Multi-Block Clustering and Analytical Visualization of Massive Time Series from Autonomous Vehicle Simulation

Etienne Goffinet

UNIVERSITÉ PARIS XIII - SORBONNE PARIS NORD

École Doctorale Sciences, Technologies, Santé - Galilée

Multi-Block Clustering and Analytical Visualization of Massive Time Series from Autonomous Vehicle Simulation

Clustering Multi-Blocs et Visualisation Analytique de Données Séquentielles Massives Issues de Simulations du Véhicule Autonome

---

THÈSE DE DOCTORAT
présentée par

Etienne GOFFINET

Laboratoire d'Informatique de Paris Nord
LIPN - UMR CNRS 7030
Équipe A3 : Apprentissage Artificiel & Applications

pour l'obtention du titre de

Docteur en Informatique

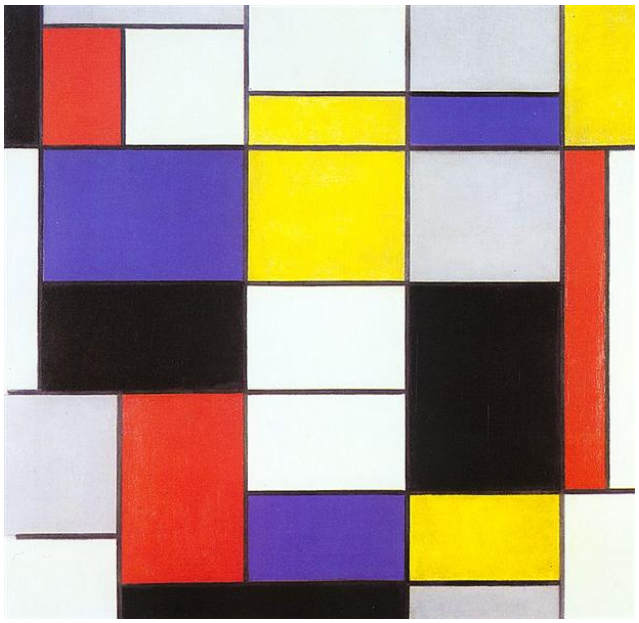
soutenue le 15 décembre 2021 devant le jury d'examen composé de :

NADIF Mohamed	Professeur	Université Paris Descartes	Rapporteur
SAMÉ Allou	CR-HDR	IFSTTAR	Rapporteur
BIERNACKI Christophe	Professeur	INRIA / Université Lille 1	Examinateur
CÉRIN Christophe	Professeur	Université Sorbonne Paris Nord	Examinateur
LEBBAH Mustapha	MCF-HDR	Université Sorbonne Paris Nord	Directeur
AZZAG Hanane	MCF-HDR	Université Sorbonne Paris Nord	Co-directrice
GIRALDI Loïc	CR	CEA	Invité
OBREBSKI Mathias	Data Scientist	Groupe Renault	Invité

Etienne Goffinet
Multi-Block Clustering and Analytical Visualization of Massive Time Series from Autonomous Vehicle Simulation
Clustering Multi-Blocs et Visualisation Analytique de Données Séquentielles Massives Issues de Simulation du Véhicule Autonome
Directeurs : M. Mustapha Lebbah, Mme. Hanane Azzag
Rapporteurs : M. Mohamed Nadif, M. Allou Samé
Examinateurs : M. Christophe Biernacki, M. Christophe Cérin

Université Sorbonne Paris Nord
Team A3 : Apprentissage Artificiel & Applications
Laboratoire d'Informatique de Paris Nord (LIPN — UMR CNRS 7030)
École Doctorale Galilée (ED 146)
99 avenue Jean-Baptiste Clément
93430 Villetaneuse

Groupe Renault
Direction de l'Ingénierie de l'Alliance Renault Nissan
$\rightarrow$ Autonomous Driving Simulation & Virtual Reality Center
$\longrightarrow$ Autonomous Driving / Advanced Driving Assistance systems Validation
Superviseurs : M. Pascal Remusan, M. Loïc Giraldi, M. Mathias Obrebski
1 Avenue du Golf
78280 Guyancourt



Multi-Block representation
Piet Mondrian (1923). Composition A.
Galleria Nazionale d'Arte Moderna e Contemporanea, Rome, Italy.

Abstract

Advanced driver-assistance systems (ADAS) development remains one of the biggest challenges car manufacturers must tackle to provide safe driverless cars. The reliable validation of these systems requires assessing their reaction's quality and consistency in a broad spectrum of driving scenarios. In this context, Groupe Renault uses large-scale simulation systems, which accurately reproduce the physical driving conditions and produce large quantities of high-dimensional time series data. The role of the ADAS developer is to explore these datasets to determine precisely the capabilities of the driving assistance system under test and, if necessary, to refine its design.

A simulated dataset can contain up to several hundreds of thousands of simulations for a given use case, described by several hundreds variables. With datasets of such dimensions, the expert's work requires a great deal of field knowledge and involves a time-consuming exploration process.

This thesis objective is to produce algorithms and tools to help explore, structure and analyze these massive simulation datasets. We propose four probabilistic approaches for time series clustering, each based on a specific datasets structure hypothesis.

The first contribution of this thesis is built on a scenario-based construction hypothesis and performs a dictionary-based classification of univariate time series. This method uses an existing piecewise polynomial model to segment the time series independently, then constructs a driving pattern dictionary. This dictionary is used to recode the time series into chains of categories, which are then partitioned with a hierarchical clustering algorithm.

The subsequent contributions focus on the analysis of multivariate datasets with multi-blocks models that simultaneously discriminate the simulations and the variables based on their distributions and row-partitions. These Bayesian Non-Parametric models natively integrate a model selection, which enables automatic model resizing with respect to the datasets contents.

Finally, the usefulness of these contributions is illustrated with several applications on industrial use cases: the validation of emergency braking, lane-keeping and emergency avoidance systems.

Résumé

Le développement de systèmes d'aide à la conduite demeure un défi technique pour les constructeurs automobiles. La validation de ces systèmes nécessite de les éprouver dans un nombre considérable de contextes de conduites. Pour ce faire, le Groupe Renault a recourt à la simulation massive, qui permet de reproduire précisément la complexité des conditions physiques de conduite et produit des jeux de données de séries temporelles en grandes dimensions. Le rôle de l'expert métier est alors d'explorer ces données pour déterminer précisément les capacités du système d'aide à la conduite étudié et, si nécessaire, de raffiner sa conception.

Un jeu de données simulées peut contenir jusqu'à plusieurs centaines de milliers de simulations, décrites par plusieurs centaines de variables. Face à de telles dimensions, le travail du développeur requiert une grande expertise, et implique un travail d'investigation minutieux et chronophage.

L'objectif de cette thèse est de produire des algorithmes et outils d'aide à l'exploration, la structuration et l'analyse de ces jeux de données issues de simulation massive. Basées sur des hypothèses de construction argumentées, nous proposons trois approches probabilistes de classification non supervisée de séries temporelles, adaptées à des jeux de données temporelles univariées et multivariées.

La première contribution de cette thèse se base sur une hypothèse de construction par scénario pour réaliser une classification de séries temporelles univariées. Cette méthode réalise une segmentation individuelle de chaque série temporelle, puis partitionne l'ensemble des segments extraits pour construire un dictionnaire de 'prototypes' de régimes de conduite. Ce dictionnaire permet de recoder les séries temporelles en séquences de catégories, qui sont alors partitionnées par une méthode de classification hiérarchique ascendante.

La suite des contributions se concentre sur l'analyse de jeux de données multivariées, grâce à des modélisations multi-blocs qui regroupent les différentes variables en fonction de leur distribution et de leur partition individuelle. Ces modèles Bayésiens Non-Paramétriques intègrent nativement une sélection de modèle qui permet d'adapter automatiquement la dimension du modèle au contenu des jeux de données étudiés.

La pertinence de ces contributions est illustrée par des applications issues des cas d'usage industriels Renault : la validation de systèmes de freinage d'urgence, d'aide au maintien dans la voie, et d'évitement d'urgence.

Remerciements

Préparer une thèse de doctorat sur un sujet aussi passionnant est une grande chance, et a été un réél plaisir. Je profite de ces lignes pour mettre des noms sur cette chance et remercier ceux qui ont permis ce parcours.

Je souhaite tout d'abord remercier chaleureusement mes superviseurs académiques, Mustapha Lebbah et Hanane Azzag, pour la confiance qu'ils m'ont accordée, leur support bienveillant et leur précieux conseils. Je remercie également Pascal Remusan, Loïc Giraldi et Mathias Obrebski pour leur encadrement au sein du Groupe Renault. Je tiens à remercier profondément Loïc pour son soutien scientifique et ses conseils, qui furent de rééls moteurs de la thèse.

Je remercie Mohamed Nadif et Allou Samé d'avoir accepté le rôle de rapporteur et qui ont permis, par leur retours, d'améliorer le contenu de la thèse. Merci également à Christophe Biernacki et Christophe Cérin d'avoir accepté d'en être les examinateurs. Par ailleurs, je dois des remerciements spéciaux à Christophe Biernacki, à qui je dois mes débuts dans la recherche et dans le domaine des modèles probabilistes, ainsi que cette orientation en thèse. Ayant accompagné une partie de ce parcours, je suis heureux que tu soies présent pour en faire l'examination.

Je remercie mes collègues de Renault ainsi que ceux du laboratoire, pour leurs encouragements, ainsi que pour les riches moments de collaborations et d'échange. Je pense en particulier aux échanges en visio qui ont été plus que salutaires lors des confinements..

Je ne peux pas omettre de remercier profondément Monsieur Lobry, à qui j'attribue l'origine de mon orientation vers les mathématiques et du parcours qui m'a mené jusqu'à cette thèse. Ce parcours représente déjà 12 ans, ce qui, dans les comptes d'apothicaires, est déjà une sacrée somme de bonheur. Je tiens également à remercier Monsieur Vincent, pour le rôle semblable qu'il a joué dans mon orientation vers l'analyse de données.

Pour conclure, je remercie ma famille et amis pour leur soutien dans cette période mouvementée.

Je dédie cette thèse à Céline et au très récent Victor, dont les premiers gazouillis ont accompagné et égayé l'écriture de ces lignes.

Contents

Introduction

> *Increasingly, connected technologies and autonomous driving will allow users to choose between driving and being driven. This fact opens the way for new scenarios for mobility. That's our next revolution.*
>
> **— Guillaume Eurin**
> Director of Autonomous Vehicle
> and ADAS Development.
> *Groupe Renault* Annual Report (2019-2020)

This thesis is a bi-party research project in collaboration between the car manufacturer *Groupe Renault*[1] and the computer science lab *LIPN*[2] (Laboratoire d'Informatique de Paris Nord, UMR CNRS 7030) at *Université Sorbonne Paris Nord*[3]. This chapter describes the industrial context that motivated this research agreement: the advanced driver assistance systems (ADAS) validation and its operational constraints. This description details the ADAS role in a vehicle and why their development is essential for the car manufacturer. It also explains the necessity of using massive simulation for validation and the nature of the data analysis work associated with the simulated data sets. The last part of this introduction presents the thesis contributions that were designed to address the datasets operational constraints, and the associated publications, before concluding with the manuscript plan.

Advanced Driver-Assistance Systems

Advanced Driver assistance systems are on-board electronic systems that assist with specific driving tasks. From the first Anti-lock Braking System (ABS) and Cruise Control systems developed in the 1950s, to the very recent Automatic Highway Pilot

[1] https://www.renaultgroup.com/
[2] https://lipn.univ-paris13.fr/
[3] https://www.univ-paris13.fr/

functionality, the ADAS development has been associated to the automotive market growth throughout its history. Every modern car contains some of these systems.

Some systems are simple information or warning providers (e.g., seat belt reminder, navigation, ...), others actively assist the user in everyday driving tasks (e.g., adaptive cruise control, automated lighting,...). Some others bring vital help in emergencies situations (e.g., emergency braking or lane-keeping, ...).

The ADAS can be classified in driving automation levels on a scale designed by the Society of Automotive Engineers (SAE), from level 0 to 5. This scale, displayed in Fig. 0.1, also represents the roadmap toward full Autonomous Driving, with ADAS serving as keystones. The following list describes three existing or in-development ADAS applications:

- Autonomous Emergency Breaking (AEB) - Level 0. While driving, an emergency brake is triggered when a target (pedestrian, car, truck, bike, ...) is detected moving at high speed towards the equipped car (called *Ego*).

- Lane Keeping Assist (LKA) - Level 1. While driving, the system detects an unintentional slow drift of Ego towards another lane or the roadside and corrects the trajectory.

- Valet Parking - Level 4 (Under development). Inside a parking lot, this automated parking system drives the vehicle, finds a parking spot, and performs the parking maneuver.

From the driver perspective, the ADAS enhance vehicle safety, comfort and efficiency. On a broader level, they are also meant to extend mobility, provide assistance for the elderly and reduce traffic congestion [SS15]. From a marketing perspective, they are selling points that allow car manufacturers to demonstrate their know-how and stand out in a highly competitive automotive market. In a recent research report [Res21], the market research and consulting organization *Precedence Research* estimated the global ADAS market size at USD 32.3 billion in 2019 and forecasted that it would reach USD 142 billion by 2027.

In the perspective of equipping millions of cars worldwide, driver and pedestrian safety is the first concern of the ADAS developers. Therefore, implementing rigorous validation processes is necessary to verify the ADAS performances in all driving situation.

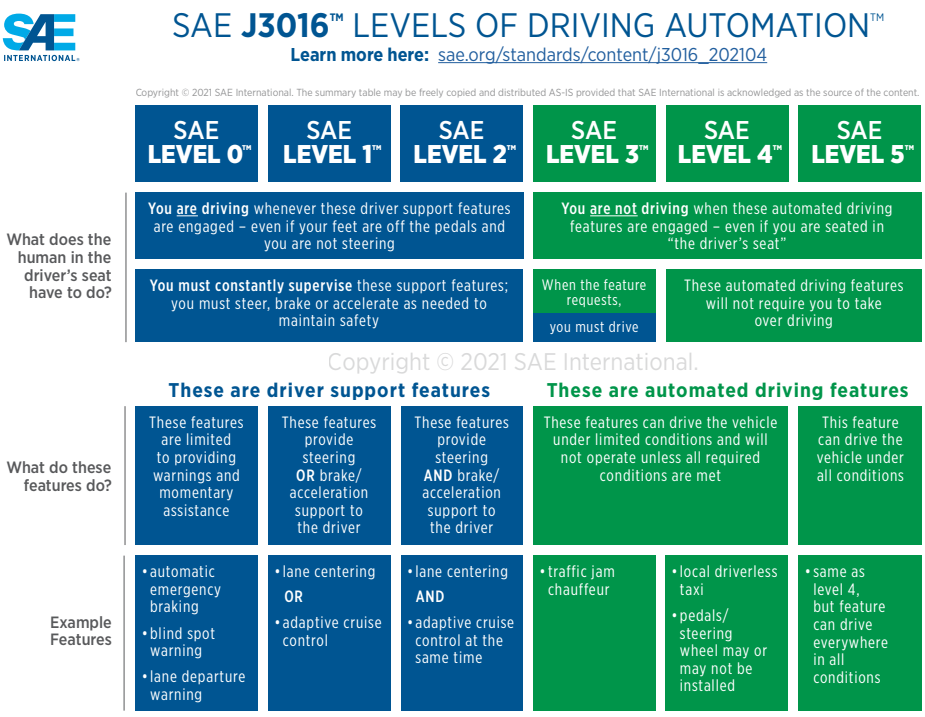


Fig. 0.1.: SAE automation scale (source: SAE).

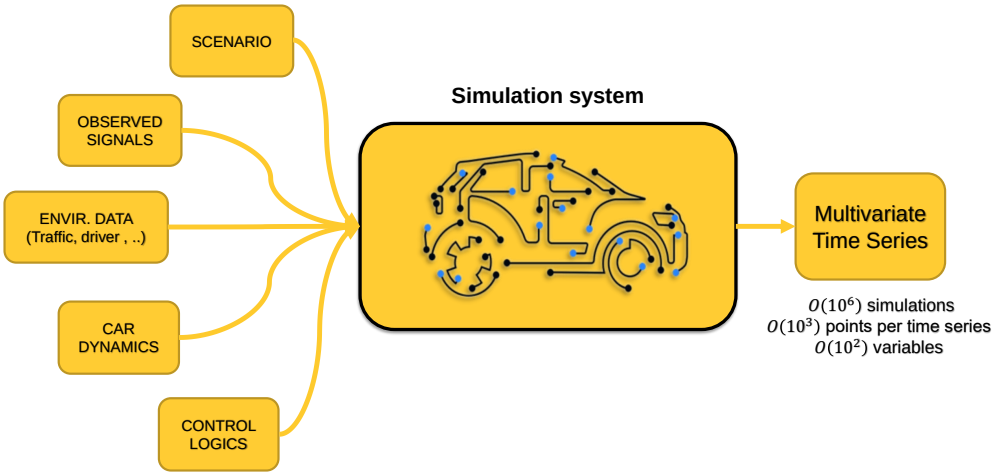


Fig. 0.2.: Simulation System Representation.

Validation

The validation process objective is to assess the ADAS conformity to technical specifications, established by Groupe Renault in accordance with car safety evaluation agencies recommendations (e.g., *Euro NCAP* in Europe).

In this context, Groupe Renault has specified a minimum reliability level of 10^{-8} casualties per hour to validate a LEVEL 4 ADAS system, which is 10 times less than the average rate of fatalities per hour on European highways, and 100 times less than the average accident rate on all types of European roads. It would require at least 2×10^{10} kilometers of driving [Wau18], equivalent to 100 years of road testing. This reliability level must be reached for every vehicles, every ADAS and every traffic regulation context. In addition, the ADAS developers must also ensure that the ADAS are inter-operable and that their combination does not provoke side effects. Given this complexity, it is impossible to test every scenario with only physical driving test on-track or open roads.

Groupe Renault has made the strategical choice to invest in massive driving simulation technology to enhance the ADAS development and validation process. The simulation systems mimic car driving conditions based on a driving scenario, vehicle physics, driver behavior, and interactions with a configurable environment. The simulation quality, i.e., its capacity to reproduce real-life driving conditions faithfully, is methodically monitored to ensure the validation process relevance. The simulation system is illustrated in Fig. 0.2.

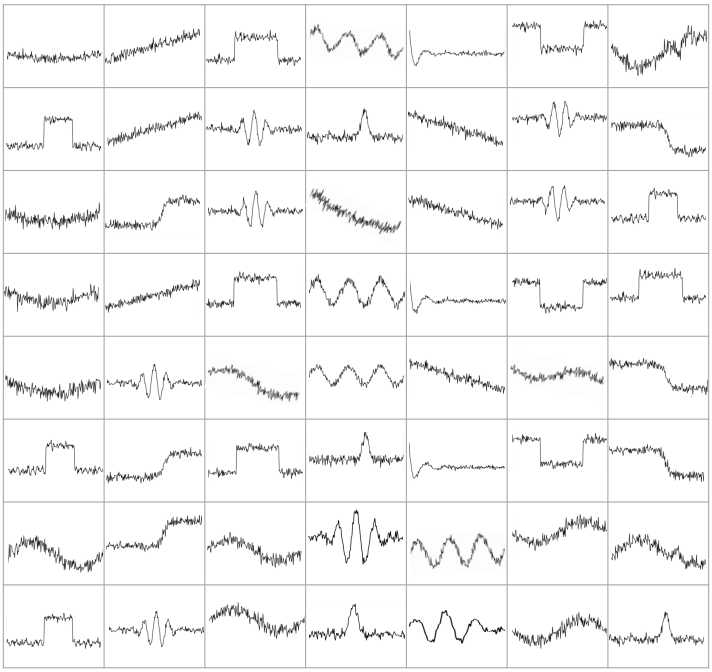


Fig. 0.3.: Illustration of the simulation dataset contents. The dataset can be represented as a matrix, where each cell is a time series. Every time series in a given row share the same length.

The simulation outputs large amounts of information coming from the observation of the simulated variables, and aggregated as one large dataset of multivariate time series. Based on this multivariate dataset, the role of the field expert's is to discriminate the driving behaviors, detect anomalies, and compare the ADAS performances to the specifications.

Data dimensions and complexity are considerable: for a given use case, the number of simulations can be as large as 10^6, described by 10^3 of simulated sensors, each recording 10^4 time steps. Overall, more than 10^{13} data points are produced to test one ADAS setting, which represents several gigabytes per simulation plan (up to 1 terabyte). Because the simulation duration depends on the driving scenario, the simulated time series lengths are unequal. However, because the variables describe the simulation at the same sampling rate, the time series lengths are equal for a given simulation.

The final dataset can be represented as a matrix, where each cell is a time series with simulation-dependent length. An instance of such matrix is displayed in Fig. 0.3, with a fictive simulation result.

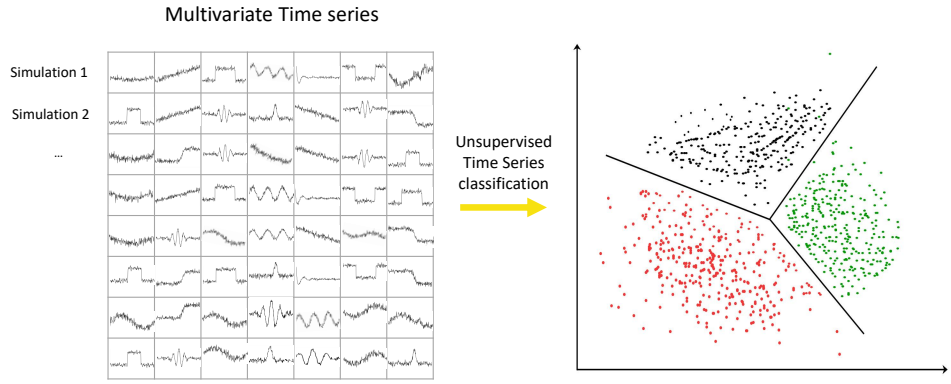


Fig. 0.4.: Multivariate Time Series Clustering Illustration. Left Matrix figure from [SAJ18].

Even though some simulated driving behaviors are predictable (e.g., the predicted valid behavior), most are discovered after simulation, which prevents from using labels to supervise the analysis. In this high-dimensional, multivariate and unsupervised context, extracting meaningful information is a tedious and time-consuming task for the expert, and requires specific exploration tools. The action of automatically regrouping time series into homogeneous clusters, without supervision, is called time series or clustering. An instance of multivariate time series clustering method applied to a driving simulation dataset is displayed in Fig. 0.4, where the scatter plot on the right shows the multivariate observations regrouped into clusters and represented in a simpler 2D space.

