportion remains the same, the row mixing proportion is now denoted $\pi = (\pi_\ell)_L$, with $\pi_\ell = (\pi_{k_\ell})_{K_\ell}$. Finally, the joint model density can be decomposed in the following:

$$\begin{aligned}\mathrm{p}(x;\theta) &= \sum_{\mathcal{Z}\times\mathcal{W}} \mathrm{p}(w;\theta)\mathrm{p}(z|w;\theta)\mathrm{p}(c|z,w;\theta)\\ &= \sum_{\mathcal{Z}\times\mathcal{W}} \prod_{j\ell} \rho_l^{w_{j\ell}} \prod_{i\ell}\prod_{k_\ell} {\pi_{k_\ell}^\ell}^{z_{ik}^l} \prod_{ij\ell}\prod_{k_\ell} \mathrm{p}(c_{ij};\theta_k^\ell)^{z_{ik}^\ell w_{jl}},\end{aligned}$$

with $\mathrm{p}(c_{ij};\theta_k^\ell) = \mathrm{p}(c_{ij}|w_j, z_i^l)$ being the density of the block (k_ℓ, ℓ) with parameters θ_k^ℓ. This density is the one of a multivariate gaussian model on the projections of the interpolated periodogram coefficients into a low-dimensional subspace. This density is parameterized by three elements:

- A matrix A_k^ℓ of size $m \times d$ which defines the linear transformation of the periodogram (of size m) in the lower-dimension subspace (of size d).
- The m-dimension mode μ_k^ℓ.
- Σ_k^ℓ, the $d \times d$ covariance matrix in that subspace.

The complete set of parameter θ=$(\pi, \rho, (\theta_k^\ell)_{K_\ell\times L})$ is inferred with a dedicated SEM-Gibbs algorithm.

3.3 Inference with SEM-Gibbs algorithm

Using the SEM algorithm is a popular practice in the model-based clustering framework. As described in section 2.1, the SEM implies that the block component parameters are updated based on sampled observations. In the co-clustering case, there is an additional constraint: the direct computation of the block belonging is intractable. A popular solution from Keribin et al. (2010) is to use a Gibbs sampler that alternatively draws the cluster belongings in one dimension conditionally to the other. Starting from an initial parameter state θ^0 and an initial column partition w^0, the SEM-Gibbs alternates between these two steps:

1. SE step:
   - For each column partition ℓ and each row i, draw the associated row cluster belonging $z_i^{\ell\,(q+1)} \sim \mathcal{M}(1, \tilde{z}_{i1}^{\ell}, \ldots, \tilde{z}_{iK_\ell}^{\ell})$, with
   $$\begin{aligned}\tilde{z}_{ik}^{l} &= \mathrm{p}\left(z_{ik}^{\ell} = 1 | c_{i\cdot}, w^{(q)}; \theta^{(q)}\right) \\ &= \frac{\pi_k^{\ell\,(q)} f_k^{\ell}\left(c_{i\cdot}|w^{(q)};\theta^{(q)}\right)}{\sum_{h=1}^{K_\ell} \pi_h^{\ell\,(q)} f_h^{\ell}\left(c_{i\cdot}|w^{(q)};\theta^{(q)}\right)},\end{aligned}$$
   where $c_{i\cdot} = (c_{ij})_{0\leq j\leq p}$ and f_k^{ℓ} the density of the row:
   $f_k^{\ell}\left(c_{i\cdot}|w^{(q)};\theta^{(q)}\right) = \prod_j \mathrm{p}\left(c_{ij};\theta_{k\ell}^{(q)}\right)^{w_{j\ell}^{(q)}}$.
   - For each column j, draw the column cluster belonging $w_j^{(q+1)} \sim \mathcal{M}(1, \tilde{w}_{j1}, \ldots, \tilde{w}_{jL})$, with
   $$\begin{aligned}\tilde{w}_{j\ell} &= \mathrm{p}\left(w_{j\ell} = 1 | c_{\cdot j}, z^{(q+1)}; \theta^{(q)}\right) \\ &= \frac{\rho_\ell^{(q)} g_\ell\left(c_{\cdot j}|z^{(q+1)};\theta^{(q)}\right)}{\sum_{r=1}^{L} \rho_r^{(q)} f_r\left(c_{\cdot j}|z^{(q+1)};\theta^{(q)}\right)},\end{aligned}$$
   where $c_{\cdot j} = (c_{ij})_{0\leq i\leq n}$ and g_ℓ is the density of the column $c(.j)$ given the multiple row partition:
   $$g_\ell\left(c_{\cdot j}|z^{(q+1)};\theta^{(q)}\right) = \prod_{ik} \mathrm{p}\left(c_{ij};\theta_{k\ell}^{(q)}\right)^{z_{ik}^{l}\,^{(q+1)}}$$
2. M Step: given the sampled block partition, and denoting by c_k^{ℓ} the observations belonging to block (k,ℓ), the mixture proportions are updated by:
   - ${\pi_{k_\ell}}^{(q+1)} = \frac{1}{n}\sum_i {z_{ik}^{l}}^{(q+1)}$, $0 \leq \ell \leq L$,
   - ${\rho_\ell}^{(q+1)} = \frac{1}{p}\sum_j {w_{j\ell}}^{(q+1)}$
   - A_k^{ℓ} the loadings matrix produced by the block-wise PCA of c_k^{ℓ}, i.e. the $m \times d$ matrix containing the d eigenvectors with highest eigenvalues.
   - μ_k^{ℓ} and Σ_k^{ℓ} the mean and covariance matrices in the lower-dimensional subspace:
   $$\begin{aligned}\mu_k^{\ell} &= \frac{1}{n_k^{\ell\,(q+1)}} \sum_{i,j} {z_{ik}^{\ell}}^{(q+1)} w_{jl}^{(q+1)} v_{ij} \\ \Sigma_k^{\ell} &= \frac{1}{n_k^{\ell\,(q+1)}} \sum_{i,j} {z_{ik}^{\ell}}^{(q+1)} w_{jl}^{(q+1)} \left(v_{ij} - \mu_k^{\ell}\right)\left(v_{ij} - \mu_k^{\ell}\right)^T,\end{aligned}$$
   with $v_{ij} = c_{ij} A_k^{\ell}$, and $n_k^{\ell\,(q+1)} = \sum_i \sum_j z_{ik}^{(q+1)} w_{j\ell}^{(q+1)}$

This algorithm is run for a given number of iterations, or until a relative convergence threshold is reached. Choosing a good initialization state is crucial in order to ensure the good behaviour of the algorithm. It is a well-known dilemma in model-based clustering, subject of several works Blömer and Bujna (2013); Baudry and Celeux (2015).

Several methods are often considered: populating components with a small random sample of the observations, shuffling the column and block partitions, or using another clustering algorithm to get a good initial starting point. In section 4 these different initializations are experimented. Independently from the method, taking the result with highest likelihood among several runs is an agreed-upon strategy.

3.4 Model Selection

With a good initialization choice, SEM-Gibbs may converge to a solution for a given clustering structure, i.e. a column cluster number L and a set of row clusters numbers $K = (K_\ell)_L$. Several criteria have been developed to address the model selection problem. In this work, we propose a dedicated criterion based on the Integrated Classification Likelihood (ICL) Biernacki et al. (2000). Initially developed for Gaussian Mixture Model Selection, extended by Lomet (2012) to co-clustering and in Bouveyron et al. (2018) to functional co-clustering, we propose the following extension to functional conditional co-clustering:

$$\begin{aligned}\mathrm{ICL}(K, L) = \log \mathrm{p}(\boldsymbol{x}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}; \hat{\theta}) - \frac{L-1}{2} \log p \\ - \frac{1}{2} \sum_{\ell} ((K_\ell - 1) \log n) - \frac{\sum_{\ell,k} \nu_k^\ell}{2} \log(np),\end{aligned}$$

where ν_k^ℓ is the component parameter number of block (k, ℓ). This score penalizes the log-likelihood with a function of the number of parameters. The best model is the one maximizing this score. In the co-clustering case, finding the best structure can be done by an exhaustive grid search, we will see in the experiments that this strategy is not suitable to FunCLBM.

4 Experiments on Synthetic Data

In order to test the capabilities of FunCLBM, experiments are first conducted on a simulated dataset. These experiments help us check that the model is suited and that the SEM-Gibbs algorithm behaves well in a controlled environment.

4.1 Simulated dataset

The first experiment is conducted on a dataset sampled from a known generative model. The objective is to check the behavior of FunCLBM, its initialization and model selection strategies. The dataset is generated by sampling around one of several "prototypes" denoted (ϕ_k^ℓ) and which represents the components modes in the original space. For each block (k, ℓ), several time series are drawn following $\mathcal{N}\left(\phi_{k\ell}(t + t_s), s^2\right)$ with $s = 0.02$ and t_s a random shift $\sim \mathcal{N}\left(0, s^2\right)$. These modes are depicted in Fig. 3 according to the dataset structure.

In the experiments, the quality of the estimated block partition is compared to the known generative partition, based on the Adjusted Rand Index (ARI). This 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. In our particular context, we compare the obtained partition based on three aspects: the column cluster partition, the rows cluster partitions (made of the binning of every row cluster partition per column), and the block partition. We generate a dataset of size 90x90, with column cluster of size $(45, 15, 30)$ and row cluster sizes of respective sizes $(20, 40, 30)$, $(60, 30)$ and $(40, 50)$.

4.2 Model Adequacy

As a preliminary test, we verify that FunCLBM objective function in lower-dimensional spaces is suited to the clustering of such dataset. To do so, we compare the Log-likelihood produced through 100 launches of SEM-Gibbs to the corresponding ARI criterion and depicted in Fig. 4. We verify this relationship by computing the Pearson's correlation coefficient between the two scores, and Kendall's correlation test. We use this latter test to avoid making assumptions on ARI or Log-likelihood normality and because of the presence of ex-aequos values that can be produced if the "true state" is reached. The test results (with 95 $\%$ confidence level) are displayed in Table 1. For this dataset, the suitability of the method is attested by the strong Pearson's correlation for every partition dimension and confirmed by Kendall's correlation test p-value at 95% confidence level.

Table 1: Kendall's correlation test p-value (conf level: 0.95)

	Row	Column	Block
Pearson's correlation	0.7110134	0.8974437	0.7111690
Kendall test p-value	7.88e-18	6.091e-08	8.09e-06