Contributions

The clustering of time series is a complex task that depends heavily on the hypotheses that the user forms on the considered dataset. Without label, it is never possible to know if the clustering result is correct, because there is no ground truth to compare the results to.

On the contrary, when an user applies a clustering method on a dataset, it is the understanding of the underlying assumptions that allows to reason and draw conclusions from the results.

The first contribution of this thesis is based on a scenario-construction hypothesis. Based on the fact that simulations are generated on the basis of driving scenarios (c.f., Fig. 0.2), we developed a method based on the segmentation of time series in

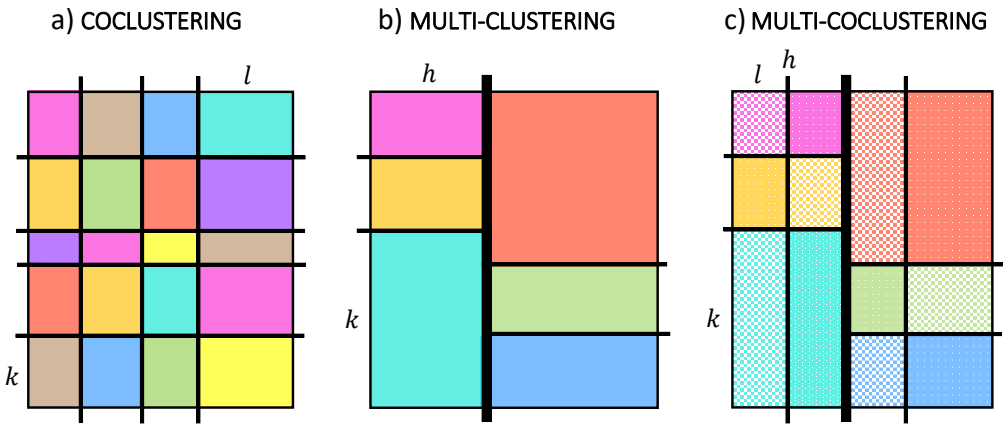


Fig. 0.5.: Structures of the proposed models: a) the coclustering infers one column-partition and one row-partition; b) the Multi-Clustering infers one column-partition and one row-partition per column-cluster; c) the multi-coclustering combines the Multi-Clustering with a coclustering layer.

chains of driving phases. With this method, the time series are recoded as chains of 'regimes', where regimes are driving patterns found in the entire datasets and aggregated in a dictionary. The final dataset is obtained by clustering the obtained character sequences on the basis of a string-specific distance. As a probabilistic method, this approach can be used for outlier detection based on probabilistic confidence interval and missing data inference.

This method is designed for univariate time series clustering. However, in most use cases, the ADAS effects are observed by several variables simultaneously (e.g. speed, braking intensity, steering wheel angle, ...), and the field expert is therefore interested in observing the distributions of several variables together. The following contributions attempt to deal with this issue and form a separate set of methods dedicated to *multivariate* time series clustering. These methods share the model-based *coclustering* as common root. The coclustering consists in inferring two partitions simultaneously: one simulation partition (called *row-partition* and one variable partition (called *column-partition*), as shown on Fig. 0.5-a.

We extend this approach with three new models:

- A Bayesian non-parametric time series coclustering model that natively integrates a model selection step.
- A parametric Time Series *Multi-Clustering* model that infers one column-partition and one specific row-partition per column cluster (c.f., Fig. 0.5-b).

Method	Chapter	Link
SDLHC	Chapter 2	https://tinyurl.com/sdlhc
FunCLBM	Chapter 3	https://tinyurl.com/FunCLBM
FunNPLBM	Chapter 4	https://tinyurl.com/funNPLBM
FunMCC	Chapter 5	https://tinyurl.com/FunMCC

Tab. 0.1.: Links to the Github repositories associated with the contributions

- A non-parametric multi-coclustering. This approach combines the multi-clustering with coclustering structures, to construct several views of the same multivariate time series dataset while reducing the number of parameters and enhancing interpretation (c.f., Fig. 0.5-c).

The source code associated with these distributions, mainly Scala code, with visualizations in R and Python, is also made available, with the public datasets used for test. The github repositories links are summarized in Tab. 0.1. The industrial use cases datasets cannot, however, be made available.

Summary of remaining chapters

The rest of the manuscript is organized as follows:

Chapter 1: Related work

The first chapter focuses on the existing works related to time series unsupervised classification, univariate or multivariate. This chapter also introduces the model-based clustering, coclustering, multi-clustering, and the Bayesian non-parametric modeling, that will intensively be used in the following chapters.

Chapter 2: Time Series Clustering with Model-Based Dictionary Representation.

This chapter details the first contribution of this thesis, an univariate time series clustering method consisting of three steps: a) a model-based piecewise polynomial regression used for time series segmentation; b) a model-based dictionary construction that regroups similar driving patterns; c) a character sequences hierarchical clustering.

Chapter 3: Functional Conditional Latent Block Model

This chapter introduces the first contribution dedicated to multivariate time series with multi-block clustering. This method consists in a parametric model-based Multi-Clustering model that extends an existing time series coclustering model by allowing to infer several row-partitions instead of one.

Chapter 4: Bayesian Non-Parametric Coclustering

This chapter presents a Bayesian Non-Parametric multivariate time series coclustering. Based on an existing coclustering model, this method adds prior distributions on the column and row proportions and on block components parameters. With this extension, this new model natively performs model selection.

Chapter 5: Multivariate Time Series Multi-Coclustering

Built on the methods detailed in Chapter 4 and Chapter 3, the multi-coclustering takes the best of coclustering and multi-clustering. This method consists in hierarchically nested Dirichlet Process Mixtures that infer a multi-clustering view of a dataset and then a coclustering structure in each view.

Chapter 6: Conclusion And Perspectives

This chapter summarizes possible extensions for the proposed methods. These extensions focus on multivariate extensions for the dictionary-based univariate clustering method, and different representations in the block-clustering approach.

Publications

These contributions were the subject of several publications, chronologically represented in Fig. 0.6, and listed below:

- Goffinet, É., Lebbah, M., Azzag, H., Giraldi, L., & Renault, S. A. S. (2020). Clustering de séries temporelles par construction de dictionnaire. In EGC (pp. 181-192).
- Goffinet, E., Lebbah, M., Azzag, H., & Giraldi, L. (2020). Autonomous Driving Validation with Model-Based Dictionary Clustering. In ECML/PKDD (4) (pp. 323-338).
- Goffinet, E., Lebbah, M., Azzag, H., Giraldi, L. & Coutant, A. Validation de Systèmes de Conduite Autonome par Co-clustering de Séries Temporelles (2020). Workshop Nouvelles méthodes pour l'analyse descriptive et prédictive de données massives et structurées. Société Francophone de Classification (SFC)
- Goffinet, E., Coutant, A., Lebbah, M., Azzag, H., & Giraldi, L. (2020). Conditional Latent Block Model: a Multivariate Time Series Clustering Approach for Autonomous Driving Validation. arXiv preprint arXiv:2008.00946.

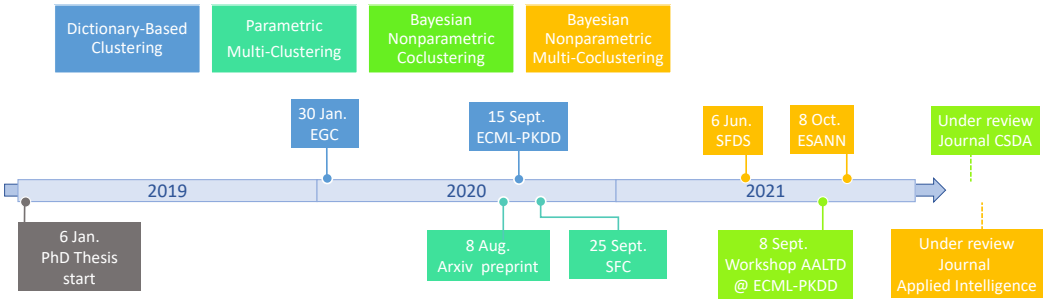


Fig. 0.6.: Publication chronology.

- Goffinet, É., Lebbah, M., Azzag, H., Giraldi, L., & Coutant. A (2020). Multiple Co-clustering de séries temporelles. Application à la validation de systèmes d'aide à la conduite. In 52emes Journées de Statistiques. Société Française de Statistique.
- Goffinet, É., Lebbah, M., Azzag, H., Giraldi, L., & Coutant. A (2021). Non-Parametric Multivariate Time Series Co-clustering Model Applied to Driving-Assistance Systems Validation. In International Workshop on Advanced Analysis and Learning on Temporal Data @ ECML/PKDD.
- Goffinet, E., Lebbah, M., Azzag, H., Giraldi, L. & Coutant, A. (2021). Multivariate Time Series Multi-Coclustering. Application to Advanced Driving Assistance System Validation. In ESANN 2021-29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.

Some publications are, at the time of writing, still under review:

- Goffinet, É., Lebbah, M., Azzag, H., Giraldi, L., & Coutant. A. (2021). Multivariate Time Series Multi-Coclustering. Application to Advanced Driving Assistance System Validation. Submitted to Applied Intelligence: Special Issue on Multi-view Learning. Springer.
- Goffinet, E., Lebbah, M., Azzag, H., Giraldi, L. & Coutant. A. (2021). Functional Non-Parametric Latent Block Model: a Multivariate Time Series Clustering Approach for Autonomous Driving Validation. Submitted to Computational Statistics & Data Analysis.

Introduction

> *De plus en plus, les technologies connectées et la conduite autonome permettront aux usagers de faire un choix entre conduire et être conduits. Ce constat ouvre la voie à de nouveaux scénarios de mobilité. Voici notre prochaine révolution.*
>
> **— Guillaume Eurin**
> Directeur du Développement des ADAS
> et du Véhicule Autonome .
> *Groupe Renault* Rapport Annuel (2019-2020)

Cette thèse a été organisée en convention entre le constructeur automobile *Groupe Renault*[4] et le laboratoire d'informatique *LIPN*[5] (Laboratoire d'Informatique de Paris Nord, UMR CNRS 7030) de l'*Université Sorbonne Paris Nord*[6]. Ce chapitre présente le contexte industriel qui a motivé cette convention de recherche: la validation de systèmes d'aide à la conduite (ADAS) et ses contraintes opérationnelles. Cette description permet de détailler le rôle des ADAS dans le fonctionnement d'un véhicule, l'interêt de leur développement pour le constructeur. Elle présente également la nécessité de l'utilisation de la simulation massive pour la validation et la nature du travail d'analyse de données associée aux jeux de données simulées. Enfin, cette partie présente également les contributions, en réponses aux contraintes opérationnelles et les publications associées, avant de conclure par le plan global du manuscript.

Systèmes d'aide à la conduite (ADAS)

Les systèmes d'aide à la conduite (ADAS) sont des systèmes électroniques embarqués qui fournissent une assistance pour certaines tâches associées à la conduite. Depuis les premiers systèmes d'*Anti-Blocage des Roues* (ABS) et de *Vitesse de Croisière*

[4]https://www.renaultgroup.com/
[5]https://lipn.univ-paris13.fr/
[6]https://www.univ-paris13.fr/

développés dans les années 1950, jusqu'à la très récente fonction de *Pilote Autoroutier Automatique*, le développement de ces systèmes a accompagné le marché automobile durant toute son histoire. Toute voiture récente en est aujourd'hui équipée.

Certains de ces systèmes sont passifs et ne font que produire une information ou un avertissement (par exemple les systèmes de navigation ou le rappel de non-bouclage des ceintures de sécurité, . . .). D'autres fournissent une aide active et sont capables d'agir sur les commandes de conduite (par exemple, la vitesse de croisière adaptative, le freinage d'urgence, . . .).

Ces ADAS peuvent être classés en fonction de leur niveau d'automatisation de conduite, sur une échelle allant de 0 à 5 et développée par la Society of Automotive Engineers (SAE). Cette échelle, representée sur la Fig. .7, peut également être vue comme une feuille de route menant au développement de la voiture autonome, et dont les ADAS constitueraient les jalons. La liste suivante présente trois exemples d'ADAS existants ou en développement:

- Le Freinage d'Urgence Automatique (AEB) - Niveau 0. Durant la conduite, ce freinage se déclenche lorsque la voiture équippée (appelée *Ego*) détecte une cible (piéton, vélo, voiture, camion, . . .) se rapprochant à grande vitesse.

- L'Aide au Maintien dans la Voie (LKA) - Niveau 1. Durant la conduite, ce système détecte une dérive non intentionnelle d'Ego en direction d'une autre voie ou du bord de la route et corrige sa trajectoire.

- Valet Parking - Niveau 4 (En développement). Dans une aire de stationnement, ce système dirige le véhicle sans intervention extérieure, cherche et trouve un espace de stationnement disponible et réalise la manoeuvre de stationnement.

Pour le conducteur, ces ADAS sont des gages de sécurité, de confort et d'efficacité. D'un point de vue plus global, les ADAS sont pensés comme une forme d'aide à la mobilité, d'assistance aux personnes agées, et une solution pour réduire le traffic automobile [SS15]. D'un point de vue commercial, ils constituent des arguments de vente décisifs qui permettent aux constructeurs de prouver leur savoir-faire et de se démarquer dans un marché de l'automobile extrêmement compétitif. Dans un récent rapport de recherche [Res21], le cabinet d'étude marketing *Precedence Research* estimait la valeur du marché mondial de l'ADAS à 32.3 Milliards de dollars en 2019, et prévoyait une hausse jusqu'à 142 Milliards de dollars en 2027.

Dans la perspective d'équipper des millions de voitures à travers le monde, la sécurité est la première préoccupation lors du développement d'un ADAS. Pour s'en assurer,

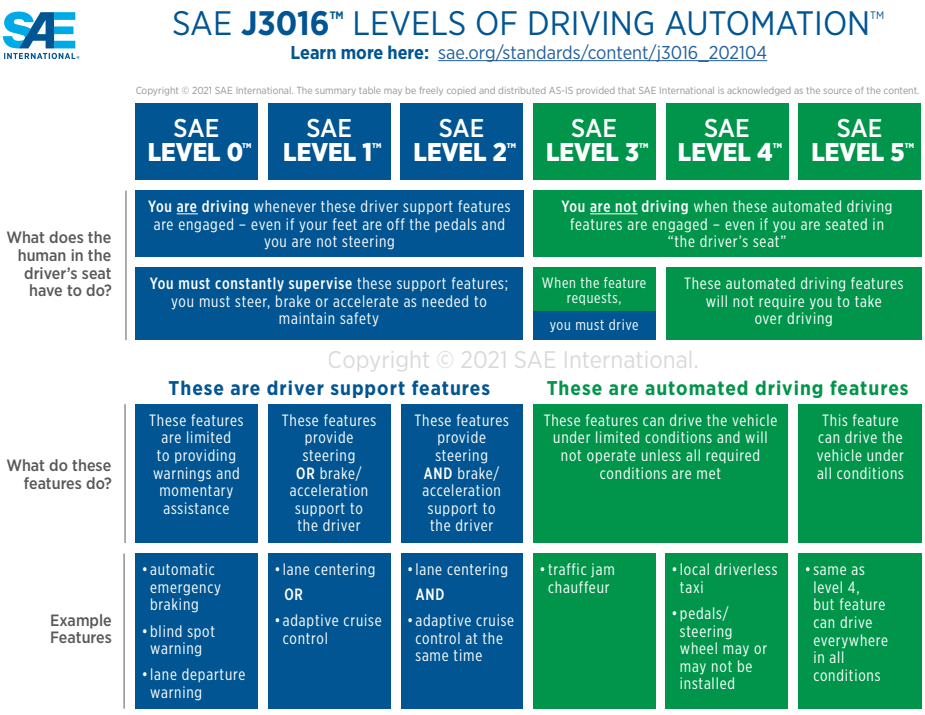


Fig. .7.: Système de classification des ADAS (source: SAE).

il est donc nécessaire de mettre en place un système de validation rigoureux qui vérifiera les performances de l'ADAS dans tous les contextes de conduite.

Validation

L'objectif de l'étape de validation est de confirmer que les réactions d'un ADAS sont conformes au cahier des charges fonctionnel, rédigé par le Groupe Renault, en accord avec les recommandations des agences gouvernementales de sécurité (par exemple *Euro NCAP* en Europe)

Dans ce contexte, pour la validation d'un ADAS hautement autonome (Niveau 4), le Groupe Renault requiert un niveau minimum de fiabilité de 10^{-8} décès par heure, ce qui correspond à 10 fois moins que la moyenne du nombre d'accidents sur autoroute en Europe, et 100 fois moins que la moyenne d'accident sur tous types de route. Vérifier un tel niveau de fiabilité demanderait de réaliser au moins 2×10^{10} kilomètres de conduite [Wau18], soit plus de 100 années sur la route.

Ce niveau de fiabilité doit être atteint pour chaque véhicule de la gamme, pour chaque ADAS et dans toutes les conditions de conduite. De plus, les développeurs doivent également s'assurer que les ADAS sont inter-opérables, et que leur combinaison ne provoque pas d'effets de bord (par exemple: l'évitement d'urgence et le freinage d'urgence). L'accumulation de tous ces facteurs de complexité rend impossible la validation basée uniquement sur des tests physiques sur piste ou sur route.

Pour résoudre ce problème et parvenir à tenir ces engagements de fiabilité, Le Groupe Renault a fait le choix d'investir dans la technologie de simulation massive. Ces systèmes de simulation reproduisent les conditions de conduite en se basant sur un scénario, la physique du véhicule, le comportement du conducteur, et les interactions avec un environnement lui-même configurable. Le procédé de simulation consiste en un processus distribué qui génére des scènes physiques en parallèle. La qualité de la simulation, c'est-à-dire sa capacité à reproduire fidèlement les conditions de conduite réelles, est méthodiquement monitorée afin de s'assurer de la pertinence du procédé de simulation. Le système de simulation est representé sur la Fig. .8.

La simulation massive produit de grandes quantités d'information provenant de l'observation de variables simulées, et aggrégées en un unique jeu de données de séries temporelles multivariées. Le rôle de l'expert métier est alors d'analyser ce jeu

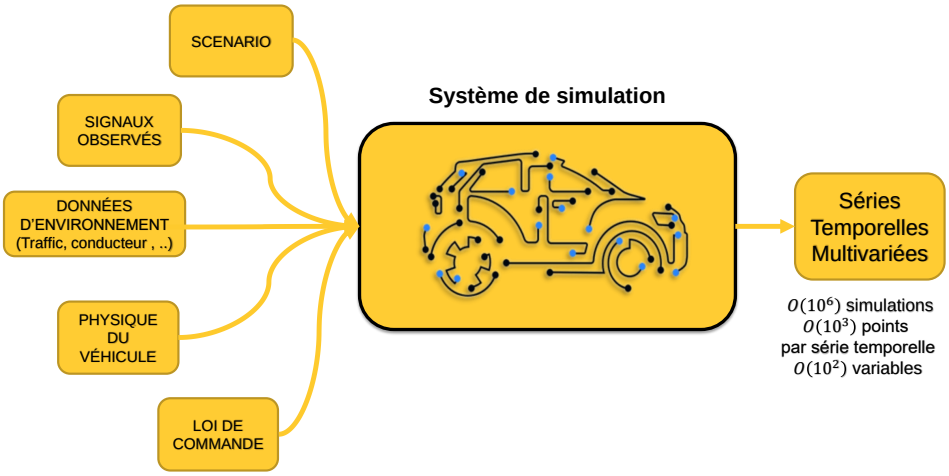


Fig. .8.: Représentation schématique du système de simulation.

de données pour discriminer des comportements de conduite, détecter des situations aberrantes, et de comparer les performances de l'ADAS au cahier des charges.

Les dimensions et la complexité des données sont choisies par l'utilisateur, et peuvent être considérables. Pour un cas d'usage donné, il est possible de voir des jeux de données de plus 10^6 simulations, décrites par 10^3 variables, chacune en constituée de 10^4 points temporels. Au total, plus de 10^{13} observations peuvent donc être produits pour tester le paramétrage d'un ADAS (ce qui représente jusqu'à 1 téraoctet). Chaque simulation a une durée spécifique, qui dépend du scénario et des interactions complexes entre la voiture et la situation de conduite. Par conséquent les séries temporelles simulées sont également de tailles différentes.

Le jeu de données final peut être représenté sous la forme d'une matrice, dont chaque cellule contient une série temporelle de taille spécifique. Un exemple fictif de matrice simulée est représentée sur la Fig. .9.

De plus, si certains résultats de simulation peuvent être prédits (en particulier, le comportement attendu), la plupart sont découverts après la simulation. Par conséquent, il n'est pas possible d'établir à l'avance la liste exhaustive de labels qui permettrait de réaliser une analyse supervisée. Dans ce contexte en grandes dimensions, multivarié et non-supervisé, extraire de l'information pertinente serait à la fois chronophage et laborieux pour l'expert, sans l'aide d'outils spécifiques d'analyse de données.

Les méthodes qui font une classification automatique de séries temporelles dans des groupes homogènes, sans supervision, sont appelées des méthodes de classification non-supervisée de séries temporelles, ou *Clustering* de séries temporelles. La Fig. .10

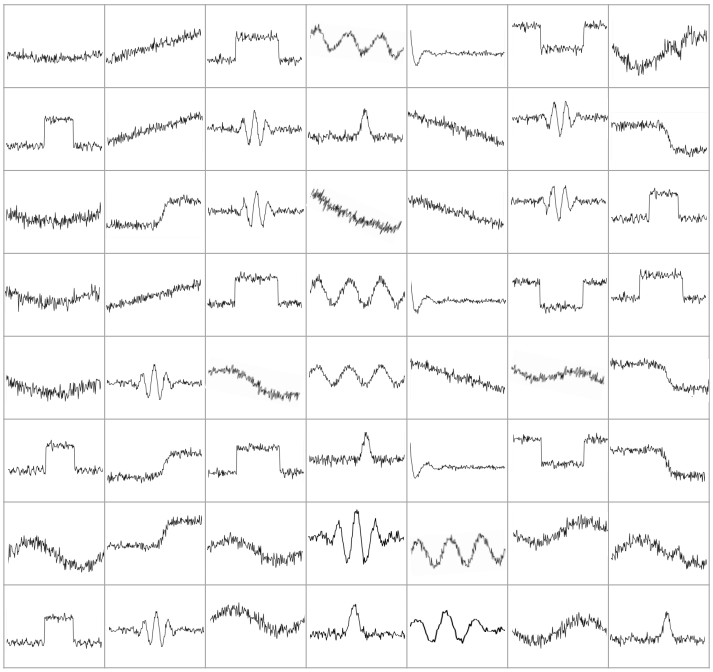


Fig. .9.: Illustration du contenu d'un jeu de données simulées. Celui-ci est représenté par une matrice, contenant une série temporelles dans chaque cellule.

représente l'exemple d'une telle méthode appliquée à jeu de données de séries temporelles multivariées. Sur cet exemple, le graphique de droite montre les séparations entre différents groupes de simulations, après une projection dans un espace 2D.