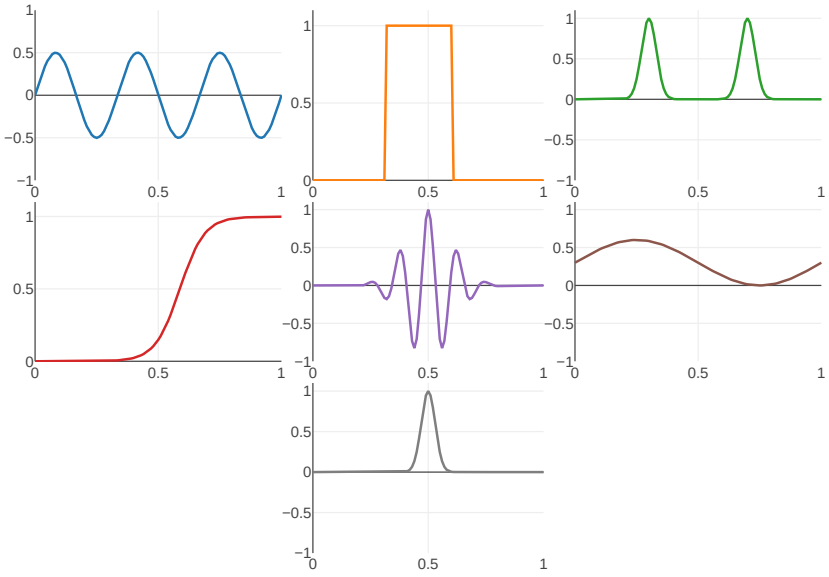

Figure 3: Prototypes used as block mode for the simulations

Figure 4: ARI versus Log-Likelihood in 100 launches of SEM-Gibbs on the simulated dataset

4.3 Initialization

The next experiments aim at evaluating the different initialization strategies. It is, once more, evaluated with the ARI criterion. We compare four strategies:

- Populate blocks with samples
- Random shuffle of the column partition, then of the row partition
- Initialize the column partition with a K-Means run on the transposed dataset, then the row partition with one K-Means run per column.

- Initialize the column partition with a model-based functional co-clustering approach.

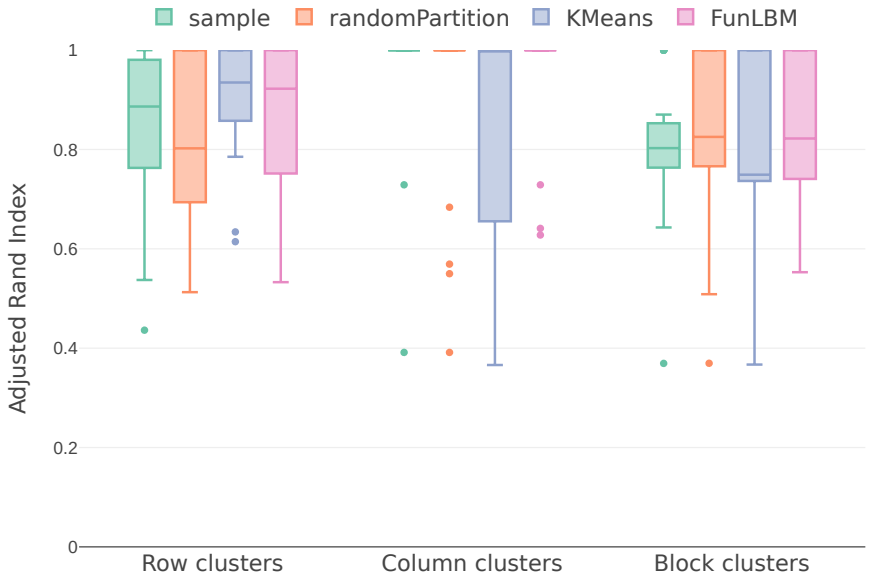

Figure 5: Results of Row, Cluster and Block partition ARI obtained with different initialization methods (median and quantile 0.9 on 30 SEM-Gibbs runs)

For the last approach, we use a modified version of the original FunLBM, closer to the FunCLBM variant: the time series are transformed into interpolated log-periodograms and component parameter updates are the same as in FunCLBM (c.f. Section 3).

In Fig. 5 are displayed the ARI obtained after launching 30 times SEM-Gibbs with each initialization method. The figure shows important differences between row and column cluster results, as expected since the model does not treat rows and column symmetrically anymore. We can also observe that the K-Means run has an unexpected behavior: while performing well on average, its results show a high dispersion for column cluster ARI. Its row cluster ARI however is slightly better than the other methods. On average, the model-based co-clustering performs well but not overwhelmingly.

On this small experimental case, the random partition seems a direct and cheap initialization strategy. Whichever initialization strategy is applied, the concurrent run of several methods allows to stabilize the results, as displayed in Fig. 7. In this experimental setup, finding the perfect structure is an easy task whenever the number of concurrent launches is higher than 4.

4.4 Model selection

The last experiments compared the initialization methods for a given choice of structure. However, the most challenging part, and also the most useful for the field expert, is the model selection strategy. In Bouveyron et al. (2018), the authors proposed the co-clustering grid search method, which requires inferring $K \times L$ models. In the clustering case, the number of components is preferably low for interpretability sake. In our case, such approach is not possible: the number of combinations is prohibitive. For a maximal number of column clusters L_M and per-column row clusters K_M, it is the number of un-ordered set of length ℓ among K_M possibilities, for $0 \leq \ell \leq L_M$, i.e. $\sum_{\ell=1}^{L_M} \binom{K_M+\ell-1}{\ell}$. The quantity is prohibitive: with $L_M = 5$, $K_M = 5$, it amounts to 251 combinations, i.e. 10 times more than in the LBM case (5x5 combinations).

In order to overcome this limitation, we propose and compare two strategies. Both are based on a different estimation of $\widehat{L}$ and then on a column-wise grid search. In the first case, $\widehat{L}$ is estimated from a standard model-based co-clustering exhaustive grid search, and in the second from a greedy algorithm. The first solution implies an exhaustive search of the $K_M \times L_M$ combinations and then $\widehat{L} \times K_M$ to produce the best number of row-clusters per column. The second one is an iterative algorithm that chooses, at each iteration, the best functional Latent Block Model between the one with

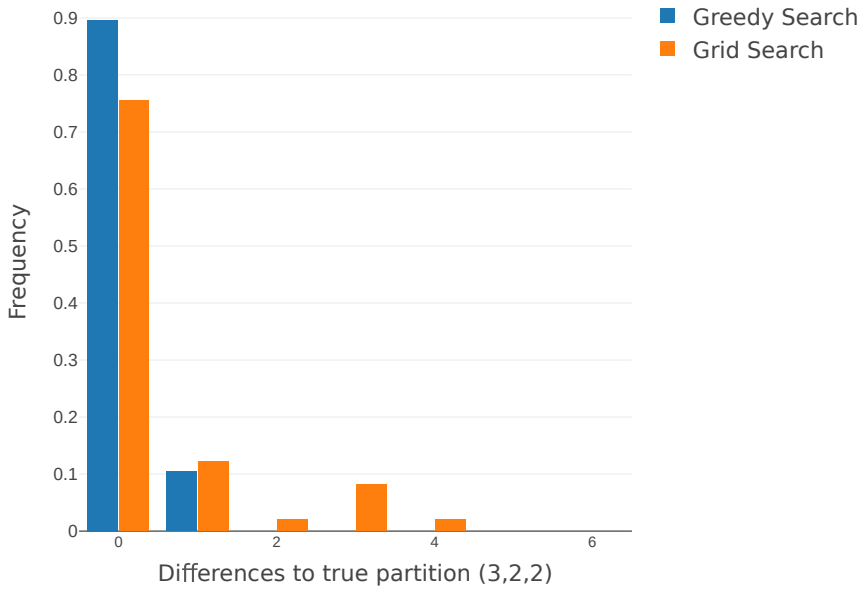

Figure 6: Results of 50 runs of each selection model strategy, in terms of differences to the generative model structure.

an added row cluster and the one with an added column cluster. The number of inferences to perform depends on the number of iterations, with upper bound $L_M + K_M$. After each construction, the candidate is finally taken as initial FunCLBM model state for a new SEM-Gibbs run.

The results of 50 launches of each method is displayed in Fig. 6, in terms of differences to the true partition. While the grid search superiority was predictable, the results illustrate the differences in results, to be compared to the computation resources required. From our perspective, the grid search approach seems better, as long as we keep the clusters number low.

5 Application to autonomous driving system validation

The Scala/Spark source code of the method is available at the github repository https://github.com/EtienneGof/FunCLBM, along with the data simulation method. The real-case data is not, however, put at disposal.

Validating an intelligent driving system is a complicated task, that can not be purely addressed with on-track tests. The numerical simulation approach circumvents the limits of these physical experiments, mainly due to the high numbers of validation check to perform to assess a system. A large scale simulation framework reproducing test conditions is intensively used to test driverless cars, producing a massive amount of time series that needs to be processed. Several aspects motivate the use of an autonomous behavior simulation platform. One of the main motivation is the physical validation cost reduction. Such validations require specific infrastructures, equipment management and maintenance, and significant human intervention to set up the experiments.

Another major disadvantage of physical testing is the impossibility to produce enough sample to prove the high reliability of a system. A validation objective may be the assessment of vehicle incident odds (e.g. $< 10^{-8}$ incidents per hour). With a classical sampling method, estimating such probability would require running prototypes over hundreds of millions of kilometers. Therefore, using a digital environment to test the different vehicles enables us to reproduce an exact experimental setting, to repeat the tests on-demand in an automated fashion in parallel of the development of the control software, as well as sample the test input parameters to assess the uncertainty of the experiments and the robustness of the cars. Even if such a large amount of real-life data were available, as is the case in some data science applications, there would be no guarantees about neither the data quality nor value. In our case, this value lies in the specific driving situation in which to test the control logic reaction. These situations might be rarely occurring in real-life driving sessions, such as emergency braking or lane departure events.

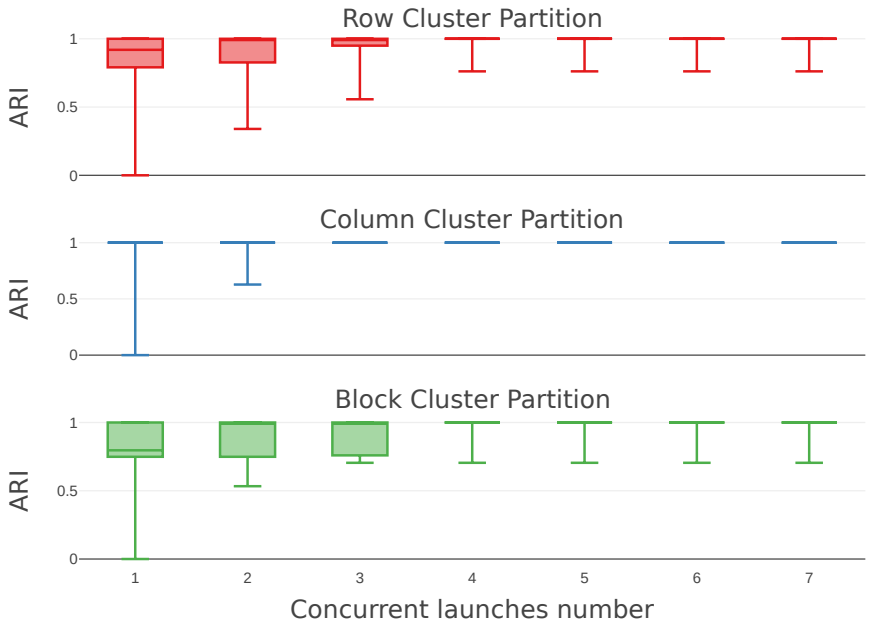

Figure 7: Best ARI obtained among several SEM-Gibbs runs (median, quantile 0.9; with variable number of concurrents)

Therefore, the use of an high-perfomance computing environment provides a mean to extensively test a driving control logic. Because of the large variety and complexity of the simulated driving situations, as well as the *possibly unknown* operating modes of the intelligent car, using a supervised approach is intractable for the massive datasets under consideration.