Contributions

La classification non supervisée de séries temporelles est une tâche complexe, qui dépend fortement des hypothèses formulées par l'analyste sur le jeu de données étudié. En l'absence de labels et de supervision, il n'est jamais possible de savoir si le résultat du clustering est correct, puisque, par construction, il n'y a aucune façon de savoir ce que "correct" signifie. Le problème est à considérer dans l'autre sens: lorsque l'analyste applique une méthode de clustering sur un jeu de données, c'est la compréhension de la méthode utilisée et des hypothèses sous-jacentes qui permet d'interpréter les résultats et d'en tirer des conclusions.

La première contribution de cette thèse se base sur une hypothèse de construction de séries par scénario. Observant que les simulations sont générées sur la base de

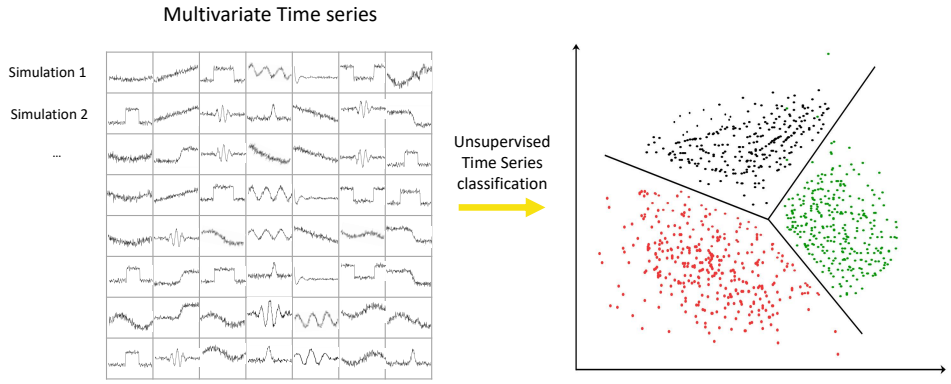


Fig. .10.: Illustration d'une méthode de Clustering de séries temporelles multivariées. Image de gauche: [SAJ18].

scénario de conduite (c.f., Fig. .8), nous avons développé une méthode basée sur la détection de différentes phases de conduite. Durant l'analyse, les séries temporelles sont recodées sous formes de chaînes de régimes de conduite, qui sont des schémas de conduite issues de l'observation du jeu de données globales et aggrégées en dictionnaire. Pour finir, ces séquences de régimes sont partitionnées sur la base d'une distance entre chaînes de caractères. Cette méthode de clustering probabiliste permet également la détection d'anomalies basée sur des intervalles de confiance probabilistes, et l'inférence de valeurs manquantes.

Cependant, par construction, cette méthode ne s'applique qu'à des séries temporelles univariées, ce qui correspond, dans notre cas, à l'analyse d'une seule variable. Cependant, dans la plupart des cas, le comportement d'un ADAS influe sur plusieurs dimensions, observées par plusieurs variables. Il est donc logique de vouloir analyser plusieurs variables simultanément. Les contributions suivantes répondent à cette problématique et forment un groupe cohérent de méthodes dédiées au clustering de séries temporelles multivariées. Elles partagent la même vision multi-blocs et la même racine commune: le *Coclustering*. Le Coclustering infère simultanément une partition de simulations (appelées partition ligne) et une partition de variables (appelées partition colonne), comme illustré sur la Fig. .11-a.

Cette thèse propose des extensions du Coclustering, avec trois méthode probabilistes :

- Une méthode de Coclustering Bayésien Non-Paramétrique, qui intègre nativement une étape de sélection de modèle.

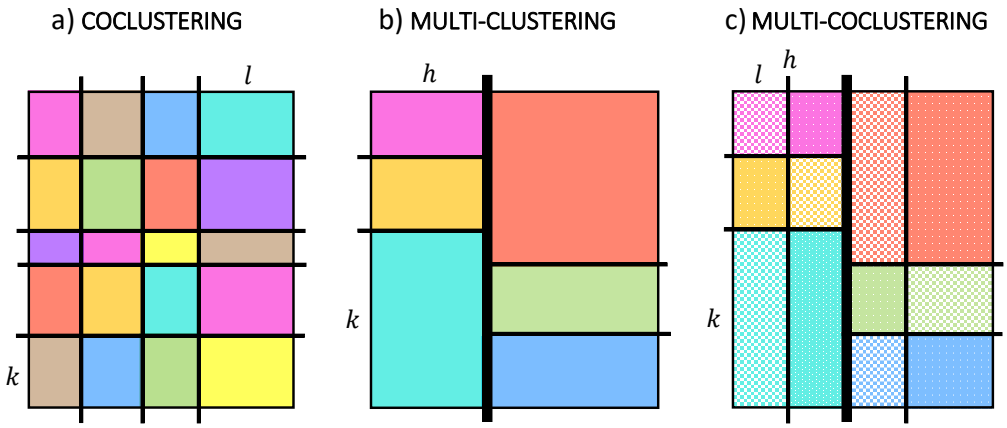


Fig. .11.: Structure des modèles proposés: a) Le Coclustering infère une partition colonne et une partition ligne ; b) le Multi-Clustering infère une une partition colonne et une partition ligne dans chaque groupe de variable ; c) le Multi-Coclustering combine les deux aspects et ajoute une couche de structures de Coclustering dans le Multi-Clustering.

Méthode	Chapitre	Lien
SDLHC	Chapitre 2	https://tinyurl.com/sdlhc
FunCLBM	Chapitre 3	https://tinyurl.com/FunCLBM
FunNPLBM	Chapitre 4	https://tinyurl.com/funNPLBM
FunMCC	Chapitre 5	https://tinyurl.com/FunMCC

Tab. .2.: Liens vers les dépôts Github associés aux contributions

- Une méthode de Multi-Clustering paramétrique qui permet d'inférer une partition colonne et une partition ligne pour chaque cluster colonne. (c.f., Fig. .11-b).

- Une méthode de Multi-Coclustering non-paramétrique. Cette méthode combine le Multi-Clustering en y intégrant des structures de Coclustering (c.f., Fig. .11-c). Cette approche crée plusieurs vues d'un même jeu de données tout en réduisant le nombre de paramètres, le tout dans un cadre Bayésien Non-Paramétrique qui permet de résoudre la problématique de sélection de modèle.

Le code source associé à ces modèles, majoritairement écrit en Scala, avec des visualisations en R et Python, est également mis à disposition, ainsi que les jeux de données publics de test. Le lien vers les dépôts Github sont rassemblés dans le tableau .2. Par souci de confidentialité, les données des cas d'usage Groupe Renault ne sont pas mises à disposition.

Plan de la thèse

La suite du manuscript se compose des articles suivants :

Chapter 1: Related work

Le premier chapitre présente les travaux existants sur le sujet de la classification de séries temporelles, univariées ou multivariées. Ce chapitre introduit en particulier les notions de clustering et coclustering probabilistes, ainsi que la modélisation Bayésienne Non Paramétrique, notions qui seront ré-utilisées régulièrement dans les chapitres suivants.

Chapter 2: Time Series Clustering with Model-Based Dictionary Representation.

Ce chapitre détaille la première contribution de cette thèse, une méthode de clustering de séries temporelles univariées construite en trois étapes : a) segmentation individuelle par régression polynomiale par morceaux; b) construction d'un dictionnaire de phases de conduites par modèle de mélange; c) clustering de chaînes de caractères.

Chapter 3: Multivariate Time Series And Functional Conditional Latent Block Model

Ce chapitre présente la première méthode de clustering de séries temporelles multivariées, avec une méthode de Multi-Clustering par blocs. Cette approche paramétrique probabiliste de Multi-Clustering étend le Coclustering de séries temporelles en permettant la création de plusieurs vues d'un même jeu de données multivariées.

Chapter 4: Bayesian Nonparametric Coclustering

Cette partie présente une seconde extension du modèle de coclustering de séries temporelles multivariées. Cette extension ajoute un cadre Bayésien Non Paramétrique, sous la forme de prior Bayésien Non-Paramétrique sur les colonnes et les lignes. Cette extension permet d'intégrer nativement une sélection de modèle durant l'inférence.

Chapter 5: Multivariate Time Series Multi-Coclustering

Basée sur les méthodes détaillées dans les chapitres 3 et 4, le Multi-Coclustering combine Multi-Clustering et Coclustering. Cette approche consiste en trois modèles de mélange Bayésiens Non-Paramétriques hiérarchiquement imbriqués et permet l'inférence d'une structure de Multi-Clustering global, puis d'un Coclustering à l'intérieur de chaque vue.

Chapter 6: Conclusion And Perspectives

Ce dernier chapitre présente de possibles extensions des méthodes proposées. Ces propositions concernent, entre autres, l'extension multivariée de la méthode de clustering par dictionnaire, et la recherche de méthode de représentation alternative pour la méthode de clustering multi-blocs.

Publications

Ces contributions ont été l'objet de plusieurs publications au cours de la thèse, représentées sur la Fig. .12 et listées ci-dessous:

- Goffinet, É., Lebbah, M., Azzag, H., Giraldi, L., & Renault, S. A. S. (2020). Clustering de séries temporelles par construction de dictionnaire. In EGC (pp. 181-192).
- Goffinet, E., Lebbah, M., Azzag, H., & Giraldi, L. (2020). Autonomous Driving Validation with Model-Based Dictionary Clustering. In ECML/PKDD.
- Goffinet, E., Lebbah, M., Azzag, H., Giraldi, L. & Coutant, A. Validation de Systèmes de Conduite Autonome par Co-clustering de Séries Temporelles (2020). Workshop Nouvelles méthodes pour l'analyse descriptive et prédictive de données massives et structurées. Société Francophone de Classification (SFC)
- Goffinet, E., Coutant, A., Lebbah, M., Azzag, H., & Giraldi, L. (2020). Conditional Latent Block Model: a Multivariate Time Series Clustering Approach for Autonomous Driving Validation. arXiv preprint arXiv:2008.00946.
- Goffinet, É., Lebbah, M., Azzag, H., Giraldi, L., & Coutant. A (2020). Multiple Co-clustering de séries temporelles. Application à la validation de systèmes d'aide à la conduite. In 52emes Journées de Statistiques. Société Française de Statistique.
- Goffinet, É., Lebbah, M., Azzag, H., Giraldi, L., & Coutant. A (2021). In International Workshop on Advanced Analysis and Learning on Temporal Data @ ECML-PKDD .
- Goffinet, E., Lebbah, M., Azzag, H., & Giraldi, L. (2021). Multivariate Time Series Multi-Coclustering. Application to Advanced Driving Assistance System Validation. In ESANN 2021-29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. ESANN.

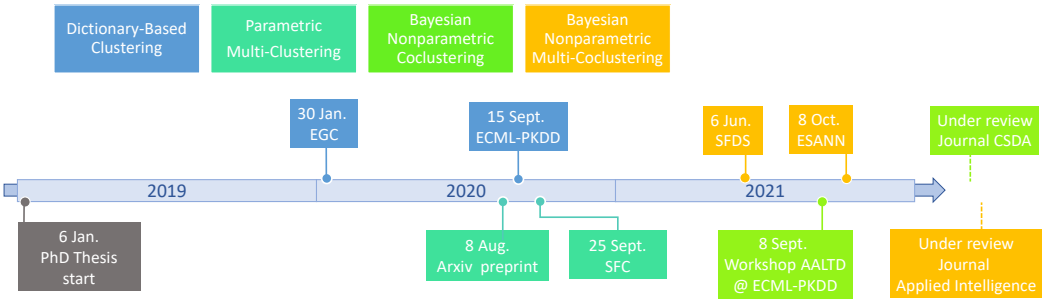


Fig. .12.: Chronologie des publications.

Plusieurs publications sont, au moment de la rédaction de cette thèse, toujours en cours de révision:

- Goffinet, É., Lebbah, M., Azzag, H., Giraldi, L., & Coutant. A. (2021). Multivariate Time Series Multi-Coclustering. Application to Advanced Driving Assistance System Validation. Soumis à Applied Intelligence: Special Issue on Multi-view Learning. Springer .
- Goffinet, E., Lebbah, M., Azzag, H., Giraldi, L. & Coutant. A. (2021). Functional Non-Parametric Latent Block Model: a Multivariate Time Series Clustering Approach for Autonomous Driving Validation. Soumis à Computational Statistics & Data Analysis.

Notations

Tab. .3.: Notations summary

Dataset			
$\mathbf{s} = (s_i)_n$ $= (s_i(t_j)_{T_i})_n$	Univariate time series dataset	$S = (s_{i,j})_{n \times p}$	Multivariate time series dataset
n	Observation number	p	Variable number
$(T_i)_n$	Time series length		
$\mathbf{x} \in \mathbb{R}^{n \times d}$	Dataset after transformation of $\mathbf{s}$	$X \in \mathbb{R}^{n \times p \times d}$	Dataset after transformation of S
d	Representation space dimension	$X_{-i,.}/X_{.,-j}$	X without row i/column j
$\mathrm{d}(.,.)$	Dissimilarity measure	$\mathbf{x}_{i,.}/\mathbf{x}_{.,j}$	row i / column j of X
Mixture Model and Dirichlet Process Mixture Model			
$\mathbf{z}$	Observation memberships	K	Component number
F	Distribution family	$(\theta_k)_K$	Components parameter
$\pi = (\pi_k)_K$	Mixture proportions	(μ, Σ)	Multivariate Gaussian parameters
M	Iteration number	Θ	Parameter set
$(n_k)_K$	Cluster Size		
α	Concentration parameter	K_{obs}	Observed component number
G_0	Base Distribution	$\mu_0, \kappa_0, \Psi_0, \nu_0$	NIW prior parameters
χ	Hyper-parameter set	$\mu_k, \kappa_k, \Psi_k, \nu_k$	Cluster k posterior distribution parameters
Dictionary-based time series clustering			
s	Time series	$(z_t)_T$	Segment membership
$\phi = (\phi_q)_Q$	Basis of polynomial functions	$\beta = (\beta_q)_Q$	Polynomial regression coefficients
$R = \{r_u\}_U$	Dictionary of regimes (size U)	$(w_{k,s})_K$	Logistic process parameters in component k
σ^2	covariance parameter		

Notations summary (continued)

Multi-Block Clustering			
$f_{i,j}$	Frequencies basis expression of $s_{i,j}$	$\widehat{f}$	Interpolated frequency basis
$\mathbf{w} = (w_j)_p$	Coclustering variable partition	$\rho = (\rho_l)_L$	Coclustering column proportions
L	Number of column cluster	β	Column-cluster concentration parameter
$(p_l)_L$ or $(p_h)_H$	Column-clusters size	$(n_{k,l})_{K\times L}$	Block-cluster size
$\theta_{k,l}$	Block distribution parameters	$\mu_{k,l}, \kappa_{k,l}, \Psi_{k,l}, \nu_{k,l}$	Block cluster (k,l) posterior distribution parameters
H	Number of Multiple-Clustering clusters	$(K_h)_H, (L_h)_H$	Row and column-clusters numbers
n_k^h, p_l^h	Row and column clusters sizes	$n_{k,l}^h$	Block-clusters size
$Z = (z_i^h)_{n\times H}$	Multiple row partitions	$(\pi^h)_H$	Multiple row partitions proportions
$(\mathbf{w}^h)_H$	Multiple coclustering column partitions	$(\rho^h)_H$	Multiple coclustering column proportions
$\mathbf{v} = (v_j)_p$	Multi-Clustering variable cluster membership	$\eta = (\eta_h)_H$	Multi-Clustering component proportions
$\theta_{k,l}^h$	Multiple coclustering block distribution parameters	$\mu_{k,l}^h, \kappa_{k,l}^h, \Psi_{k,l}^h, \nu_{k,l}^h$	Block cluster (k,l) posterior distribution parameters
$\gamma, (\alpha_h)_H, (\beta_h)_H$	Multiple coclustering concentration parameters	a_γ, b_γ a_β, b_β a_α, b_α	Beta hyper-prior parameters

1 Related Work

In the previous chapter we described the industrial context that motivated this thesis contributions: during the ADAS validation step, the field expert deals with large quantities of driving simulations, with unequal lengths and without labels. As a consequence, he must rely on unsupervised tools to explore the datasets contents. The purpose of this exploration is to recognize driving scenarios and identify them as correct or incorrect.

1.1 Time Series Clustering

This chapter describes the existing works on the topic of time series clustering, starting with the basic definition of Clustering in Sect. 1.1.2. The subsequent Sect. 1.1.3 and 1.1.4 respectively introduce a review of time series dissimilarities and transformations. Section 1.1.5 describes existing clustering algorithms, with an emphasis on the methods that are used and extended in the present thesis.

1.1.1 Time Series Analysis

Time series is a specific kind of data generated from successive observations of a system state. This data type can be observed in every time-depending field. In Finance, they result from the observations of sales, stock prices or exchange rates evolution. They are also used extensively in Health use cases, for the description of a patient state through medical sensors. In Industry applications, they can monitor machine activity, production rates, and optimize predictive maintenance. Logistic, Demography, Retail, Climate, Astronomy,... The application possibilities are numerous.

In correlation with the use case number, processing power and storage capacities, the time series analysis topic has seen a sharp increase of interest in the recent decades. Time series data analysis can be applied to several kinds of usage, including missing data inference (which includes forecasting when these data are posterior to the observed time series), anomaly detection, or classification.

In our applications, we are particularly interested in two of these usages: the detection of anomalous driving behaviors, and the unsupervised classification of driving patterns.

1.1.2 Unsupervised Classification

> *Knowledge is a process of piling up facts; wisdom lies in their simplification.*
>
> **— Martin H. Fischer**

An informal definition of unsupervised classification, or *clustering*, is: "(...) to discover groups of similar examples within the data (...)" [Bis06]. This partitioning is performed without the help of labels to supervise the estimation. The clustering transforms a potentially complex dataset (in our case, high-dimensional multivariate time series) into a simplified information, expressed as a categorical vector of memberships.

As opposed to supervised classification (simply called *classification*), that seeks to correctly infer an observation class based on a ground-truth, clustering creates a partition without supervision, i.e., only based on the dataset content. It is an exploration tool that helps the user understand the content of a dataset, and its structure.

The result partition, represented by a set of labels, depends on the choice of the clustering method, but also on the methods parameters (called *hyper-parameters*). For instances, some methods requires to know the cluster number *a priori*, to specify a time series dissimilarity measure, or an initial inference state.

The choice of clustering method and the hyper-parameters specification strategy have a considerable impact on the clustering result. For instance, the user can choose to discriminate time series based on trends, shape, or simply length or sampling frequency. These features have nothing in common (two time series can have similar trends but completely different length). As a consequence, depending on user hypothesis, the results can change completely. It is up to the user to choose among all the possible representations, features, dissimilarities and every possible hyper-parameter settings, according to its need and prior knowledge. It is therefore crucial that the user fully understands how the clustering process works, and the role and impact of each hyper-parameter.

In the following we detail some of the existing approaches and strategies, with an emphasis on the methods used in the present thesis.

1.1.3 Time series similarity measure

The broad definition of clustering ("grouping similar elements together") outlines the importance of the similarity definition in a clustering process. One way of clustering time series is to use a specific time series distance and to perform a clustering in the original time series space. This approach is also called raw-data-based, or shape-based approach (this denomination is, for instance, given in [ZMK12] or in [ASW15]).

In the univariate time series clustering context, let denote $\mathbf{s} = (s_i)_n$ a set of time series observations, such that each element $s_i = (s_i(t))_{T_i}$ is a time series indexed by T_i. In the following we call time series dissimilarity measure d a function of two time series with the following properties, for two time series (s_1, s_2):

$$\mathrm{d}(s_1, s_2) \geq 0,\ \mathrm{d}(s_1, s_2) = \mathrm{d}(s_2, s_1),\ \mathrm{d}(s_1, s_1) = 0.$$

The definition of distance adds the triangle inequality property, i.e., for all (s_1, s_2, s_3), $\mathrm{d}(s_1, s_2) \leq \mathrm{d}(s_1, s_3) + \mathrm{d}(s_3, s_2)$.

Defining a time series dissimilarity measure is not straightforward, because there are many different ways to discriminate time series. As a consequence, many distances or similarity measures have been proposed in the literature [IK13].

The most popular time series distance is theℓ^2-norm (also called Euclidean distance) that compares time series point-to-point. This distance is defined by:

$$\mathrm{d}(s_1, s_2) = \left(\sum_T (s_1(t) - s_2(t))^2 \right)^{1/2}.$$

However, this distance is strongly impacted by time axis distortion and requires aligned time series with equal length.

The Pearson's correlation distance is used for comparing time series based on shape similarity. However, as stated by [IK13], this distance "is sensitive towards outliers and generally not suitable for other probability distributions except for the Gaussian one", which makes it impractical in many situations.

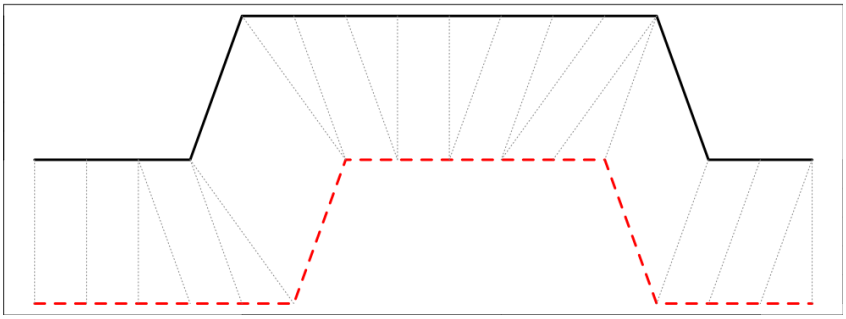


Fig. 1.1.: DTW measure alignment between two rectangular time series (generated with [Gio09]).

The Mahalanobis distance is based on the Euclidean distance and integrates the correlation information to form a distribution-based distance [PL12].

The Dynamic Time Warping (DTW) [Sak71] measure is another popular time series dissimilarity, especially suited to deal with local or uniform temporal scaling (a.k.a. warping). Inspired by the edit distance (used in the context of string comparison), DTW is a measure of the effort required to align two time series. An illustration of this alignment is displayed on Fig. 1.1, with the example of two rectangular shapes.