5.1 Use case description

In this situation, the objective is to test the reactions of a car (called Ego) equipped with the control logic. Ego runs in a straight line and starts drifting laterally towards the road side or the other lane, simulating a sleeping driver. We expect the drifting detection system to trigger the control logic, which in turn puts the car back in its line center, as an emergency maneuver. The situation is depicted in Fig. 8.

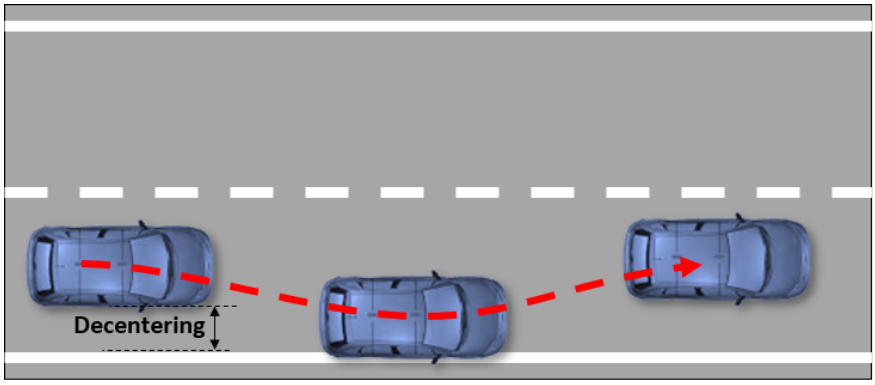

Figure 8: Use case illustration: Ego drifts from the runway's center line and cross the white line on the side of the road, before being put back in the runway center

The simulated datasets contain the data from 56 simulations, each described by 20 signals. Some signals are duplicated in order to test FunCLBM ability to regroup them, and some uninformative ones are kept on purpose.

5.2 Results

The experiments on simulated datasets lead us to choose the following setup for the real case analysis: initialization is always performed by sampling column and row cluster partitions, and the FunLBM grid search approach is applied for model selection. Each combination is tested with 30 concurrent runs.

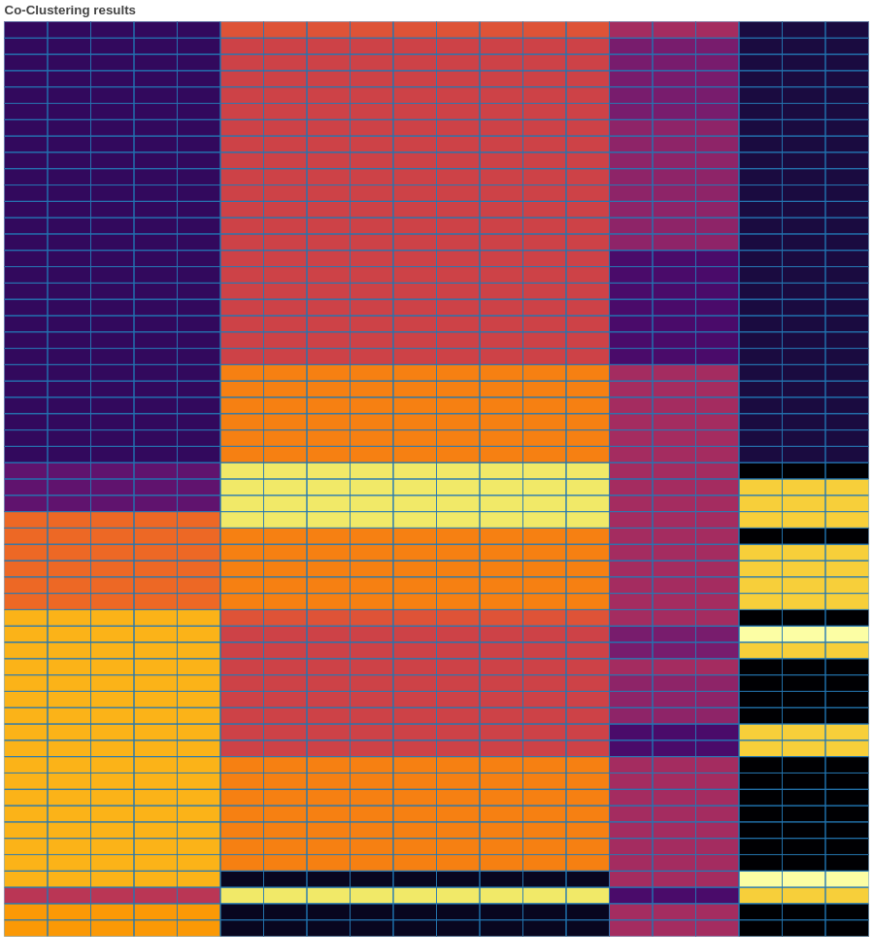

Figure 9: Final structure obtained on real case dataset

The final clustering structure is presented in Fig.9. It consists of 4 column clusters, each one with a different number of row clusters: (6x5x4x4). Due to constraints on article length, each of these 19 clusters cannot be analyzed in-depth here. However we give the main insights below.

The first column cluster groups the following features: Ego's current lateral lane position (continuous), Ego's current lane index (discrete), type of the lane on Ego's right side (discrete), and type of lane on Ego's left side (discrete). The last two signals seem to be wrongly clustered at first sight, but are in fact redundant Ego's position, as they uniquely identify Ego's current lane index. Interestingly, this first column cluster therefore gathers every features related to the position of Ego.

The conditional row partitioning in this column cluster is also interesting: the partition of Ego's position signal is represented in Figs. 10, 11. The clusters adequately gather simulations that share the same behavior. In Fig. 10 case, the control logic is activated and the car is recentered in its lane, and then repeatedly bounces back on the exact same road markings. In Fig. Fig 12 case, the decentering happens later, and the car bounces once before changing direction and going straightforwardly to the other side of the road. In Fig 12 scenario, the car bounces only once and either goes to other side of the road of comes back after a large drift.

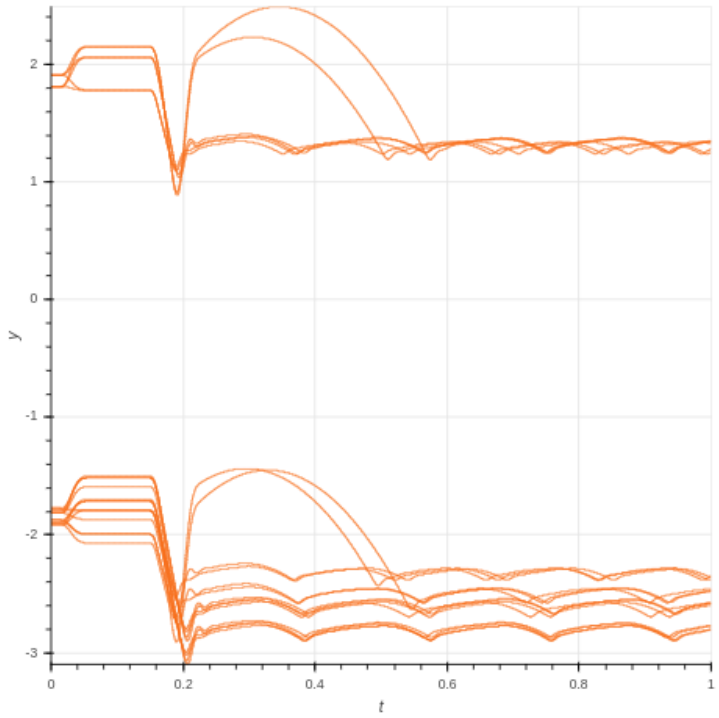

Figure 10: Ego lateral position in Block Cluster (5,1)

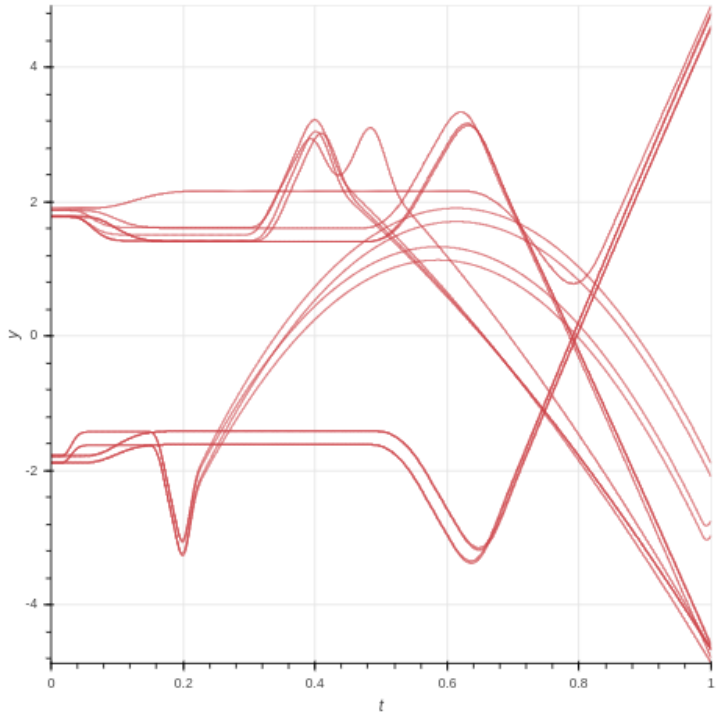

Figure 11: Ego lateral position in Block Cluster (2,1)

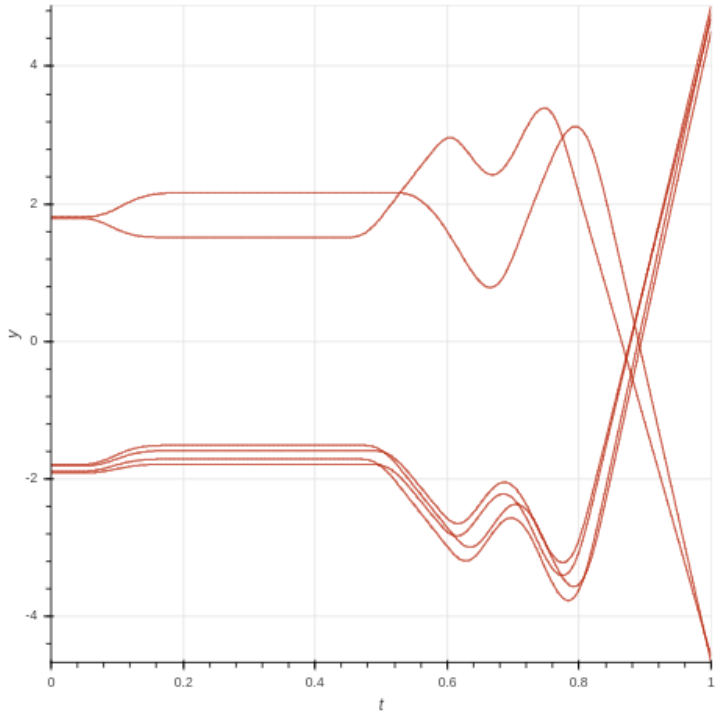

Figure 12: Ego lateral position in Block Cluster (3,1)

In the second, and largest, column-cluster can be found the uninformative signals, that either give constant values (vehicle length, width, distance between wheels, road bend radius) or increasing linearly (distance to origin). Fig. 13 and Fig. 14 illustrates some of these signals.