The distance computation was quite resource-intensive originally ($O(n^2)$ complexity), but several works have been developed over the year [SC07; KR05; HW21] that reduced this cost (to $O(n)$ in some cases [RK05]). The exhaustive list of time series similarity measures (Longest Common Subsequence [CS75], Compression-based similarity measure[Keo+07], ...) cannot be exhaustively presented here, but can be found in the following extensive reviews [ZHT06; IK13; ASW15; AML19].

In addition to the high number of existing dissimilarity measures, it is also possible to use a weighted or nested combination of them [BNS11], which transforms the similarity choice from a discrete into a continuous optimization problem. However, this approach seems more adapted to a supervised context where it is easier to estimate the distances combination weights.

Because time series data are high-dimensional objects, computing distances in the original space is often expansive, as is every step of the analysis process (storage, preprocessing, analysis, visualization, ...). In addition, in high-dimensional space, discriminating time series is complex because of the *curse of dimensionality* [VF05], an effect that tends to amplify the observations separation when the observation space dimension increases. A preprocessing step is often applied to represent time series into smaller-dimensional spaces, where this separation is easier.

1.1.4 Time Series Transformation

The feature-based approach consists in designing a meaningful condensed time series representation. This process assumes that the transformation process keeps the data informative value. Two situations can be distinguished: the first when the transformation process is known, the second when it is guided by an external criterion. Either way, this method requires prior knowledge on the time series.

Explicit transformation definition

This first category of methods (also called Non Data Adaptive) is used when the user can explicitly define the transformation that he finds relevant. The mean, median, quantiles, length, sampling frequency, first or last element's values, are instances of such features. These "single-point" features are usually easy to compute, and greatly reduce the dataset dimensions [RK09], but the feature selection step may require a long prospective work. This class of methods also includes sub-sampling, or segmenting with fixed interval windows. The Piecewise Aggregate Approximation (PAA) [Keo+01] is a popular representation based on an uniform time step interval segmentation, then compute the mean value on each segment.

On the basis of the PAA representation, the Symbolic Aggregate approXimation (SAX) [Lin+07] is one of the first developed dictionary method that regroups segment mean values belonging to the same gaussian quantile interval. This grouping produces a dictionary that can be used to recode the original time series in chains of symbols. An illustration of the method is presented in Fig. 1.2.

In the Bag-Of-Pattern (BOP) method, the same principle is applied to subsequences of a time series to form word (i.e., chains of symbols from SAX). The final BOP representation consists in the words frequencies. In the Bag-Of-SFA Symbols (BOSS) [Sch15] approach, the authors uses a truncated Discrete Fourier Transform (DFT) to represent the segment values, instead of the mean, and the dictionary is constructed by clustering the obtained Fourier coefficients. Other variants and extensions following the same scheme have been developed (WEASEL [SL17], SAXSVM [SM13], BoTSW [Bai+15], ...)

The dictionary representations reduce greatly the representation space dimensions, by segmenting both the representation space (from numeric to categorical) and the time index (from T_i points to several segments). These dictionary-based approaches allow, to some extent, the comparison of similar segments between time series, but require some strong hypothesis on time interval sizes, and produces a coarsed

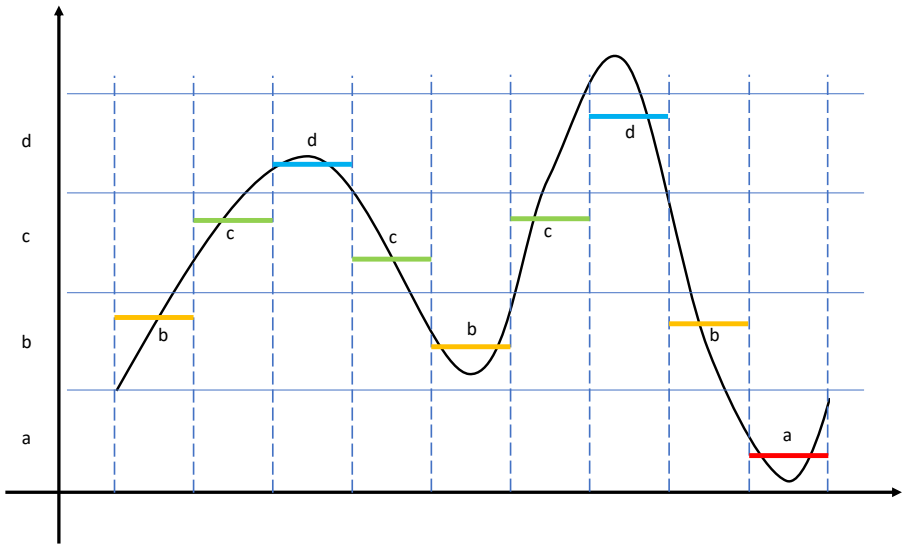


Fig. 1.2.: The Symbolic Aggregate ApproXimation (SAX) method recodes the time series with a dictionary built with the Piecewise Aggregate Approximation (PAA) representation (here, 9 segments described with a four-symbol dictionary).

segmentation of the time series phases, which is an issue for scenario-based time series clustering with asynchronous regimes.

Projections

A common choice of representation (especially suited to periodical sequences) consists in representing the time series in the frequency domain with a Fourier Transform (for instance used in [FRM94]), i.e., as a linear combination of trigonometric functions. A toy example is shown in Fig. 1.3.

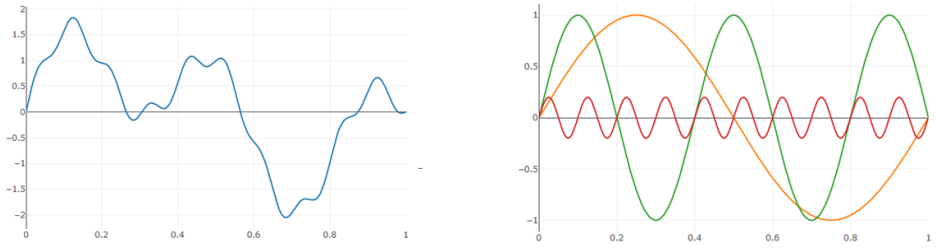


Fig. 1.3.: On the left the original time series, on the right its decomposition in a three-dimensional trigonometric basis.

Short-Time Fourier Transform (STFT) (used for feature extraction in [Wan14]) applies the Fourier Transform on sliding windows sub-samples, and represents the frequencies evolution over time, which allows to obtain a representation in both frequency and time. Closely related, the Discrete Wavelet Transform (DWT) representation applies a convolution with a wavelet function, which links window duration and frequency scale.

Another popular choices of representation consists in representing time series as weighted combinations of functions from a common functional basis (e.g., polynomial, spline, Fourier, Wavelets, ...). Combined with a Principal Component Analysis (PCA) [Hot33] step that performs a weighted linear dimension reduction on the regression coefficients, the overall transformation is known as the functional Principal Components Analysis (fPCA) [RS02], which is a popular representation for model-based time series clustering [JP14].

Closely related to the PCA, the Multi-Dimensional Scaling (MDS) uses a singular value decomposition of the distance matrix to represent the time series [LM14], and finds a low-dimensional representation that preserves the similarity measure in the original space.

When the functional regression basis is different for each time series, a common representation basis can be constructed to allow the time series comparison [CCP09].

Latent Representation Learning

Also called Data Adaptive transformations, some transformations are guided by an internal criterion optimization such as the reconstruction error or model likelihood.

In [ZMK12], the authors uses the ℓ^2 norm to extract the subsequences that best separates the time series (called *shapelets*), and use these subsequences as discriminant features.

With specific stationarity assumptions, time series can be represented with coefficients of AutoRegressive / Integrated / Moving Average models (ARIMA), that model the time series trends [Min+06], and/or with the AutoRegressive Conditional Heteroskedasticity (ARCH) [CC07] (or one of the many extended model, c.f. [Bol+08]) that model the time series variance evolution. These models parameters are inferred with ordinary least square or likelihood maximization.

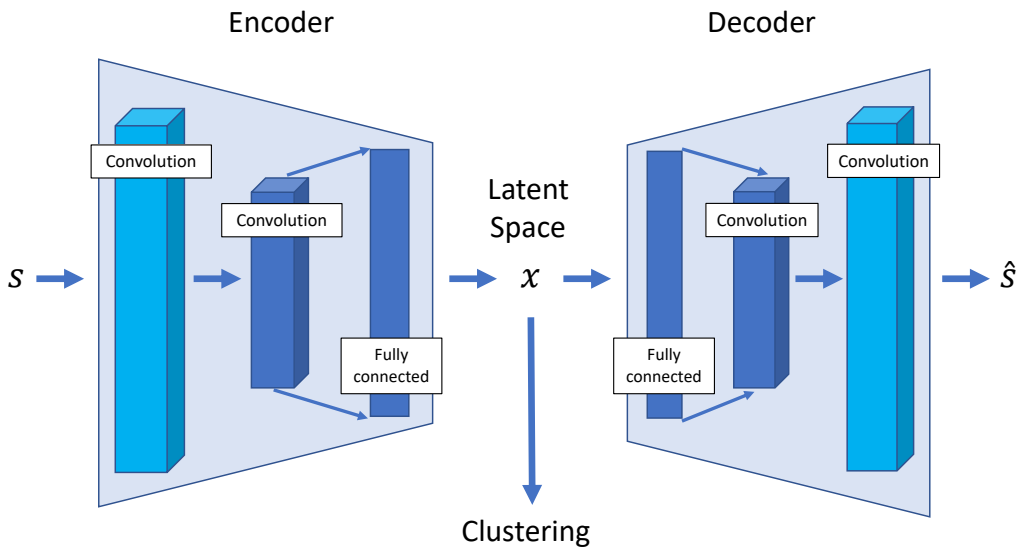


Fig. 1.4.: Non-linear time series representation based on reconstruction error optimization with a convolutional autoencoder.

The time series segmentation is another specific transformation that seeks to segment a time series according to relevant cutpoints (e.g., based on autoregressive model [TY06] or piecewise regression [Sam+11]). It can be considered a variant of the PAA representation where the time steps are not uniform but estimated from the data. This allows a more accurate representation of the time series, and also reduces the final representation space.

Several works have also addressed the reconstruction-based feature extraction based on non-linear representation. For instance, [Ngu+17] uses the t-Distributed Stochastic Neighbor Embedding [VH08] (t-SNE), that computes grouping probabilities of two time series based on relative DTW distance to other time series. The Uniform Manifold Approximation and Projection (UMAP) [MHM18] is a non-linear representation method that is also based on relative distances between elements, and has been recently used for time series clustering [PBC21].

Time series non-linear representation methods also include neural-network (NN) based representations, such as auto-encoders [Tav+20; Ric+20], that optimizes a reconstruction error $\mathcal{L}$ to produce a lower-dimensional latent space. An illustration of the autoencoder based non-linear transformation [Ric+20] is given in Fig. 1.4.

Combined with a segmentation objective, the same principle is used by the Time-series Forest [Den+13]. The Self-Organizing Map (SOM), and its NN version

DeepSOM [For+19], is another instance of nonlinear representation that expresses time series on a lower-dimensional grid such that observations that are close in the projection space are also close in the original space (with respect to the Euclidean distance).

Finally, inspired by the triangulation principle, a recent contribution [Kat16] has shown the interest of using the distances to other elements as features.

After this transformation step, the user obtains a representation of the time series in a simpler space. In this space, the final clustering is often obtained with the Euclidean distance [ASW15] associated to a clustering algorithm.

1.1.5 Clustering Algorithm

Whether in the original space or in a lower-dimensional representation, the clustering objective is to form groups of similar elements and discriminate dissimilar ones. The next subsections describe the existing types of time series clustering algorithms.

Density-Based

The density-based approach consider that a cluster is defined as an high density area in the observation space (i.e., with an important concentration of observations), separated by areas with low density. By construction, these methods creates clusters based on density contiguousness and without assumption on clusters shape.

The DBSCAN [Est+96] algorithm attributes a state to each observations, among: *core* member, *border* member, or *noise* of a cluster, based on its neighborhood cardinality. The clusters are then defined as the neighborhood of the core points. This algorithm assumes that the clusters share the same density and may fail to detect clusters with varying densities [AD15]. In the time series analysis domain, DBSCAN has been applied in the original time series space (e.g., in combination with DTW, as in [Dua+19]), or in combination with a lower-dimensional representation (e.g., after an UMAP projection as in [PBC21]). Several extensions of this algorithm have been developed (see [LP14] for a complete review).

The Density Peaks [RL14; Du+19] algorithm works in two steps: a) first it selects the cluster centroids based on local density and distance to others density peaks; b) it assigns the remaining observations to the closest centers with higher density. This algorithm has been recently applied to time series clustering in the original space

[Put+19; JLR20] but has also be seen used after a feature extraction transformation [Di+18]. The Density Peaks algorithm has also been extended with TADPole [Beg+15] that combines the DP with the DTW measure [SC78].

Clustering with Neural Network

The growing development of deep learning since their successful applications in image recognition [KSH12], has given birth to NN-based solutions in every field of data analysis. It has been used for time series transformation, as presented in Sect. 1.1.4, but also in time series clustering.

Neural Network clustering is based on two elements: a neural network architecture and a clustering objective (called the *loss*). The architecture is composed of interconnected *neurons*, that act individually as parameterized functions: each neuron takes a value as input and outputs a transformed value. The neurons are disposed in layers, such that each layer is a weighted combination of the precedent. The inference process consists in estimating the neurons parameters and the layers combinations weights by optimizing the loss.

In the time series clustering domain, the loss is often a weighted combination of a reconstruction error objective and a clustering loss. For instance, the Deep Temporal Clustering (DTC) [Mad18] uses a an auto-encoder reconstruction loss and a clustering assignment hardening loss [XGF16]. The DeepSOM method [For+00], illustrated in Fig. 1.5, uses an auto-encoder reconstruction loss, in combination with a Self Organizing Map objective

One drawback of the NN-based clustering is that the NN architecture is as a black-box for the user, equivalent to a complex and high-dimensional hyper-parameter set. It is not straightforward to understand the impacts of the architecture setting on the time series representations. As a consequence, the user obtains a partition, but doesn't know how the time series are transformed during the clustering process, and how to improve the NN architecture if the results is irrelevant. In addition, the NN training is often computationally expensive and requires massive quantities of data.

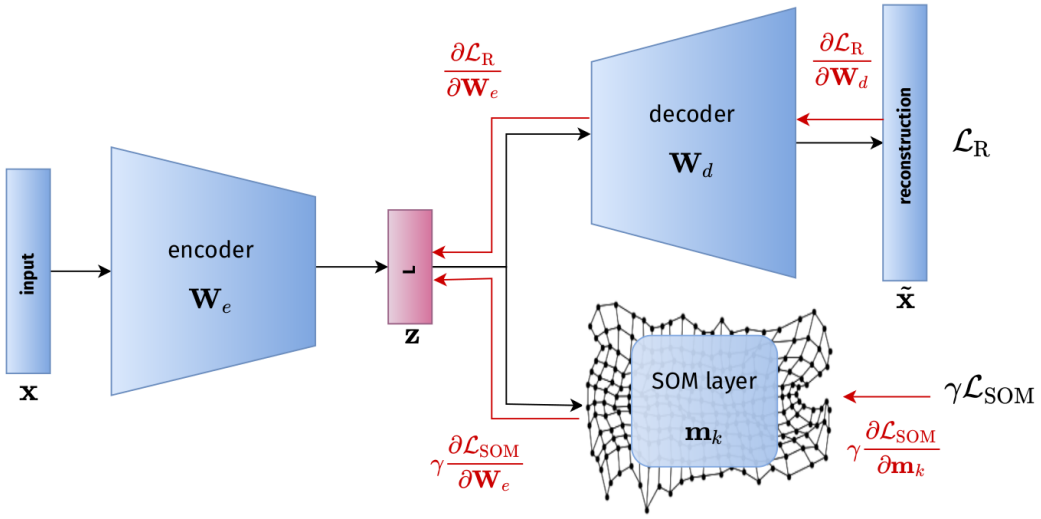


Fig. 1.5.: Deep Self Organizing Map [For+19] architecture. The clustering is based on the simultaneous optimization of the reconstruction and SOM loss functions.

Hierarchical-Based

A partition $P = \{P_k\}_K$ over $\mathbf{s}$ is defined as a finite collection of non-empty and non-overlapping clusters, such that the union of every clusters contains every elements of $\mathbf{s}$. Formally:

$$\bigcup_{k=1}^{K} P_i = \mathbf{s},$$

$$h, k \in \{1, \ldots, K\}, \quad P_k \neq \emptyset, \; P_k \bigcap P_h = \emptyset.$$

We denote $\mathbf{z} = (z_i)_n$ the membership vector, such that $z_i = k$ means that element s_i belongs to P_k, (equivalently called cluster k) and n_k the number of elements belonging to cluster k.

The hierarchical clustering algorithms seek to create a hierarchy of nested clusters. This tree is composed of a *root* containing all elements, and each branch sees a split of the previous clusters in two, until reaching the *leaves*, i.e., when every cluster is a singleton. As a result, the user obtains a hierarchy of partition, that in turn must be pruned to produce the final partition. The pruning threshold (or equivalently the cluster number) is specified by the user. This clustering can be performed with an agglomerative or divisive procedure.

The agglomerative (also called *bottom-up*) approach starts from the leaves, and recursively merges the clusters up to the one-cluster root. This deterministic approach is fully described by two elements: a dissimilarity measure (that evaluate the element-

to-element proximity), and a linkage criterion (that evaluates the cluster-to-cluster proximity).

At each cluster merge step, the linkage criterion is used to select the "best" cluster couple to merge. Among the many existing criteria, the most used in the time series clustering domain are the single-linkage [PG17], average-linkage [HT05; Łuc16] and Ward's criterion [JLR20; ZA18]. This last criterion (defined in [War63]), that we also use in our first contribution (c.f. Chapt. 2), is used to select the cluster merge that maximizes the cluster separation and minimize the dispersion inside cluster. Formally, it measures the *intra-cluster inertia*, expressed as a mean of pairwise dissimilarities:

$$\sum_{k=1}^{K} \frac{1}{2n_k} \sum_{x_i, x_j \in p_k} \mathrm{d}(x_j, x_j)^2.$$

It can be noted that this expression differs from the standard Ward's criterion expression [War63], as it uses pairwise distances inside clusters instead of mean distance to the cluster center. This pairwise-distance expression, presented by [Cha+18] and advocated by [RVN21], extends the inertia definition to non-Euclidean dissimilarity measures, which is why we use it in our first contribution (c.f. Chapt. 2) based on a string distance. The main advantage of this definition is that it does not require to estimate cluster center, which can be troublesome for some dissimilarity measures (e.g., the average DTW sequence computation is an NP-hard problem).

With this hierarchical clustering approach, the user obtains a tree partition representation such that, for every value of cluster number, the obtained partition minimizes the global intra-cluster inertia. The agglomerative approach is described in Algorithm 1.

Algorithm 1: Hierarchical Agglomerative Clustering.

Input: a dissimilarity measure and a linkage criterion ℓ
Set the initial partition with singleton clusters: $P^{(0)} = \{\{s_1\}, \ldots, \{s_n\}\}$
for $m \leftarrow 1$ **to** n **do**
    Select the best candidate merge:
    $(p^*, q^*) = \arg\min_{p,q \in P^{(m)}} \ell(p, q)$
    Remove p^* and q^* from $P^{(m)}$
    $P^{(m+1)} = \{P^{(m)}, p^* \bigcup q^*\}$
end

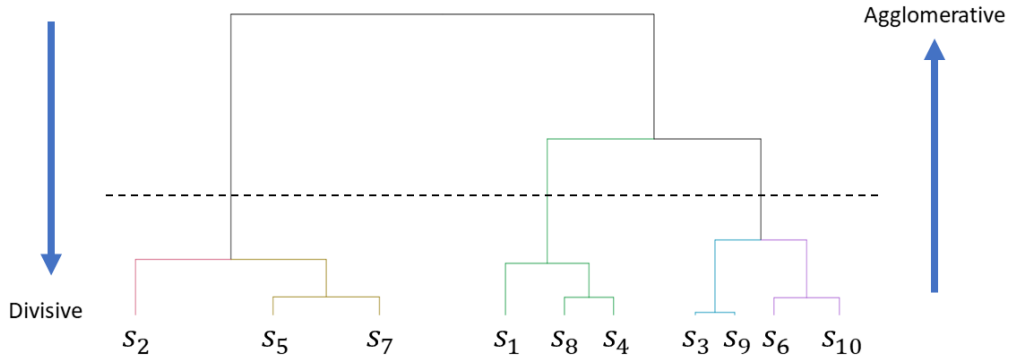


Fig. 1.6.: Dendrogram representation produced with hierarchical clustering algorithm.

As opposed to the agglomerative approach, the divisive (also called *top-down*) hierarchical clustering starts from the tree's root, containing all elements, and recursively splits the clusters to obtain the final singleton clusters. The complexity of this approach is considerable ($\mathcal{O}(2^n)$), due to the important number of splitting possibilities. The splitting approach is usually performed with an heuristic to reduce this complexity, for instance a partition-based algorithm (e.g., as in [SKK00]). The hierarchical-based clustering is illustrated on Fig. 1.6.

Partition-based

The partition-based methods seek to estimate a membership set $\mathbf{z}$, based on two inputs: a fixed number of clusters K and a similarity measure d over the observation space.

This category of algorithm includes the most used and known clustering algorithm: the k-means [Mac+67]. Starting from a random membership state, the k-means algorithm alternates two steps: a) compute $\{\mu_k\}_K$ the centers of each cluster; b) associate each observation to its closest center, with respect to d. As a result, the k-means produces a clustering and partition of the whole observation space, where each point is associated with the closest center. This result can be represented with a Voronoï diagram, shown in Fig. 1.7.

In combination with time series specific distances (e.g., with the correlation measure [PG15], Mahalanobis distance [Nel12], ...), the k-means has been used extensively and can be considered a "classical" approach. Its combination with DTW, where

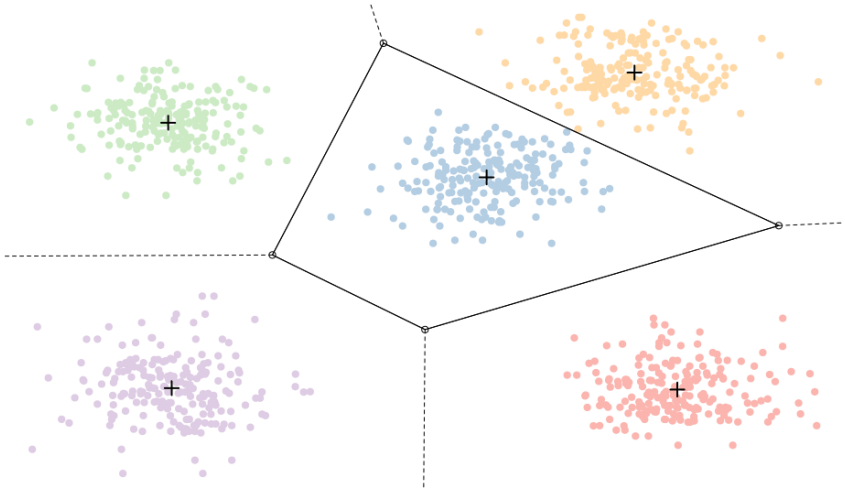


Fig. 1.7.: Voronoï diagram associated with a k-means solution.

the average sequence estimation is an NP-hard problem, has been addressed by an important amount of papers [Gup+96; NR07; NR09; PKG11; AT15; SDG16].

Other partition-based approaches are based on the Partitioning-Around-*Medoid* [KR09; NH02] (PAM) method, which consists in using the best candidate among the observed time series set $\mathbf{s}$ instead of the cluster average. This approach has also been considered as a work-around for the DTW averaging problem.

The k-means algorithm can also been applied after time series transformations (c.f. Sec 1.1.4), for instance in association with shapelet representation [ZMK12].

Fuzzy clustering (also called also called *soft*-clustering) is another approach to clustering in which observations can be assigned to several clusters. This method can also be combined with different types of distances, and has been applied to time series clustering in [TW02; Liu+18]. Fuzzy clustering can be considered a distance-oriented interpretation of the model-based clustering, with probabilities replaced by a relative similarity measure.

Model-Based Clustering

The model-based clustering approach infers a generative model over the observation space by associating a latent distribution to each cluster. In the same way partition-based algorithms assume that a given dissimilarity measure is a suitable discriminator, model-based methods assume that a given distribution is a suitable cluster generator. An illustration is shown in Fig. 1.8.

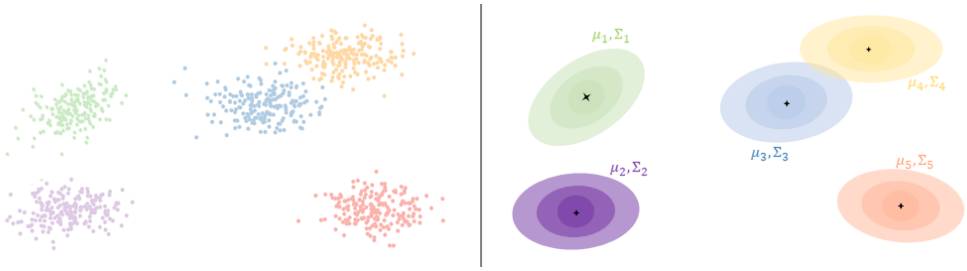


Fig. 1.8.: Illustration of a Gaussian Mixture Model assumption: there are several latent gaussian components that generate the dataset. The ellipses are the Gaussian components isoprobability contours.

The model-based approach can be applied in the original time series observations space, in that case the whole time series is represented. This approach depends on hypothesis on the data structure. In [XY02; FK08], the authors use a mixture of auto-regressive models to cluster time series generated from auto-correlated processes. Several model-based works have also addressed the clustering based on time series shapes, for instances [Gaf04] that uses a mixture of spline regression, or [GS99; Gaf04; BJ05; Cha16] that use polynomial regressions.

In some situations, it is relevant to assume that the time series are the results of chains of different states. In that case, it can be relevant to assume a piecewise-distribution modeling. In some cases, the state transition time is shared by every time series (i.e., the transitions cutpoints are synchronous), in some others it can be shared among time series in the same cluster. This last case has been addressed by the Hidden Markov Model (HMM) [Smy97] and the piecewise regression mixture [Sam+11; CN18].

The model-based clustering can also be applied after time series transformations. The fPCA (c.f., Sect. 1.1.4) is a popular choice of representation in this context [BJ11; JP13].

The Bayesian model framework allows to add prior information on the mixture parameters under the form of prior distribution. This addition allows to automatically infer the model dimension during the inference process. This framework is used for instance in [Bou12] to perform univariate time series clustering based on a grid-model.

As stated in this section preamble (c.f. Sect. 1.1.1), our use case specifications also include the detection of anomalous driving behaviors. It is one of the reasons that motivates the choice of the model-based approach, that natively includes a probabilistic anomaly detection feature. This model-based framework is presented in details in the following section.

1.2 Mixture Model Framework

1.2.1 Definition

Let denote $\mathbf{x} = (x_i)_n$ the set of vectors representing the time series after transformation in a d-dimensional real space (c.f., Sect. 1.1.4). We emphasize that all observations $(x_i)_n$ share the same dimension d. The variable x designates a random vector following the mixture density, and associated to a membership z.

Mixture-Modeling is a standard model-based clustering approach [MLR19]. This model assumes that the overall density on the d-dimensional space is a convex combination of densities belonging to the same distribution family F. The *weights* of this combination are called the mixture *proportions* $(\pi_k)_K$, and the weighted densities are the mixture *components*, $(F_k(.))_K$, with respective parameters $(\theta_k)_K$.

In the following we consider the specific case of F being the Multivariate Gaussian distribution family, as is the case in many of our applications. In this case, the components parameters are therefore $\theta_k = \{\mu_k, \Sigma_k\}$.

With the notation defined in the introduction of this section, the Mixture Model (MM) density is given by $\mathrm{p}(x) = \sum_{k=1}^{K} \pi_k F(\theta_k)$, with $\pi_k = \mathrm{p}(z = k)$, and $F(\theta_k) = \mathrm{p}(x|z = k)$. With this generative model definition, sampling an observation x is performed by first drawing a component membership $z \sim Mult(\pi)$ then drawing from $F(\theta_z)$. The associated graphical model is represented in Fig. 1.9.

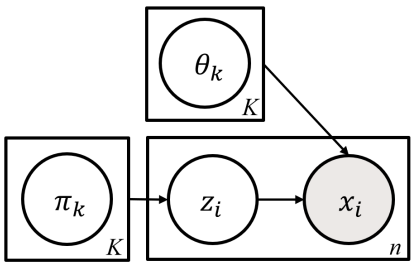


Fig. 1.9.: Mixture Model graphical model

1.2.2 Inference

With Θ the set of all parameters, the model log-likelihood of the dataset $\mathbf{x}$ is given by:

$$ln\ \mathrm{p}(\mathbf{x} \mid \Theta) = ln \sum_{\mathbf{Z}} \mathrm{p}(\mathbf{x}, \mathbf{z} \mid \Theta), \tag{1.1}$$

where $\mathbf{Z}$ is the set of all possible combination of the membership vector $\mathbf{z}$. A direct likelihood optimization estimation would require to set the derivatives of Eq. (1.1) to zero, but the summation on $\mathbf{Z}$ prevents from obtaining a closed form solution [Bis06]. Several methods have been developed to sort out this limitation, of which the Expectation Maximization [DLR77] (EM) is the most popular.

The EM is an iterative inference algorithm, that repeats two steps until likelihood convergence.

E step estimating the Expectation of the memberships posterior probabilities $\mathrm{p}(z \mid x)$ given a fixed value of Θ;

Considering fixed values of parameters Θ, the posterior distribution estimation is given by Bayes conditional probability formula:

$$\mathrm{p}(z = k \mid x) = \frac{\mathrm{p}(z)\ \mathrm{p}(x \mid z)}{\mathrm{p}(x)} = \frac{\pi_k\ \mathrm{p}(x \mid z = k)}{\sum_K \pi_h\ \mathrm{p}(x \mid z = h)} = \frac{\pi_k\ F_k(x)}{\sum_K \pi_h\ F_h(x)}.$$

In the following, we denote $\tau_{i,k} = \mathrm{p}(z_i = k \mid x_i)$ for brevity.

M step estimating Θ by Maximizing the likelihood with fixed values of the posterior distributions.

Considering fixed values of $(\tau_{i,k})_{n \times K}$ for every membership $\mathbf{z}$ of the dataset elements $\mathbf{x}$, the parameter set Θ update is given by maximizing the expectation of the complete log-likelihood [Bis06]:

$$\mathcal{Q}(\theta_{old}, \theta) = \sum_{n,K} \mathrm{p}(z_i = k \mid x_i, \theta_{old}) \ln \mathrm{p}(x_i, z_i \mid \theta),$$

with θ_{old} the parameter values used for the estimation of the posterior distribution in the previous step E. The optimization of this expectation yields the following expressions [Bis06] in the multivariate Gaussian case:

$$n_k = \sum_i \tau_{i,k},\ \pi_k = \frac{n_k}{n},\ \mu_k = \frac{1}{n_k}\sum_i \tau_{i,k}\ x_i,$$
$$\Sigma_k = \frac{1}{n_k}\sum_i \tau_{i,k}(x_i - \mu_k)(x_i - \mu_k)^T.$$

Because the posterior distribution $\mathrm{p}(z \mid x\theta)$ can be computed individually, and the M step can be estimated by batch, the EM algorithm is easy to parallelize and to distribute on map-reduce frameworks [@Spa].

With this construction, the model log-likelihood necessarily increases at each iteration [Bis06], or stagnates in case of convergence. However, the obtained solution is sensible to the EM initialization, and the EM can get stuck in locally optimal solutions.

This sensitivity to initialization can be reduced with specific strategies [BCG03; BB13; BC15; SSB17], including random search, initialization with k-means, or hierarchical divisive initialization strategies. Keeping the best result among several tries is also a popular strategy.

The Stochastic EM (SEM) is a variant of the EM that adds a membership sampling step based on the posterior distributions estimated in the E step. With this version, the likelihood monotonic increase property is not conserved, but the stochastic step allows a better exploration of the solution space and reduces the risk of getting stuck in local optima [CD85; CD92].

The Classification EM (CEM) algorithm aims to optimize the *complete* likelihood $\mathrm{p}(\mathbf{x}, \mathbf{z} \mid \Theta)$ instead of the marginal likelihood $\mathrm{p}(\mathbf{x} \mid \Theta)$. During the CEM, the $\tau_{i,k}$ are used to select the most probable partition (as opposed to the SEM sampling). This strategy can be considered as *hard* clustering.

It can be noted that the CEM and the k-means are tightly connected. The k-means is in fact strictly equivalent to the CEM in the multivariate Gaussian case, when assuming identity covariances matrices (i.e., independent variables, and shared variance among cluster and variable). From this perspective, the EM can be considered a probabilistic soft-clustering generalization of the k-means. The EM, SEM and CEM are described in Algorithm 2.

Algorithm 2: (-/S/C) Expectation Maximization Algorithm for Mixture Model Inference

Input : cluster number K, iteration number M
Output : MM parameters $\hat{\Theta}$, partition $\hat{\mathbf{z}}$
Initialize partition $\mathbf{z}^{(0)}$
Estimate initial mixture parameters $\Theta^{(0)}$
for $m \leftarrow 1$ **to** M **do**
  for $i \leftarrow 1$ **to** n **do**
    Estimate $\tau_{i,k} = \mathrm{p}(z_i^{(m+1)} = k \mid x_i, \Theta^{(m)})$
    $\tau'_{i,k} \leftarrow \tau_{i,k}$
    if *SEM* **then**
      Draw membership $z_i \sim Mult((\tau_{i,k})_K)$
      Set τ'_{i,z_i} to 1, the others to 0
    if *CEM* **then**
      Choose $z_i = \arg\max_K \tau'_{i,k}$
      Set τ'_{i,z_i} to 1, the others to 0
  Estimate $\Theta^{(m+1)}$ given $(\tau'_{i,k})_{n \times K}$
Estimate $\hat{\mathbf{z}}$.

As is the case with k-means, a model selection step is often required in association to the EM-based algorithms applications.

Based on the obtained mixture model density, it is possible to produce confidence intervals and probabilistic outlier detection (e.g., by comparing the observations marginal likelihood).

1.2.3 Model Selection

The EM requires to make several assumptions: on the number of clusters K (as the k-means) but also on the component distribution family F. In the Multivariate Gaussian case for instance, the clusters can be assumed with equal covariance matrices (homo-scedasticity), with diagonal covariance matrices (variable independency), ... These assumptions can be a strong limitation to real-life applications, where the number of clusters or the covariance matrices properties are rarely known.

To circumvent these limitations, model selection strategies have been developed, on the basis of selection criteria and search strategy.

Model Selection Criterion In the parametric model-based framework, several model selection criteria have been developed. Most of them consists in a score that

penalizes the model likelihood by its complexity (e.g., the Akaike Information Criterion (AIC) [Aka74], the Bayesian Information Criterion (BIC) [Sch78], the Integrated Completed Likelihood (ICL)) in an attempt to approximate the integrated likelihood[BCG00]). Under the hood, this approach aims at optimizing the bias-variance trade-off, and reducing the model over-fitting. Other non model-based approach have also been developed, as the recent Stadion [For+21] that measures the time series clustering stability against dataset transformation.

We chose the BIC in our parametric application model selection (Chapt. 2), as it has been shown in [BCG00] that it well suited for density estimation tasks. Its definition is:

$$BIC = ln\ \mathrm{p}(\mathbf{x} \mid \Theta) - \frac{C}{2}\ ln(n),$$

with C the parameter number. However, the BIC's drawback is that it tends to overestimate the number of clusters in order to obtain better density estimation, which may happen when the mixture components are not adapted to the true cluster shape. This over-estimation phenomenon has been taken into account in our application, and several measures have been implemented to mitigate the problem (e.g, Sect. 2.2.4).

Search Strategies Several search strategies have been developed in the parametric setting for selecting the best model structure.

The grid-search (also called exhaustive search) tries every model possibility and select the best, i.e., the model optimizing the model selection criterion value.

The hierarchical search is a divisive hierarchical algorithm (c.f. Sect. 1.1.5), that iteratively splits the obtained clusters until the model selection criterion stops improving.

The random search [BB12] is another popular hyper-parameter optimization strategy, that consists in exploring the solution space randomly.

These model selection strategies make several assumptions. For instance, the grid search assumes that the true number of clusters is within the grid. The greedy search is a divisive hierarchical heuristic, and assumes that the clusters construction is based on a binary tree structure. When the parameter set to explore is high-dimensional, the random search may fail to find the sweet spots.

For all of these reasons, the parametric model selection can sometimes become problematic problematic. For instance, when the number of possible models is extremely important.

Another solution to mixture model selection consists in estimating the model structure during the inference process. This is one advantage of using the Bayesian framework and assuming prior distributions over the model parameters, as presented in the following section.

1.3 Bayesian Non-Parametric Modeling

1.3.1 Alternative MM Representation

With the introduction of the discrete distribution $G = \sum_{k=1}^{K} \pi_k \delta_{\theta_k}$, with δ_θ the Dirac delta function, the MM admits the alternative equivalent representation:

$$x_i \mid \theta_i \sim F\left(\theta_i\right), \theta_i \sim G, i \in \{1, .., n\}.$$

In this setting, each observation x_i is drawn from the distribution F with specific θ_i, and each θ_i drawn $i.i.d$ from distribution G. G is a linear combination of Dirac delta functions, weighted by the proportions π, and acts as a distribution over θ_i, i.e., G is a distribution over distributions. An instance of G, with $K = 4$ is shown in Fig. 1.10, in the case where $(\theta_k)_K$ are modes of bivariate normal distributions.

Because G is a finite discrete distribution and the number of cluster K is (supposedly) specified lower than n, some groups of observations share a common parameter, which creates a partition. The Dirichlet Process Mixture Model (DPMM) can be seen as a Bayesian extension of the MM where the number of components is infinite.

1.3.2 Dirichlet Process Mixture Model

The DPMM (c.f. Fig. 1.11) is a Bayesian Non-Parametric (BNP) model that assumes the presence of an infinite discrete number of latent components, of which only a fraction is observable in the dataset. In this setting, G is given an additional Dirichlet Process (DP) prior distribution that takes two hyper-parameters: a concentration

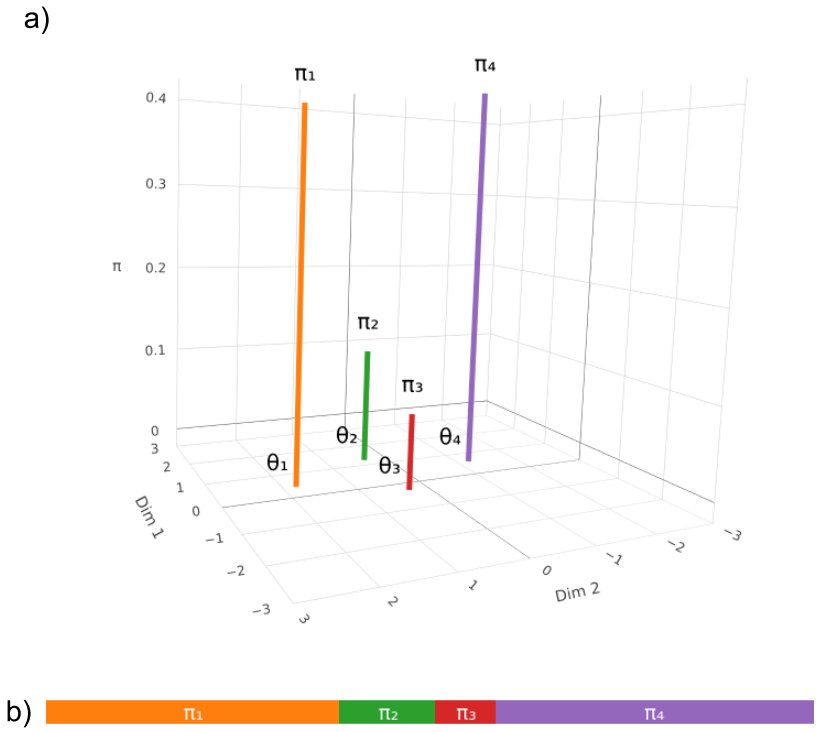


Fig. 1.10.: a) Illustration of G with four components, when the component parameters are bivariate normal distributions modes, i.e, elements of $\mathbb{R}^2$. b) Stick representation of the component proportions.

α and a base distribution G_0. In the following we denote $\chi = (\alpha, G_0)$ the set of hyper-parameters. The DPMM is formally defined by:

$$x_i \mid \theta_i \sim F(\theta_i), i \in \{1, .., n\},$$
$$\theta_i \mid G \sim G, G \sim DP(\alpha, G_0).$$

Two alternative representations are often use to help interpreting the DP: the *Chinese Restaurant Process* (CRP), which is a distribution over partitions, and the *Stick-Breaking* (SB) scheme, that models the proportions construction. These two representations are strongly connected, as one can be derived from the other [Mil19].

Chinese Restaurant Process

The CRP is a distribution over partitions that depends on a parameter concentration α. The process is named after the culinar metaphor that is often used to illustrate its definition: a chinese restaurant contains several tables (each table representing a cluster k), and each table offers one unique dish specific to that table (representing a

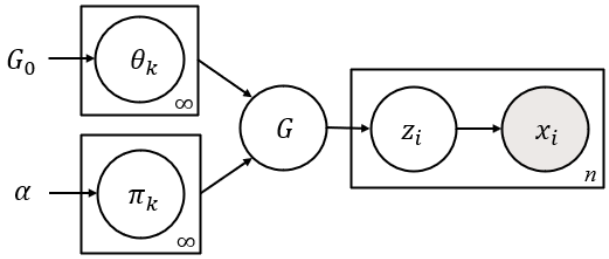


Fig. 1.11.: Dirichlet Process Mixture Model graphical model

component parameter θ_k). A given number of customers are already seated at each table (denoted n_k, which represents the number of observations already assigned to cluster k). This state defines a partition P of the existing customers, equivalently represented by the membership vector z. The CRP defines that a new customer entering the restaurant will choose a table with probability:

$$\mathrm{p}(z_i \mid \mathbf{z}_{-i}, \alpha) \propto \begin{cases} \dfrac{n_k}{n-1+\alpha} & \text{existing cluster/table } k, \quad (1.2) \\ \dfrac{\alpha}{n-1+\alpha} & \text{new cluster/table.} \quad (1.3) \end{cases}$$

With this definition, the density of a partition P containing K clusters with cardinality $(n_k)_K$ can be obtained iteratively, with the product of conditional memberships probabilities:

$$\begin{aligned} \mathrm{p}(\mathbf{z}) &= \mathrm{p}(z_1 \mid z_2, \ldots, z_n)\mathrm{p}(z_2 \mid z_3, \ldots, z_n) \ldots \mathrm{p}(z_n) \\ &= \frac{\alpha^K}{\prod_i (i-1+\alpha)} \prod_k (n_k - 1)!. \end{aligned} \quad (1.4)$$

The second line expression is obtained by noting that:

- Because there is K clusters in P, the "new cluster" discovery option has been chosen exactly K times, which explains the α^K numerator.
- The denominator $\frac{1}{n-1+\alpha}$ is present in both membership probabilities Eq. (1.2) and Eq. (1.3), which yields the product $\prod_i (i-1+\alpha)$.
- For each cluster k, the final cardinality n_k implies that the choice "entering existing cluster k" has been made exactly n_k times, with n_k varying from 1 (for the second customer choosing table k) to $n_k - 1$ (the last time a customer entered this table). This yields the term $1 \times 2 \times .. \times n_k - 1 = (n_k - 1)!$.

As shown by [Ant74], the expression in Eq. (1.4) corresponds to the density of a partition in a DP process. Moreover, it can be noted that this density is invariant to the order of the customer arrival.

With this construction in mind, the DP can be seen as a CRP prior on the memberships that also integrates the predictive distribution information.

Stick-Breaking

The stick-breaking process is another representation that focuses on the construction of the mixture proportions $(\pi_k)_\infty$. This construction comes from [Set94] which states that, if $G \sim DP(\alpha, G_0)$, then it can be written as:

$$G = \sum_k \pi_k \delta_{\theta_k},$$

the component distributions θ_k follows the prior G_0, and the weights $(\pi_k)_\infty$ are obtained with the following *stick-breaking* construction:

$$\pi_i(\mathbf{v}) = v_i \prod_{j=1}^{i-1} (1 - v_j), \quad \mathbf{v} = (v_i)_{\{1,\dots,n\}}, \quad v_j \overset{\text{i.i.d.}}{\sim} \text{Beta}(1, \alpha). \tag{1.5}$$

This construction is named after the following construction metaphor. Starting from a stick with length one, the first proportion π_1 is the product of the stick length (1) with $\text{Beta}(1, \alpha)$ sample v_1. The stick is then broken at the value of π_1. The second proportion π_2 is the product of the remaining stick length (1-π_1) times another $\text{Beta}(1, \alpha)$ sample, v_2... Each new weight is the product of a sample of a $\text{Beta}(1, \alpha)$ with the remaining of the stick length. Because the remaining of the stick is always strictly positive (while tending toward zero), this process can be used to produce an infinite set of proportions.