The third column-cluster regroups two other interesting features: the rectangular function indicating the activation of the control logic, and the changes in Ego's heading. While the first column-cluster was grouping position, this one gathers the control features. Fig.15 shows the content of subcluster (3, 3) which illustrates the relationship between them. Overall, every set of duplicate features have been correctly grouped together.

This conditional clustering partition shows, in conclusion, that the FunCLBM approach has correctly discriminated uninformative signals, while creating meaningful clusters of features (position and leverage). In each column-clusters, the observatins are also informative and provide good insights of the dataset content.

6 Conclusions and Future Work

This paper describes FunCLBM, a model-based method which addresses the problem of clustering multivariate time series in multi-views. This new model enables regrouping redundant signals, discriminating uninformative ones and provides the user with multiple clustering views of a multivariate time series dataset.

The time series are transformed into interpolated log-periodogram before being projected into low-dimensional space. This space is adapted to each block-cluster, and updated at each iteration of a SEM-Gibbs algorithm for model inference.

Several initialization methods and model selection strategies are proposed and experimented on a simulated dataset, which shows the model adequacy and give insights on the most interesting implementation strategies. Finally, we apply the method to a real-case dataset from the autonomous driving system validation domain. In this application, FunCLBM has been able to simultaneously discriminate groups of signals and produce meaningful driving behavior clusters. These results shows the usefulness of the model and the effectiveness of the initialization and model selection strategy.

The FunCLBM approach was applied here to an autonomous driving context, however we are confident that it can be used in many other domains. Several improvements are being considered in order to facilitate its use. The model selection, for instance, can become computationally expensive when the number of observations and signals increases. Similarly, the initialization can become problematic for higher numbers of clusters. In order to overcome these

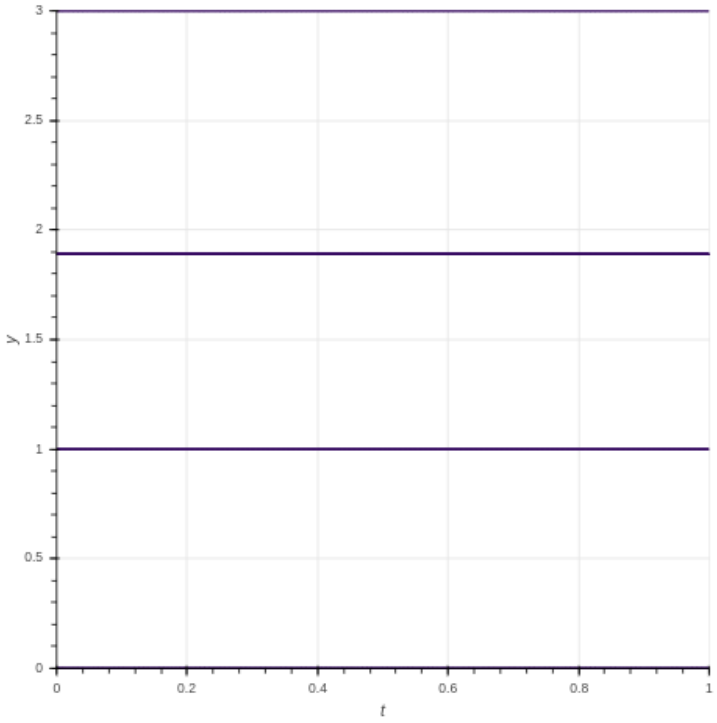

Figure 13: Uninformative signals in Block Cluster (2,2): linearly increasing feature (vehicle's width, length, headlights activation..)

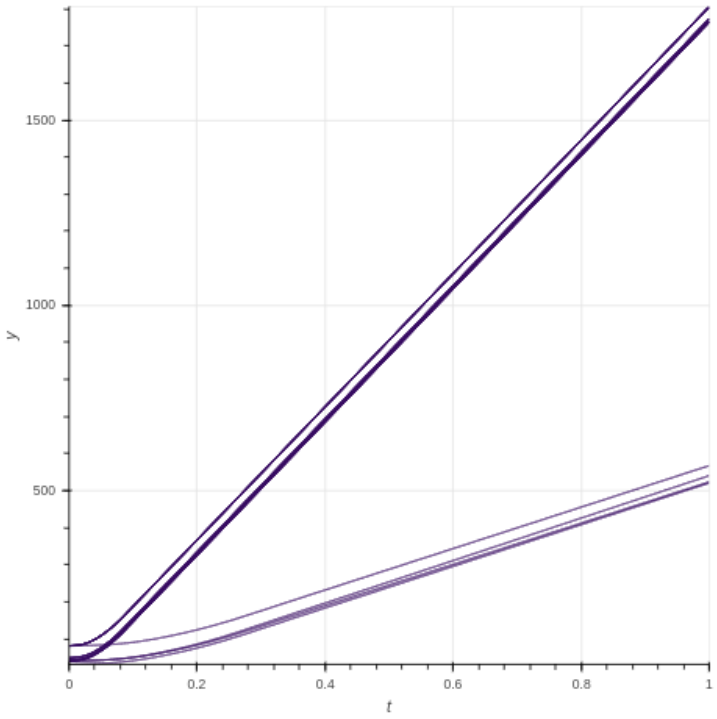

Figure 14: Uninformative signals in Block Cluster (1,2): linearly increasing feature (distance to origin)