With this construction, [Set94] proved that the obtained distribution $G = \sum_{k=1}^{\infty} \pi_k(\mathbf{v})\delta_{\theta_k}$ follows a DP(α,G_0) distribution.

The DPMM can be seen as a BNP extension of the MM where the number of components is infinite. Because the observation set is finite, only a finite number of clusters (noted K_{obs}) are observable. Moreover, because the number of observed cluster grows in fact as $\mathcal{O}(\log n)$ [Teh+06], some observations share the same cluster, which creates a clustering. This property is illustrated on Fig. 1.12. It it possible to influence the value of K_{obs} by tuning the concentration parameter α, which

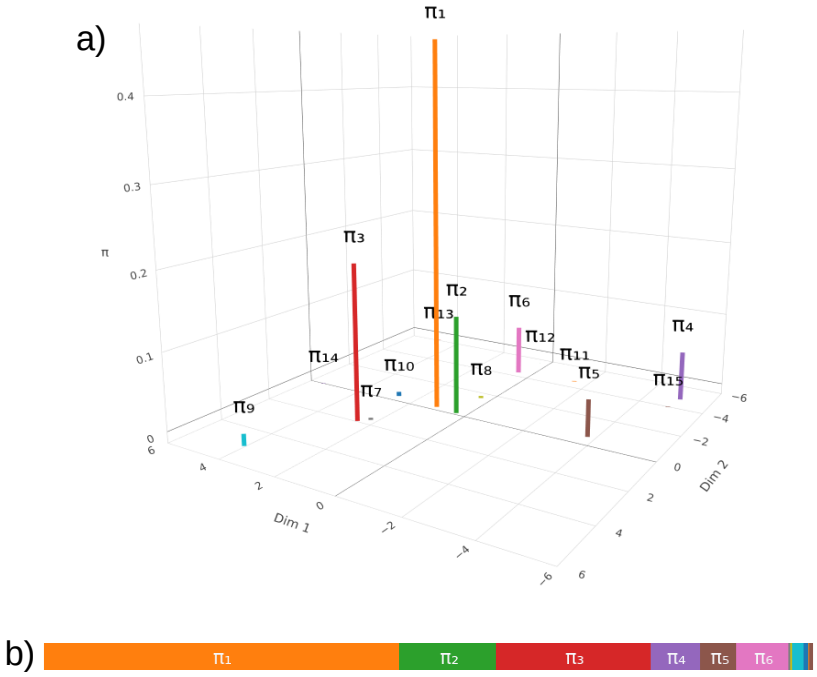


Fig. 1.12.: a) Illustration of the first components of G, when the component parameters are bivariate normal distributions modes, i.e, elements of $\mathbb{R}^2$. b) Stick representation of the component proportions. This illustration illustrates the link between the Stick-Breaking construction and the fast proportion decrease.

influences the probability of new cluster discovery. The role of α is also visible in the expression of K_{obs} distribution [Ant74]:

$$\mathrm{p}(K_{obs} = K \mid \alpha) = |S_{n,K}| n! \alpha^K \frac{\Gamma(\alpha)}{\Gamma(\alpha + n)}, \tag{1.6}$$

where $|S_{n,K}|$ is the unsigned Stirling number of the first kind.

The DPMM can automatically adapt to dataset augmentation (or diminution) by adjusting the observable component number, which makes it relevant for exploration tasks where the user can use additional resources to explore new observation space areas.

1.3.3 Inference

Several methods have been developed to infer DPMM's parameters. For instance, based on variational inference [BJ+06] or Markov chain Monte Carlo (MCMC)

[Esc94; Nea00]. 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 [WDX13; Meg+19] have made MCMC processes scalable, which rehabilitates their use for industrial purposes.

In the following we present two MCMC algorithms, based on the Gibbs Sampler estimation. The Gibbs Sampler objective is to mimic the posterior distribution $\mathrm{p}(\mathbf{z} \mid \mathbf{x}, \chi)$ by sampling values for each z_i from its conditional distribution given the data and the remaining observation $\mathrm{p}(z_i \mid \mathbf{x}, \mathbf{z}_{-i}, \chi)$.

Gibbs Sampler

The Gibbs Sampler, corresponding to algorithm 2 of [Nea00], consists in alternating the update of the clusters memberships and the clusters parameters. In a first step, the memberships $\mathbf{z}$ are updated one by one. Each membership z_i is sampled from the conditional distribution given the observations $\mathbf{x}$, the remaining memberships and the associated parameters: $\theta_{-i} = (\theta_j)_{j \neq i}$, according to:

$$\mathrm{p}(z_i \mid \mathbf{z}_{-i}, x_i, \chi) \propto \begin{cases} \dfrac{n_k}{n-1+\alpha}\ \mathrm{p}(x_i \mid z_i = k), & \text{existing cluster } k, \quad (1.7) \\ \dfrac{\alpha}{n-1+\alpha}\ \mathrm{p}(x_i \mid \chi), & \text{new cluster,} \quad (1.8) \end{cases}$$

where $\mathrm{p}(x_i \mid z_i = k) = F(x_i, \theta_k)$ is the density function associated with component k. It is interesting to note the presence of the same prior membership expressions $\frac{n_k}{n-1+\alpha}$ and $\frac{\alpha}{n-1+\alpha}$ than in the CRP definition.

In Eq. (1.8), the density $\mathrm{p}(x_i \mid \chi)$ is the prior predictive distribution of x_i, that can be obtained by marginalizing over the component parameter:

$$\mathrm{p}(x_i \mid \chi) = \int_\theta \mathrm{p}(x_i, \theta \mid G_0)\ d\theta = \int_\theta \mathrm{p}(x_i \mid \theta)\mathrm{p}(\theta \mid G_0)\ d\theta.$$

This integral has a closed-form when F is conjugate to G_0. With F the multivariate Gaussian density and G_0 the Normal Inverse Wishart (NIW) conjugate prior with hyper-parameter $(\mu_0, \kappa_0, \Sigma_0, \nu_0)$, this integral sums up to a multivariate t-distribution density (c.f. derivations in Appendix A.1):

$$t_{\nu_0 - d + 1}\left(x | \mu_0, \frac{(\kappa_0 + 1)\Psi_0}{\kappa_0(\nu_0 - d + 1)}\right).$$

After this first step, i.e., after having updated every membership once, the clusters parameters are also updated by sampling according to the posterior distribution $\mathrm{p}(\theta_k \mid \mathbf{x}_k, G_0)$. This distribution is also an NIW distribution, with hyper-parameters updated with $\mathbf{x}_k = \{x_i : z_i = k\}$ the observations belonging to cluster k. The hyper-parameters update equations are given [Mur07] by:

$$\kappa_k = \kappa_0 + n_k,\ \nu_k = \nu_0 + n_k,\ \mu_k = \frac{\kappa_0 \mu_0 + \sum_{\mathbf{x}_k} x}{\kappa_n}, \tag{1.9}$$

$$\Sigma_k = \Sigma_0 + \sum_{\mathbf{x}_k} xx^T + \kappa_0 \mu_0 \mu_0^T - \kappa_k \mu_k \mu_k^T. \tag{1.10}$$

These two steps (membership update and parameter update) are repeated until convergence or for a given iterations number.

At each iteration m, n memberships are updated, and each membership update involves the computation of $K^{(m)}$ existing clusters membership (Eq. (1.7)) and one new cluster memberships (Eq. (1.8)). The prior predictive distribution in Eq. (1.7) is fixed given the observations and prior G_0, and can be computed once before the inference and cached. This first computation has a complexity $\mathcal{O}(nd^3)$, due to the multivariate t-distribution estimation cost.

The K existing membership probabilities must be computed again at each iteration, but this computation is quite cheap as it only consists in a multivariate Gaussian density estimation ($\mathcal{O}(d^2)$). The membership update step has therefore a $\mathcal{O}(n\bar{K}d^2)$ complexity, with $\bar{K} = \max_{m=1}^{M} K^{(m)}$ an upper bound of the cluster number, and M the maximal iteration number.

During the parameter update step, each observation contributes to the update of the NIW posterior distribution associated with its current membership. This update step complexity is bounded by the covariance estimation term, in $\mathcal{O}(nd^2)$. The global complexity is therefore:

$$\mathcal{O}(nd^3 + Mn\bar{K}d^2).$$

The sequential aspect of the inference may seem a prohibitive feature at first sight, as it prevents the membership update parallelization. However, some works have addressed the problem [WDX13; Meg+19], and turned this algorithm into distributed and scalable versions.

One drawback of this inference process is that the clusters parameters are updated only once per iteration, i.e., after the update of every memberships. Because of this low update rate, the convergence can be slow. In the next algorithm, the cluster memberships state is taken into account at all time.

Collapsed Gibbs Sampler

The Collapsed Gibbs Sampler, corresponding to algorithm 3 of [Nea00], is a natively fast MCMC's method that also assumes the prior distribution G_0 conjugate to the density family F and skips the parameter sampling step. In this method, the prior conjugacy assumption enables the closed-form computation of the posterior predictive distributions, that is used to estimate posterior cluster membership probabilities. The density F in Eq. (1.7) is replaced by this posterior distribution:

$$\mathrm{p}(z_i = k \mid x_i, \mathbf{x}_k, G_0) = \int_\theta \mathrm{p}(x_i \mid \theta)\mathrm{p}(\theta \mid \mathbf{x}_k, G_0)d\theta. \tag{1.11}$$

In this expression, $\mathrm{p}(\theta \mid \mathbf{x}_k, G_0)$ is the same distribution than the one used for parameter update in the (non-collapsed) Gibbs Sampler from Sect. (1.3.3), i.e., a NIW distribution with hyper-parameters obtained by updating the prior G_0 parameters with Eq. (1.9) and Eq. (1.10).

With these updated hyper-parameters, the integral Eq. (1.11) is equivalent to the following multivariate t-distribution:

$$t_{\nu_n - d + 1}\left(x | \mu_n, \frac{(\kappa_n + 1)\Sigma_n}{\kappa_n(\nu_n - d + 1)}\right).$$

This direct computation of the posterior membership distribution allows to *collapse* the clusters parameters, and to avoid their estimation during inference.

At each iteration, the update of one row-membership z_i involves the computations of $(K^{(m)} + 1)$ row-cluster membership probabilities ($K^{(m)}$ existing clusters and the new cluster). For each cluster k, the membership probability computation involves the cluster hyper-parameter update (based on Eq. (1.9) and Eq. (1.10)) and one posterior predictive distribution density estimation.

In practice, instead of updating the cluster k posterior hyper parameters $(\mu_k, \kappa_k, \Psi_k, \nu_k)$ with the full cluster contents x_k at each iteration and at each membership update, it is more efficient to cache these hyper-parameters, and only remove

the observation $\mathbf{x}_i$ from its previous cluster before the membership probabilities computation, and add it to its new cluster after sampling, which is simply $\mathcal{O}(d^2)$. The posterior membership density estimation has a complexity $\mathcal{O}(d^3)$ because of the multivariate t-distribution computation. This membership must be performed once for each observation and iteration, which gives a total complexity of

$$\mathcal{O}(Mn(d^2 + \bar{K}d^3))$$

This complexity is comparable to the one of the Gibbs Sampler, and highlights the interest of keeping a representation space dimension low. Both the Gibbs Sampler and Collapsed Gibbs Sampler are MCMC that produces samples that mimics the joint distribution $\mathrm{p}(\mathbf{x}, \mathbf{z} \mid \chi)$. These samples must in turn be processed to obtain the final partition.

Inferring the final partition

These Gibbs Sampler outputs a chain of partitions, that correspond to samples from an approximation of the posterior distribution. These sampled must, in turn, be aggregated over the iterations, usually after a given number of burnin iterations. Given a chain of partitions $(\hat{z}_m)_M$, the objective is to estimate

$$\hat{z} = \arg\min_{\mathcal{Z}} \sum_m \mathrm{d}(\hat{z}_m, z),$$

with d a dissimilarity measure between permutations and $\mathcal{Z}$ the set of all possible row-partition with size n. This *consensus partition* estimation is an NP-complete problem [KM86], that several works have addressed in the past (c.f. [Xan14] for an extensive review).

This question is crucial for real-life applications, where the inference process can be sensitive to initialization and produce different solution. One objective of the consensus partition is to obtain a "better" partition, in the sense that it is more stable and relevant. The consensus partition also has the advantages to resolve the label switching problem.

We advocate to use the recent method from [GOK18], which proposes an efficient extension of the combinatorial optimization method from [HA07] that constructs the partition with minimal partition-distance [KLB05] to the samples, without assumptions on the final number of clusters or on the clustering structure. The complete inference process is described in Algorithm 3.

Algorithm 3: DPMM Inference

input : α, G_0, Iteration number M
output: Estimated row-partition $\hat{\mathbf{z}}$

Initialize $\mathbf{z}^{(0)}$
for $m \leftarrow 1$ **to** M **do**
    for $i \leftarrow 1$ **to** n **do**
        Remove $\mathbf{x}_i$ from its cluster.
        Compute $\mathrm{p}(z_i \mid \mathbf{z}_{-i}, X, \alpha, G_0)$ as defined by eq. (1.7) and (1.8)
        Sample $z_i^{(m+1)}$
        Add $\mathbf{x}_i$ to its new cluster.
Compute the average partition $\hat{\mathbf{z}}$ on the last samples (c.f. averaging methods in Sect. 1.3.3).

The methods presented in this previous section are dedicated to univariate time series clusters. When several temporal variables are present in a dataset, additional difficulties appear, that must be addresses with specific methods.

1.4 Multivariate Time Series Model-Based Clustering

As stated in this section preamble (c.f., Sect. 1.1.1), the time series clustering is applied in many domains and contexts. In most of the use cases, the time series can be described by several variables simultaneously, i.e., the time series datasets are multivariate. For instance, in Health, the patient state is often observed with several sensors (e.g., temperature, hearth rate, blood pressure, toxin concentration). In finance, the analyst is often interested in understanding the correlation between the stock price of several items.

In our industrial applications, the ADAS numerical validation also generates an important number of variables (position, angle, activation systems, radar, engine, ...). This Multivariate Time Series (MTS) context requires multi-dimensional extensions of the univariate case methods, with additional difficulties that we present in the next sections.

1.4.1 Notation

As a natural extension of the univariate case notation, let denote $S = (s_{i,j})_{n \times p}$ the matrix of time series observations, such that each cell $s_{i,j} = (s_{i,j}(t))_{T_i}$ is a time series indexed by T_i. This notation highlights one key property of this dataset: the

time series length T_i is shared among simulation (i.e., matrix row) and not among variable (i.e., matrix column). From the ADAS perspective, it comes from the fact that the simulated variables observe a simulation with a common duration. An instance of MTS dataset is given in Fig. 0.3.

Let denote $(X_{i,j})_{n\times p}$ the dataset transformed in a lower dimension space, where each cell share the same dimension d. In line with this matrix notation, we denote (abusively and for brevity) $\mathbf{x}_{i,.}$ the i-th row/simulation of X, and $\mathbf{x}_{.,j}$ its j-th column/variable.

1.4.2 Multivariate Context Constraints

In the multivariate setting, the observation space dimension is the product of all the individual temporal variable dimension, which aggravates the separation effect of the dimensionality curse. Even after individual time series transformations (which corresponds to the analysis of dataset X where each $x_{i,j}$ is a d-dimensional vector), the representation space has an $d \times p$ dimension. In our application, p can be as large as several hundreds, which makes the transformation space dimension prohibitively important even for small values of d.

Multivariate clustering must therefore include one or several dimension reduction features to address this dimension increase problem.

In the presence of several variables, the methods must also take the variables relationships into account. For instance, the Gaussian MM model use specific hypothesis on the covariance matrices to model the variables correlation.

In the following we describe some existing multivariate clustering methods, before introducing the block-clustering framework.

1.4.3 Related Work on Multivariate Time Series Clustering

Existing MTS methods are the direct result of an existing univariate method extension.

Multivariate Similarity Measure

An important number of papers have been focused on the extension of DTW [SWK15; Sho+17]. In [Sho+17], the authors discriminates the possible DTW extensions in two categories:

- *Independent* strategies (also called Match-By-Dimension), that compute the DTW dissimilarity on each variable independently, and use the sum as global dissimilarity score.
- *Dependent* strategies (also called overall Matching), that sums up the multi-dimensional DTW dissimilarity step by step. One main advantage of this approach is that the time axis warping is shared among dimensions, which conserves the time dependency information in the dissimilarity evaluation.

In a recent work, [Shi+21] uses this independent/dependent categorization as a basis for a broad similarity extension framework, and applies it to several existing measures (L2 Norm, Longest Common Subsequence, Move-Split-Merge, ...).

At the crossover between feature extraction and MTS similarity, several MTS similarity measures have been defined on the basis of Singular Value Decomposition (SVD). It includes:

- Weighted Sum SVD (WSSVD) [SY03], which consists in computing the scalar product of the MTS matrices eigenvectors, weighted by the eigenvalues.
- PCA similarity factor (S_{PCA}) [SS05] considers the independent PCA projection of two MTS, keeping only the first d axes, and evaluates the difference between every PCA axes of the first projection with every PCA axes of the second, as the squared value of their angle cosine, weighted by the eigenvalues
- Extended Frobenius norm (Eros) [YS04] is close to S_{PCA}, but considers the absolute value of the angle cosines, and considers every axes of the PCA representations.

Multivariate Transformation

The feature extraction strategy (c.f., Sect 1.1.4), which consists in representing time series with discriminating characteristics, has also been an important research field in the last years.

In the similar way than the MTS similarity measure can be categorized in dependent or independent extensions, the transformation can lead to an independent transformation (each dimension is individually transformed), or dependent transformation (the overall MTS is transformed). Note that the strategy choice has no implications on the final dataset dimensions, as independent transformations can lead to an univariate dataset of representations, and dependent transformations may produce multi-dimensional representations.

Few works have addressed the unsupervised transformation, compared to the univariate case. For instance, [DSP05; WWW07] that perform single-point feature (mean, deviation, quantile, ...) extraction.

The supervised feature extraction methods have received more attention. The vector linear autoregression [TP94] method uses the autocorrelation to represent the dynamical information in the MTS. For instance, [DUr+14] uses a multivariate wavelet transform and uses this representation as basis for the training of a SOM.

The PCA projection has also been extensively used for MTS representation. In [Li+13], the authors use the common principal component analysis [Li19] representation, which consists in estimating a covariance matrix on each MTS independently, perform the PCA on the basis of the covariance matrices average, and projects the MTS in this common representation space. An important number of neural-network representations have also been recently (an extensive review is available in [Faw+19]). For instance, [II20] uses a deep embedding based on a recurrent auto-encoder, or [Kip+18; PRB19] that models the interaction between MTS with a latent interaction graph.

As extensions of the univariate u-shapelet [ZMK12], multivariate u-shapelets methods have been developed [Spe+18; OW20], based on dependent or independent extensions of the DTW.

Another approach consists in extracting relevant subsequences with a multivariate segmentation (e.g. with a model-based joint segmentation [DTS07; Cha+13; HNB19]).

Multivariate Time Series Clustering Algorithm

As in the univariate case, MTS clustering algorithms can be based on multidimensional models, similarity measures, in the original space or in a lower-dimensional representation. The clustering algorithms can be categorized in the same taxonomy than the univariate methods:

- The most popular partition-based algorithm, k-means, has been applied in [DSP05] on single-points features, and in [Lee+20; SRV21] with multivariate DTW in the original space. In [II20], the k-means is applied on a non-linear representation based on auto-encoder representation.
- In [CS08], the authors have extended the density-based algorithm DBSCAN to handle MTS.
- The Hierarchical clustering (used for instance in [HK21]) can be applied directly on any MTS dissimilarity matrix, and does not require additional extensions.
- Model-based clustering of MTS has been addressed by [Kir05; GGM14] with multivariate HMMs

Extending an univariate time series clustering method is not straightforward, for several reasons. First, when several variables are present, a new information emerges: the relationship between variables. This information can take several form (same row-partition, same distribution, ...).

Another difficulty is that, when the number of variable is important, the model dimensions also become high, which can in turn cause inference issues and amplify the curse of dimensionality effects.

By assuming the presence of a latent variable partition, such that variables in a given cluster share the same distribution, the Latent Block Model addresses both the model dimension problem and the variable relationship modeling.

1.5 Multivariate Clustering With Block Structure

As described in Sect. 1.4.3, a first approach to MTS dataset clustering consists in producing only a row partition without considering variable partition. An illustration is displayed on Fig. 1.13-a).

The block clustering methods extend this principle by inferring a column-partition and one or several row-partitions. The crossing of these row and column partitions divides the MTS dataset into sub-matrices, called blocks.

In the following we detail the existing work on model-based coclustering and multi-clustering.

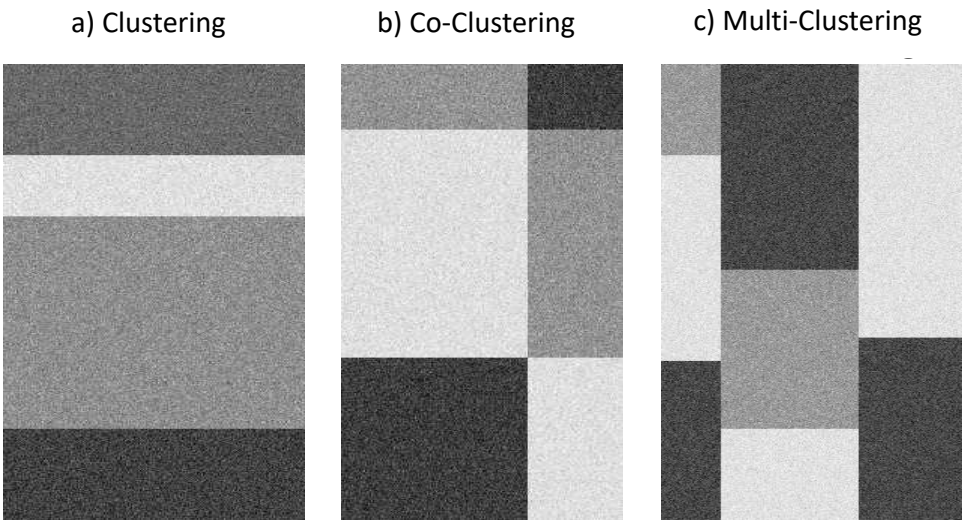


Fig. 1.13.: Comparisons of the multivariate clustering, coclustering, and multi-clustering structures.

1.5.1 Coclustering

The coclustering, also called biclustering, simultaneous clustering, two-way clustering or two-mode clustering, consists in estimating simultaneously a column partition and a row partition, which creates a block representation of a dataset. The associated structure is represented on Fig. 1.13-b).

The following section presents the related works on the topic, focusing on the models used in this thesis (see [BL15; PGA15], for exhaustive reviews).

Originally developed by [Goo65] and [Har72]), coclustering methods are specifically designed to deal with multivariate datasets. These methods have been applied in various domains: Genetics [Hed+01; Han+02; Klu+03], Biology [Xie+19], Sentiment Analysis [CHZ15], Text Mining [Dhi01], and recommendation systems [GM05; Dar+09; FZZ20].

Several approaches have been developed to address the coclustering representation. A group of methods are based on matrix reconstruction and on the optimization of a dissimilarity-based criteria. It is the case of the *CRO* algorithms (CROEUC, CROBIN, CROKI2 [Gov83] and CROINFO [GN13]). For instance the CROEUC algorithm (designed for quantitative dataset coclustering) minimizes the least square criteria:

$$\sum_{k,l} \sum_{x \in X_{k,l}} (x - \mu_{k,l})^2,$$

where $\mu_{k,l}$ is the mean of block (k, l). Some other dissimilarity-based methods express the coclustering as a matrix factorization [LS01], [Din+06] or an optimal transport problem [Lac+17]. The problem has also been addressed with NN-based optimization [Xu+19] and evolutionary algorithm [BPZ04].

Existing coclustering variants includes the Stochastic Block Model [NS01; CL15] (which performs a block clustering based on a similarity graph observation-observation) and the diagonal Latent Block Model [Roo95; Li05; LN11; LN17] which adds the constraints of a block-diagonal structure.

In the following we detail the model-based coclustering approach.

1.5.2 Model-Based Coclustering

Model Definition

The Latent Block Model (LBM), first introduced in [GN03], assumes the presence of block components, such that the contents of a given block follow independently the same component distribution. The number of row-clusters K and column-clusters L are specified by the user.

The method has been applied to binary [GN08], contingency [GN07], ordinal [JB18] and continuous data [NG10].

In the following, in addition to $\mathbf{z} = (z_i)_n$ that designates the observation partition (equivalently called *row-partition*), the vector $\mathbf{w} = (w_j)_p$ designates the variable partition (equivalently called *column-partition*) such that $w_j = l$ means that the variable $\mathbf{x}_{\cdot,j}$ belongs to the column-cluster l. p_l denotes the column-cluster l cardinality. In the multivariate continuous case and with respect to this notation and the notation detailed in Sect. 1.4.1, the model likelihood is given by:

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

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 $\prod_{i=1}^{n} \mathrm{p}(z_i) = \prod_{i=1}^{n} \pi_{z_i}$, with $\pi = (\pi_k)_K$ the row proportions, and $\mathrm{p}(\mathbf{w}) = \prod_{j=1}^{p} \mathrm{p}(w_j) = \prod_{j=1}^{p} \rho_{w_j}$, with $\rho = (\rho_l)_L$ the column proportions. The conditional density of X given the block memberships, is $\mathrm{p}(X|\ \mathbf{z}, \mathbf{w}) = \prod_{k,l} \prod_{x\in X_{k,l}} \mathrm{p}(x \mid \theta_{k,l})$, with $x \sim F(\theta_{k,l})$. The complete parameter set is denoted $\Theta_{LBM} = (\pi, \rho, (\theta_{k,l})_{K\times L})$. The corresponding graphical model is displayed in Fig. 1.14.

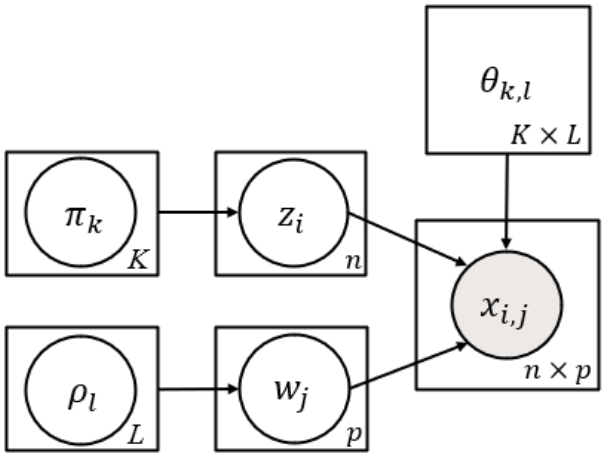


Fig. 1.14.: Latent Block Model graphical model

Inference

The LBM likelihood optimization is not straightforward, and requires specific algorithms [GN13]. As in the clustering case, the inference process is usually performed in an Expectation-Maximization fashion (c.f., Sect. 1.2.2). However, the EM algorithm would require to access the block membership posterior distribution $\mathrm{p}(\mathbf{z}, \mathbf{w} \mid X, \Theta)$, which is not tractable [GN13]. Alternative approaches include the variational inference (VEM), Stochastic-Gibbs EM (SEM) , and the Classification-EM (CEM) algorithm.

In the following we detail the SEM, that has the advantage to reduce both the risk of getting stuck in a local optima and the dependence to initialization.

SE step Considering fixed values of the parameter set Θ, the block membership posterior distribution $\mathrm{p}(\mathbf{z}, \mathbf{w} \mid X, \Theta)$ is approximated by a Gibbs sampler that alternate the sampling of $\mathbf{z}$ given $\mathbf{w}$, and $\mathbf{w}$ given $\mathbf{z}$. For a given row membership z_i update, the sampling is performed by drawing from the conditional distribution $\mathrm{p}(z_i \mid \mathbf{w}, \mathbf{x}_{i,\cdot})$:

$$\mathrm{p}(z_i = k \mid \mathbf{w}, \mathbf{x}_{i,\cdot}) = \frac{\mathrm{p}(z_i)\ \mathrm{p}(\mathbf{x}_{i,\cdot} \mid z_i = k, \mathbf{w})}{\mathrm{p}(\mathbf{x}_{i,\cdot})} = \frac{\pi_k\ \mathrm{p}(\mathbf{x}_{i,\cdot} \mid z_i = k, \mathbf{w})}{\sum_{k=1}^{K} \pi_h\ \mathrm{p}(\mathbf{x}_{i,\cdot} \mid z_i = h)},$$

where $\mathrm{p}(\mathbf{x}_{i,\cdot} \mid z_i = k) = \prod_{j=1}^{p} F_{k,w_j}(x_{i,j})$ is the distribution of the row $\mathbf{x}_{i,\cdot}$ in the block-clusters. The column membership sampling is performed symmetrically, according to the following conditional distribution

$$\mathrm{p}(w_j = l \mid \mathbf{z}, \mathbf{x}_{\cdot,j}) = \frac{\pi_k \, \mathrm{p}(\mathbf{x}_{\cdot,j} \mid z_i = k, \mathrm{z})}{\sum_{k=1}^{K} \pi_h \, \mathrm{p}(\mathbf{x}_{\cdot,j} \mid z_i = h)},$$

M step Considering fixed values of the block membership $(\mathbf{z}, \mathbf{w})$, the parameter set Θ update is, multivariate Gaussian case, given by:

$$\begin{aligned}
n_k = \sum_{i=1}^{n} \mathbb{1}_k(z_i), \; p_l = \sum_{j=1}^{p} \mathbb{1}_l(w_j), \; n_{k,l} = \sum_{x \in X_{k,l}} 1, \\
\pi_k = \frac{n_k}{n}, \; \rho_l = \frac{p_l}{p}, \; \mu_{k,l} = \frac{1}{n_{k,l}} \sum_{x \in X_{k,l}} x, \\
\Sigma_{k,l} = \frac{1}{n_{k,l}} \sum_{x \in X_{k,l}} (x - \mu_{k,l})(x - \mu_{k,l})^T.
\end{aligned}$$

The algorithm iterates these two steps for a given number of iteration or until membership convergence. This inference process is described in Algorithm 4.

Algorithm 4: SEM algorithm for Latent Block Model inference

Input : row-cluster number K, column-cluster L, iteration number M
Output : LBM parameters $\hat{\Theta}$, block partition $(\hat{\mathbf{z}}, \hat{\mathbf{w}})$
Initialize row-partition $\mathbf{z}$
Initialize column-partition $\mathbf{w}$
for $m \leftarrow 1$ **to** M **do**
  for $i \leftarrow 1$ **to** n **do**
    Estimate $\mathrm{p}(z_i = k \mid \mathbf{w}, \mathbf{x}_{i,\cdot})$ given Θ
    Sample membership z_i
  for $j \leftarrow 1$ **to** p **do**
    Estimate $\mathrm{p}(w_j = l \mid \mathbf{z}, \mathbf{x}_{\cdot,j})$ given Θ
    Sample membership w_j
  Update Θ given $(\mathbf{z}, \mathbf{w})$.

As in the BNP clustering case, the final block membership can be inferred with a consensus partition estimation [GOK18].

The model selection can be performed in several ways (see [Lom12; Ker+15; Bra14] reviews), usually with an exhaustive search associated to a model selection criterion. The BIC criterion has also been derivated [Lom12] for continuous data:

$$BIC_{LBM} = ln\ \mathrm{p}(\mathbf{x} \mid \Theta) - \frac{K-1}{2}\ ln(n) - \frac{L-1}{2}\ ln(p) - \frac{KLC}{2}\ ln(np),$$

with C is the number of parameter per block.

Functional Latent Block Model

As stated in Sect. 1.5.2, the LBM has been applied to various types of data. Recently, several applications to time series clustering have been developed. The first application [CB17] extends the clustering method [Cha+13] to coclustering, and assumes that the contents of a block can be represented by a piecewise polynomial regression model. This method supposes that the time series can be adequately modeled by piecewise representation, and the resulting segmentation can be a valuable information for the expert in that case. However, this method has the drawback to work in the original time series space, which becomes computationally prohibitive in high-dimension cases.

The other existing functional LBM [SAJ18; Bou+18; Sch+19] share the same time series preprocessing step, based on a fPCA representation that allows to work in a lower-dimensional space. One important difference between these methods is the modeling of the fPCA coefficient: while, in [SAJ18], the time series are represented by the first fPCA coefficient, the extended methods in [Bou+18; Sch+19] apply a subspace projection representation that associates a specific combination of fPCA coefficients to each block representation.

In [Bou+20], the authors apply the same principle to MTS (i.e., each cell $x_{i,j}$ is itself an MTS). Very recently, the method [Cas+21] has been developed, that models the block components in the original space with the *Shape Invariant Model*.

Specific model selection criteria have been developed for the Functional Latent Block Model, including an ICL-BIC extension [SAJ18] and the ICL criterion [Bou+18; Cas+21], based on the existing ICL developed for LBM [Lom12].

Non-Parametric Latent Block Model

As the DPMM can be seen as an extension of the MM with additional BNP prior, the Non-Parametric Latent Block Model can be seen as an extension of the LBM.

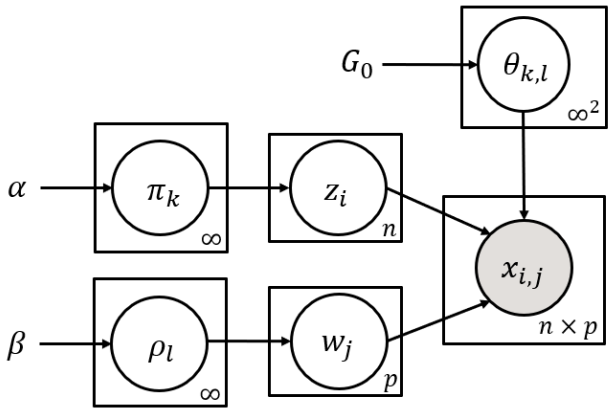


Fig. 1.15.: Non-Parametric Latent Block Model graphical model

For the univariate case, the BNP extension of the LBM has been developed and presented in [MR07], and consists in specifying two separate prior on the proportions, and a prior on the block component distribution. In [MR07], the authors use Pitman-Yor (PY) processes as proportion priors (equivalent to the Dirichlet prior with an additional discount parameter, and which tends to create clusters with uniform sizes). With Dirichlet priors (used for instance in [SB08; WF12]), the model is defined by:

$$
\begin{aligned}
x_{i,j} \mid z_i, w_j, \theta_{z_i,w_j} &\sim F(\theta_{z_i,w_j}),\\
z_i \mid \pi \sim Mult(\pi),\ w_j \mid \rho &\sim Mult(\rho),\\
\pi \mid \alpha \sim SB(\alpha),\ \rho \mid \beta &\sim SB(\beta),\\
\theta_{k,l} &\sim\ G_0,
\end{aligned}
$$

with SB the Stick-Breaking generation process. The associated graphical model is displayed in Fig. 1.15. As in the clustering case, the inference can be performed in different ways, including variational inference, split-merge, and symmetrical Gibbs-sampler. This last inference method can be seen as two separate Gibbs-sampler inference on the rows and on the columns, where the objective is to approximates the joint distribution of the column and row memberships. The update of a row-membership z_i given the other row-membership $\mathbf{z}_{-i}$ and the column-partition $\mathbf{w}$ consists in sampling according to:

$$
\begin{cases}
\dfrac{n_k}{n-1+\alpha}\ \mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}, X_{-i}, \mathbf{z}_{-i}, G_0), & \text{existing row-cluster } k, \quad (1.12)\\
\dfrac{\alpha}{n-1+\alpha}\ \mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}, G_0), & \text{new row-cluster,} \quad (1.13)
\end{cases}
$$

Where $\mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}, X_{-i}, \mathbf{z}_{-i}, G_0) = \prod_{j=1}^{p} F(x_{i,j}, \theta_{k,w_j})$ is the product of the block component densities, conditional to the fixed column-memberships values. In eq. (1.13), the joint prior predictive distribution of $\mathbf{x}_{i,\cdot}$ is $\mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}, G_0) = \prod_{l=1}^{L^{(m)}} \mathrm{p}(\mathbf{x}_{i,l} \mid G_0))$, with $L^{(m)}$ the current number of column-partition in $\mathbf{w}$ at iteration m, and $\mathbf{x}_{i,l} = \{x_{i,j} : w_j = l\}$ the elements of row $x_{i,\cdot}$ belonging to column-cluster l. With this notation, the joint prior predictive of $\mathbf{x}_{i,l}$ is the density of a multivariate t-distribution similar to the one used in Eq. (1.11).

The extension of the NPLBM to the case of time series coclustering is the topic of the contribution detailed Chapt. 4, which also details an adapted symmetrical Collapsed Gibbs Sampler.

Parametric or BNP versions of the LBM share one major assumption: there is only one row-partition, and, by construction, this row-partition is the same for every column-clusters. The next collection of methods relaxes this assumption and consider that every column-cluster can be associated with a specific row-partition.

1.5.3 Multi-Clustering

In a multivariate dataset, every variable taken independently is susceptible to produce a specific and distinct row-partition. The Multi-Clustering consists in assuming a small number of row-partitions are shared by the variables, which defines a column-partition. The associated structure is represented on Fig. 1.13-c).

It is different from clustering subspaces, which consists in estimating partition associated to specific linear variable combinations, in the way that Multi-Clustering estimates a variable partition such that variables in a column-cluster share the same row-partition. In some way, the Multi-Clustering can be considered a hard subspace clustering method, i.e., with hard column-cluster membership.

Parametric Multi-Clustering

The model-based Multi-Clustering has been developed by [GS07], which assumes a multi-partition block structure adapted to continuous multivariate datasets, where each cell is an univariate continuous observation. This method regroups the variable with a forward/backward search on the basis of a model selection criterion (in the example, variants of AIC and BIC), and models each column-cluster content with a full-covariance GMM. In this setting, the variables inside a block are considered dependent. The method has been extended in [GMS18], which relaxes the block

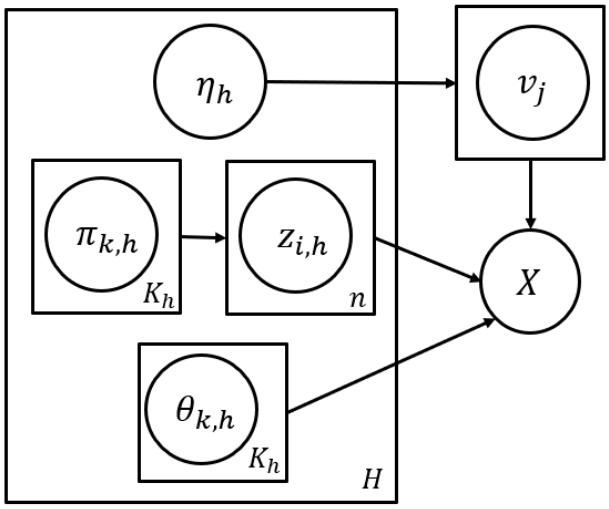


Fig. 1.16.: Multiple Partitions Model graphical model

independence assumption, and separates the variables in three group: classifying, redundant and non-classifying.

Recently, [MV19] proposed a the Multiple Partitions Model (MPM) that deals with heterogeneous data (including categorical), and which consider the variable independence inside the block (resulting, for the Gaussian case, in diagonal covariance matrices). This model can be described by:

$$x_{i,j} \mid \{v_j = h,\ z_i^h = k\} \sim F(\theta_k^h),$$
$$v_j \sim Mult(\eta),\ z_j^h \sim Mult(\pi_h),$$

where π_h are the row-cluster proportions corresponding to partition h, η are the column-cluster proportions and $\mathbf{v} = (v_j)_p$ are the column-cluster membership. Note that, in this model, the block component distribution is a $p_h \times d$-dimensional distribution and not d-dimensional, as each variable is modeled independently. The associated graphical model is displayed in Fig. 1.16. The authors propose two methods to perform simultaneously the model inference and the model selection based on an modified EM process [Gre90] that maximizes a penalized log-likelihood. However, the authors state that these methods may suffer from combinatorial issues when the number of column-cluster is high (more than 5), in which case they suggest to use a forward/backward search.

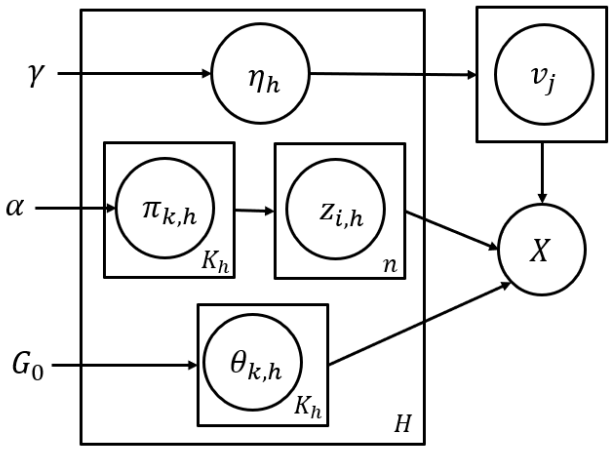


Fig. 1.17.: Non-Parametric Multi-Clustering graphical model

Bayesian Non-Parametric Multi-Clustering

In parallel to the parametric model-based methods, the BNP Multi-Clustering (BNPMC) has also been developed in two independent works, the Cross-Categorization model from [Man+09] and the Non-Parametric Multiple Clustering model from [Gua+10]. These two papers share the model definition, which consists in a hierarchical Dirichlet prior: first on the column partition, which allows to automatically infer the number of column-clusters, then one independent Dirichlet prior on the proportions of each row-partitions. The model, with graphical model displayed in Fig. 1.17, can be described as:

$$
\begin{aligned}
x_{i,j} \mid \{v_j = h, z_i^h = k, \theta_k^h\} &\sim F(\theta_k^h), \\
\theta_k^h \sim G_0,\ v_j \sim Mult(\eta),\ z_i^h &\sim Mult(\pi_h), \\
\eta_j(\mathbf{r}) = r_j \prod_{j'=1}^{j-1} \left(1 - r_{j'}\right), \quad r_j &\overset{\text{i.i.d.}}{\sim} \text{Beta}(1, \gamma), \\
\pi_j^h(\mathbf{t}^h) = t_j^h \prod_{j'=1}^{j-1} \left(1 - t_{j'}^h\right), \quad t_j^h &\overset{\text{i.i.d.}}{\sim} \text{Beta}(1, \alpha).
\end{aligned}
$$

The inference is performed with a variational inference approach in [Man+09], and a MCMC-based in [Gua+10].

The contributions detailed in Chapt. 3 and Chapt. 5 consists in two Multi-Clustering models: the first one addresses the problem of time series Multi-Clustering with a parametric conditional latent block model, the second contribution is a BNP model

that combines Multi-Clustering and Coclustering, and automatically infers the model selection without assumption on the partition dimensions.

1.6 Conclusion

Time series clustering is a large and open problem that has been addressed with an important number of approaches, based on dissimilarity measures, feature extraction methods, clustering algorithms, ...

The approach choice is made by the user, based on prior knowledge of the dataset, and leads to the creation of *natural* groups of observations, w.r.t. the clustering objective. This knowledge is expressed as assumptions on the information that the user seeks to use for the cluster creation, and that should be conserved and outlined in the clustering result (e.g., a specific feature, a shape, the presence of a shapelet, the dissimilarity in a lower dimensional space, ...). When several variables are present in the datasets, additional assumptions are needed to deal with the dimension increase and make use of the variable dependencies information.

In the following chapters, the proposed univariate time series clustering is based on a latent scenario hypothesis, and the multivariate approach is based on a block clustering structure.

2 Time Series Clustering with Model-Based Dictionary Representation

In this chapter, we present an univariate time series clustering method applied to time series produced by autonomous driving numerical simulations. The method is based on a driving pattern dictionary construction and consists in three steps: automatic segmentation of each time-series, regime dictionary construction, and clustering of produced categorical sequences. In this chapter's first section, we present the detailed simulation method and the time series structure. In the second part, we discuss the existing approaches and describe our contribution. In the third section, we present the results obtained on public datasets and on an industrial use case: the Autonomous Emergency Braking (AEB) system validation. Finally, we conclude on our method's capabilities and perspectives.