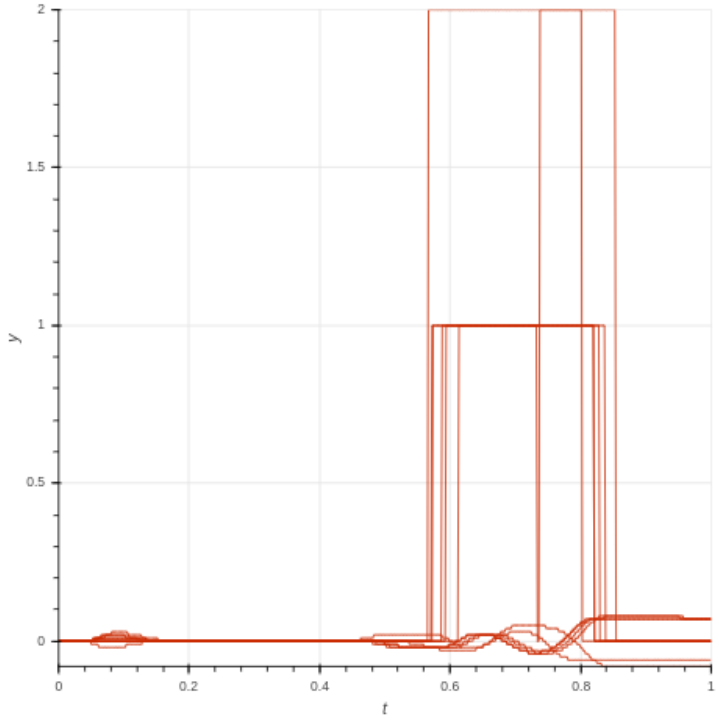

Figure 15: Control logic activation and changes in Ego's heading in Block Cluster (3,3)

constraints, we plan to investigate new initialization methods based on Importance Sampling, as well as new model selection strategies. In this context, the development of a non-parametric functional conditional latent block model seems a promising lead.

References

Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. 2015. Time-series clustering–a decade review. *Information Systems* 53 (2015), 16–38.

Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery* 31, 3 (2017), 606–660.

Jean-Patrick Baudry and Gilles Celeux. 2015. EM for mixtures. *Statistics and computing* 25, 4 (2015), 713–726.

Christophe Biernacki, Gilles Celeux, and Gérard Govaert. 2000. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE transactions on pattern analysis and machine intelligence* 22, 7 (2000), 719–725.

Christophe Biernacki, Gilles Celeux, and Gérard Govaert. 2003. Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. *Computational Statistics & Data Analysis* 41, 3-4 (2003), 561–575.

Johannes Blömer and Kathrin Bujna. 2013. Simple methods for initializing the em algorithm for gaussian mixture models. *CoRR* (2013).

Charles Bouveyron, Laurent Bozzi, Julien Jacques, and François-Xavier Jollois. 2018. 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 (2018), 897–915.

Charles Bouveyron and Julien Jacques. 2011. Model-based clustering of time series in group-specific functional subspaces. *Advances in Data Analysis and Classification* 5, 4 (2011), 281–300.

Jorge Caiado, Nuno Crato, and Daniel Peña. 2009. Comparison of times series with unequal length in the frequency domain. *Communications in Statistics—Simulation and Computation®* 38, 3 (2009), 527–540.

Faicel Chamroukhi and Christophe Biernacki. 2017. Model-Based Co-Clustering of Multivariate Functional Data.

Faicel Chamroukhi, Allou Samé, Gérard Govaert, and Patrice Aknin. 2010. A hidden process regression model for functional data description. application to curve discrimination. *Neurocomputing* 73, 7-9 (2010), 1210–1221.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* 39, 1 (1977), 1–22.

Inderjit S Dhillon. 2001. 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*. 269–274.

Gérard Govaert and Mohamed Nadif. 2003. Clustering with block mixture models. *Pattern Recognition* 36, 2 (2003), 463–473.

Gérard Govaert and Mohamed Nadif. 2013. *Co-clustering: models, algorithms and applications*. John Wiley & Sons.

Julien Jacques and Christophe Biernacki. 2018. Model-based co-clustering for ordinal data. *Computational Statistics & Data Analysis* 123 (2018), 101–115.

Christine Keribin, Gérard Govaert, and Gilles Celeux. 2010. Estimation d'un modèle à blocs latents par l'algorithme SEM.

Aurore Lomet. 2012. *Sélection de modèle pour la classification croisée de données continues*. Ph.D. Dissertation. Compiègne.

Matthieu Marbac, Vincent Vandewalle 2019. A tractable multi-partitions clustering. In *Computational Statistics & Data Analysis*. 167–179.

JO Ramsay and BW Silverman. 2005. Principal components analysis for functional data. *Functional data analysis* (2005), 147–172.

Amandine Schmutz, Julien Jacques, Charles Bouveyron, Laurence Chèze, and Pauline Martin. 2019. Co-clustering de courbes fonctionnelles multivariées.

Yosra Ben Slimen, Sylvain Allio, and Julien Jacques. 2018. Model-based co-clustering for functional data. *Neurocomputing* 291 (2018), 97–108.

Nikos Vlassis and Aristidis Likas. 2002. A greedy EM algorithm for Gaussian mixture learning. *Neural processing letters* 15, 1 (2002), 77–87.

Dongkuan Xu, Wei Cheng, Bo Zong, Jingchao Ni, Dongjin Song, Wenchao Yu, Yuncong Chen, Haifeng Chen, and Xiang Zhang. 2019. Deep co-clustering. In *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 414–422.

Xu Yang, Cheng Deng, Feng Zheng, Junchi Yan, and Wei Liu. 2019. Deep spectral clustering using dual autoencoder network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4066–4075.