2.1 Regime-changing Time Series Clustering

AEB use case time series are the result of the chaining of distinct phases, also known as regimes. Provided the ability to detect those regimes, it is possible to use their estimated distribution (order, frequency, amplitude...) to characterize the observations and discriminate them. During the last decades, several papers have been proposed to detect optimal regime change points. Those methods sum up to piece-wise polynomial regression models. The common strategy relies on optimizing an approximation error in different ways: sliding windows of increasing size as in [Keo+04] and [Fuc+10], by dynamic programming as in [LM00], Hidden Markov models in [Keh04] or by regression mixture models in [Cha+09b]. We selected this last model for two reasons: on the one hand, the benefits of using a mixture model (confidence intervals, model selection strategy...) and on the other hand, the particular performances of this model compared to hidden Markov model approaches and its computational efficiency compared to dynamic programming methods [Cha+09b; Cha+10].

In $mixRHLP$ from [Sam+11], the same author combines the piece-wise regression model in a finite mixture to construct a one-step model-based clustering method. The proposed approach aims at regrouping time series with common regimes cut-points. $mixRHLP$ also assumes that the number of regimes is known. These assumptions are not suitable to our use case, in which time series belonging to the same class can exhibit varying regime duration and cut-points. Moreover, the number of segment is not known in advances, nor the polynomial regression order.

Our contribution is an attempt to adapt $mixRHLP$ to our constraints. It consists in a three-steps workflow with the addition of an original strategy of segmentation model selection. In the first step, we apply Individual time-series segmentation with a polynomial regression mixture. In the second step we build a standard dictionary of regimes by clustering the extracted segments. Finally, The clustering of these sequences using Levenshtein distance in categorical sequence space produces the final result. Our method, called $SDLHC$ for Segmentation, Dictionary construction, Levenshtein Hierarchical Clustering, has the following advantages:

- Clustering based on regime detection is intuitive and easily interpretable by experts.
- The method can be applied to a dataset of time-series with unequal lengths. Moreover, it is independent of the time-series synchronicity and the regime's moment of appearance synchronicity.
- The segmentation phase can be applied independently on each time-series, which makes the computation an embarrassingly parallel task. This step drastically reduces the data dimension.
- Our segmentation strategy optimizes automatically both the number of segments and polynomial regression on each segment, which allows to automatizes this step and makes the segmentation step a turn-key solution.

2.2 A three-step time-series clustering algorithm (SDLHC)

The method $SDLHC$ is composed of three steps: segmentation, dictionary construction, and categorical sequence clustering. The first two steps are addressed with a mixture model approach.

2.2.1 Segmenting time-series with a mixture of polynomial regressions

In a first step, each time series is individually segmented with a piecewise polynomial regression model with hidden logistic process from [Cha+09b]. This model is based on a polynomial regression model mixture, with time-dependent proportions following a hidden logistic process. Given a time-series $s = (s_t)_T$ and $\phi = (\phi_q(t) = t^q)_Q$ a functional basis of size Q. A Polynomial Regression Model (PRM) of sequence s in the basis ϕ is defined by

$$\tilde{x} = \sum_{q=1}^{Q} \beta_q \phi_q(t) + \sigma^2 \epsilon,$$

with $(\beta_q)_{q \in 1,\dots,Q} \in \mathbb{R}^Q$, $\sigma \in \mathbb{R}_*^+$ and $\epsilon \sim \mathcal{N}(0,1)$. These PRMs are the segmentation mixture model components.

Given a number of components K, the time-indexed vector $(z_t)_T$ designates the segment memberships. At any given time t, z_t follows a Multinomial distribution with parameters $\pi(t) = (\pi_k(t))_K$. The distribution of s_t is given by

$$p(s_t) = \sum_{k=1}^{K} \pi_k(t) f_{\theta_k}(s_t),$$

and the sequence s log-likelihood by

$$l(s;\theta) = \sum_{t=1}^{T} log \left(\sum_{k=1}^{K} \pi_k(t) f_{\theta_k}(s_t) \right), \tag{2.1}$$

with $f_{\theta_k}(s_t)$ the density associated to a PRM component. With this notation, the time-varying proportions $\pi_k(t)$ can be seen as the parameters of a Multinomial distribution followed by the clusters memberships at a given time t. These proportions vary according to a logistic process. More formally, for $k \in \{1,\dots,K\}$ and $t \in T$,

$$\pi_k(t) = p(z_t = k) = \frac{\exp(\sum_{s=1}^{S} w_{k,s} \phi_q(t))}{\sum_{h=1}^{K} \exp(\sum_{s=1}^{S} w_{h,s} \phi_q(t))}, \tag{2.2}$$

with $\mathbf{w}_k = (w_{k,q})_Q$ the associated model parameters. In the following paragraphs, we denote by W the set of parameters $(\mathbf{w}_k)_K$. The complete set of parameters is finally $\theta = (W, \beta, \sigma)$. The log-likelihood (2.1) optimization requires a specific version of the Expectation Maximization (EM) algorithm, with steps E and M described as follows.

Expectation step (E) Given the parameters θ, the first step of the EM algorithm consists in optimizing the complete log-likelihood defined as:

$$\begin{aligned}\mathbb{E}_{x,\theta}\left[l(x,z;\theta)\right] &= \mathbb{E}_{x,\theta}\left[\sum_{i=1}^{n}\sum_{k=1}^{K}\mathbb{I}_{z_i=k} log\left(p(x_i,z_i=k;\theta)\right)\right] \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K}\tau_{i,k} log\left(\pi_k f_{\theta_k}(x_i)\right).\end{aligned}$$

The development of the equation (2.3) shows that this step is simplified to the estimation of $\tau_{i,k} = p(z_i = k|x_i;\theta)$, the posterior distribution of $(z_t)_T$ conditionally to s. The Bayes theorem gives the following estimation of this quantity:

$$\begin{aligned}\tau_{t,k} = p(z_t = k|x_i;\theta) &= \frac{p(z_t=k,s_t;\theta)}{p(s_t)} \\ &= \frac{\pi_k f_{\theta_k}(s_t)}{\sum_{h=1}^{K}\pi_h f_{\theta_h}(s_t)}.\end{aligned}$$

Maximization step (M) At each iteration, the model parameters are updated during the Maximization step. In this phase, the following decomposition of the complete log-likelihood expectation is maximized:

$$\begin{aligned}\mathbb{E}_{x,\theta}\left[l(x,z;\theta)\right] &= \sum_{t=1}^{T}\sum_{k=1}^{K}\tau_{t,k}\ log\left(\pi_k f_{\theta_k,t}(s_t)\right) \\ &= \sum_{t=1}^{T}\sum_{k=1}^{K}\tau_{t,k}\ log\pi_k + \sum_{t=1}^{T}\sum_{k=1}^{K}\tau_{t,k}\ log f_{\theta_k,t}(s_t) \\ &= Q_1(\pi) + Q_2((\theta_k)_{k\in\{1,\dots,K\}}).\end{aligned}$$

with $\tau_{t,k} = p(z_t = k|s_t,\theta)$ the posterior segment membership distribution estimated in (2.3) during the expectation step, and $f_{\theta_k,t}$ the density associated to cluster k regression model at time t. This optimization can therefore be achieved by the separate maximization of Q_1 and Q_2. The optimization of Q_2 with respect to the parameters $\theta_k = (\beta_k,\sigma_k)$ consists in the estimation of a polynomial regression model on the points $(s_t)_T$ weighted by the posterior membership distribution $(\tau_{t,k})_T$, which gives the following expressions:

$$\tilde{\beta}_k = arg\min_{\beta_k} \sum_{t=1}^{T} \tau_{t,k}(s_t - \sum_{r=1}^{R} \beta_k \phi_r(t))^2, \tag{2.3}$$

$$\tilde{\sigma}_k^2 = \frac{1}{\sum_{t=1}^{T} \tau_{t,k}} \sum_{t=1}^{T} \tau_{t,k}(s_t - \tilde{\mu}_k(t))^2, \tag{2.4}$$

with $\tilde{\mu}_k(t) = \sum_{s=1}^{S} \tilde{\beta}_{k,s}\phi_q(t)$ the expected value of s_t in the regression model of component k.

The maximization of Q_1, which consists in estimating the time-dependent segment membership probabilities $(\pi_k(t))_K$, is performed with an Iterative Reweighted Least Squares (IRLS) algorithm [Cha+09a].

The algorithm has a linear complexity in the time series length and in the number of EM iteration, but quadratic in the polynomial regression order and cubic in the segment number. If this number is important (>15), we suggest to use a top-down hierarchical segmentation (not presented here) to reduce the effects of this cubic complexity.

2.2.2 Adaptive model selection strategy

In the initial model [Sam+11], the regression polynomial basis is common to every component, while in our contribution each regression order is specific. Moreover, we do not make *a priori* assumptions on the segment's number, which is also estimated by our strategy.

To estimate both the segment's number and the polynomial regression order on each segment, we combine this model with a new top-down strategy. This strategy is iterative and consists, at each step, in identifying the 'worst' component, in terms of the partial likelihood defined as:

$$l_k(x;\theta) = \frac{1}{\sum_{t=1}^{T} \pi_{t,k}} \sum_{t=1}^{T} \pi_{t,k} log\left(f_{\theta_k}(s_t)\right), k \in 1,..,K.$$

This criterion can be seen as the component representation quality weighted by the conditional membership probabilities. By improving the component $k_{old} \in \{1,\ldots,K\}$ that minimizes this score, two candidate models are created and compared. Splitting k_{old} in two sub-components, while conserving the other components, produce the first candidate model. We denote by k_1 and k_2 these new clusters.

We denote t_m the weighted median of the sequence $\{1, \ldots, T\}$ with weights $\pi_{k_{old}}$, and consider this time as the optimal cut-point for splitting the component $\pi_{k_{old}}$. The new components membership probabilities associated are based on the former component membership probabilities. The membership probabilities of component k_1 are defined by:

$$\pi_{k_1} = \begin{cases} \pi_{t,k_{old}} & ,t \in \{1, \ldots, t_m\} \\ \epsilon & ,t \in \{t_m + 1, \ldots, T\} \end{cases}, \tag{2.5}$$

with ϵ the threshold precision. The new cluster k_2 membership probabilities are obtained likewise, with inverted time indices. A regularization of the $(\pi_k)_K$ is necessary at this point to enforce the constraint $\sum_{k=1} \pi_{k,t} = 1, \forall t \in \{1, \ldots, T\}$.

The second candidate is obtained by increasing the polynomial regression associated to k_{old} by one. Two runs of EM are then launched, each of them considering one of the candidates as the initial state. After the convergence of both EM, the candidate optimizing the BIC is selected for the next iteration. This strategy is summarized in algorithm 5.

Algorithm 5: Top down segmentation strategy.

Fix the convergence threshold $c > 0$
Choose an initial state for the first EM run:
$\theta_{old}^{init} := ((w_k, \beta_k, \sigma_k^2)_{k \in \{1,\ldots,K\}})_{old}^{init}$
Compute π_{old}^{init} using equation (2.2)
Estimate θ_{old}^{end} by applying the EM algorithm
while *relative increment in BIC* $> c$ **do**
  Construct the first candidate model θ_{addSeg}^{init} with equation (2.5)
  Estimate θ_{addSeg}^{end} by applying the EM algorithm
  Construct the second candidate model θ_{incDeg}^{init} by increasing the least efficient component of the former mixture by one.
  Estimate θ_{incDeg}^{end} by applying the EM algorithm
  $\theta_{old}^{end} = \arg\max_{\theta \in \{\theta_{addSeg}^{end}, \theta_{incDeg}^{end}\}} BIC(\theta)$
end

After convergence of BIC criterion, the regime change cutpoint are estimated by post-processing the time-dependent memberships probabilities. This segmentation method is applied independently to each time-series and transforms each one in a set of sub-sequences.

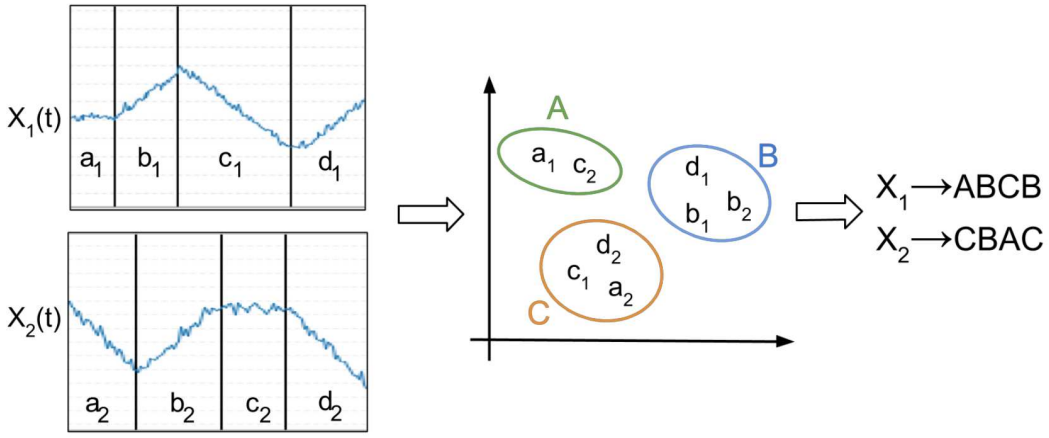


Fig. 2.1.: $SDLHC$ step 2: From time-series to categorical sequences.

2.2.3 Dictionary construction

Expressing the extracted segment in a common basis is mandatory to compare and cluster the sequences. The objective is to encode the original time-series in the new dictionary, as represented in the left part of Figure 2.1.

This common basis, or dictionary, is constructed with a clustering algorithm applied to the dataset composed of all segments from time series.

The sub-segments are first scaled, expressed on a common support, and regressed in a polynomial regression basis. Other informative descriptors can be added depending on the case, as the regime's duration, offset, or variance. These features are then clustered with a GMM to produce the dictionary. In section 2.2.1, we mentioned an implicit assumption based on the segmentation polynomial basis. In this section we make the additional implicit assumption that the GMM is adapted to the regimes density estimation and makes sense from the field expert point of view.

The EM algorithm is initialized with the k-means++ algorithm, which is a standard approach [BB13]. At the end of this step, the modes of the Gaussian mixture components are the reference regimes, entitled "patterns" in the following, with which to recode the original time series. The dictionary size is determined by the field experts, assisted by the BIC. The final dictionary is denoted $R = (r_u)_U$, with U the number of patterns.

After this re-coding phase, data dimension is greatly reduced: for a time-series of size n, the dimension goes from $\mathbb{R}^n$ to R^K, with R the categorical space and K the

number of regimes composing the sequences. The third and last step of $SDLHC$ regroups these sequences to produce the final clustering result.

2.2.4 Categorical Sequences Clustering

We use the Levenshtein distance [Lev66] combined with Ward's hierarchical clustering method to obtain the final clusters. Levenshtein distance between two categorical sequences a and b (respectively composed ofK_a and K_b segments) is defined as the minimum number of operations (insertion, deletion, substitution) needed to transform a into b. In this categorical space, Levenshtein Distance complexity is $O(K_a \times K_b)$.

However, with the original Levenshtein distance, replacing a symbol with another has a fixed unit cost, independent from the target and replacement symbols. Therefore, it does not take into account the possible similarity between patterns.

For instance, in Fig. 2.4, some speed patterns appear similar (different phases of acceleration of cruising speed) and distance on categorical sequences should take this similarity into account.

Weighted Levenshtein Distance

In order to take this information into account, we propose to use a Weighted Levenshtein Distance (WLD). Considering a set of patterns R, the weight on the edition cost between $r_1, r_2 \in R$ is symmetric and defined by:

$$\frac{||r_1 - r_2||_p}{\max_{r_a, r_b \in U} ||r_a - r_b||_p}, \tag{2.6}$$

where $||.||_p$ is the p-norm on the pattern space. The choice of the p-norm influences moderately the final clustering. Based on our practical experience in the AEB use case, we advocate the use of $||.||_\infty$.

Variant representations

The following paragraphs describe other possible representations and dissimilarities, that can be use in combination or in replacement of WLD.

Removing duplicate successive regimes The categorical sequence contains sometimes successive identical symbol (e.g., the sequence $abaac$). This effect can have several origins: it can be that the time series exhibits two close (but different) regimes, that are clustered together in the dictionary construction step. It can also be caused by an overestimation of the segment number in the independent segmentation step.

To mitigate these effects, we suggest to assume that the successive symbols do not add any information and to delete the redundant symbols (e.g., to transform $abaac$ into $abac$).

Integrating additional features in the Levenshtein Distance During the second phase of $SDLHC$, the dictionary has been constructed based on scaled and segments on same support, with the optional addition of the offset, variance and phase duration information. These characteristics can also be integrated in the weighting function of the Levenshtein Distance.

It is also possible to assume a hierarchy of these features, and consider a nested clustering (for instance, a partition based on pattern shape, then duration, then offset, ...).

Histogram-based representation A common representation choice in dictionary approaches consists in considering the recoded symbol histograms [LL09; Sch15]. This representation can also be used with our approach. However, as the number of patterns may be sensitive to the over-estimation of the segment number or of dictionary size, we suggest to also take into account the proximity between patterns in the comparison of these histograms. The final clustering can be obtained with the same hierarchical clustering approach based on the ℓ^2 norm.

Based on the same logic, the frequency or an absence/presence binary value representations can be considered instead of the histograms.

Integrating temporal relevance information In some situations, the user may be interested in comparing specific temporal sub-sequences of the time series. For instance, the last moments of the emergency braking scenarios are often more relevant. Instead of extracting the sub-sequences before $SDLHC$ run, it is possible to integrate this prior information in this last symbol sequence clustering step. For instance, by weighting the ℓ^2 norm in the histogram dissimilarity or in adding a temporal weighting in the edit cost of the Levenshtein Distance computation.

Tab. 2.1.: Parameters grid for ARI evaluation.

Method	Parameters	Range
SAX	Number of segments	(5,10,20,30,40,50)
	Number of gaussian bins	(2,3,5,7,10,20,30,40,50)
SDLHC	Dictionary size	(2,...,12)
MIXRHLP	Number of segments	(1,...,10)
	Polynomial regression order	(1,...,3)

Once the weighted Levenshtein Distance Matrix computed, Ward's hierarchical clustering method is applied to produce the final clusters.

In the following section we compare $SDLHC$ (with the WLD) to other state-of-the-art methods.

2.3 Experiments

We present, in this section, the results of several experiments on public datasets and on a real-world use case AEB obtained from Renault's simulation system. The method described in this article was implemented in Scala for the segmentation step and R for the hierarchical step. Code and (public) datasets available at https://tinyurl.com/sdlhc. The following baseline methods are selected:

- Three methods based on classical measures (Euclidean distance and DTW) associated with Partitional Around Medoid clustering approach. We have also tested the combination of DTW with center construction using the popular Dynamic Barycenter Averaging (DBA) method [PKG11]. These methods are denoted respectively ℓ^2, DTW_p, and DTW_d in the following.

- The $K-Shape$ method [PG15], a partitional clustering using the shape-based distance based on the cross-correlation measure.

- The SAX method, a dictionary-based methods from [Lin+03] that builds representations of the time series based on uniform time step segmentation. Based on this representation and associated distance, hierarchical clustering with Ward's criterion produce the clusters.

- In order to compare to the original method we aimed to extend, the results of $mixRHLP$ are also reproduced here.

Tab. 2.2.: Adjusted Rand Index on the UCR archive datasets.

Name	ℓ^2	DTW_p	DTW_d	KS	SAX	$MixRHLP$	$SDLHC$
CBF	0.28	0.66	0.68	0.63	0.46	0.47	0.71
OliveOil	0.46	0.53	0.40	0.50	0.00	0.40	0.55
Trace	0.32	0.40	0.66	0.57	0.32	0.41	0.94

Whenever needed, we interpolate time-series to equal-length sequences. We used the R package *TSclust*'s distance-based and SAX methods implementations and $mixRHLP$ using *flamingos* R package. Some of these methods depend on parameters, usually estimated by optimizing a risk in a supervised framework. The ARIs obtained here are always the maximal ARI obtained when testing the method on a parameter grid, displayed in Table 2.1, reproducing the results that experts can obtain after fine-tuning.

The comparison is based on the Adjusted Rand Index (ARI), a popular score in the clustering validation context. This criterion represents the proportion of correctly grouped and separated observations with respect to the observed classes.

2.3.1 Public datasets results

In order to validate $SDLHC$ adequation to the regime-changing time series clustering problematic, we selected a subset of the UCR archive [Dau+18] whose data exhibit regime structure. The ARI score obtained are shown in Figure 2.2. Although performant when applied to Renault's dataset (c.f. next subsection), we found that the weighted Levenshtein hierarchical clustering requires fine-tuning to adapt to the considered data characteristics. The test ran in this section therefore use the non-weighted Levenshtein distance.

The results confirm that the method perform well when addressing regime-changing time series. In these tests, the considered datasets contain equal-length time-series. However, $SDLHC$ can also be applied, without data preprocessing, to unequal-length time-series, which is the case in our application.

2.3.2 Real dataset results

In the following section, we evaluate the clustering performance of $SDLHC$ on an industrial use case: the Autonomous Emergency Braking (AEB) system validation. In

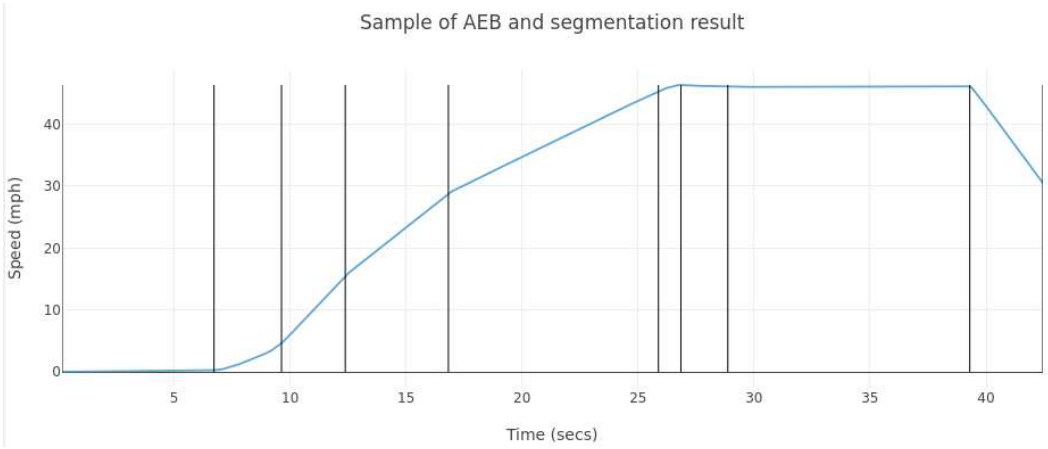


Fig. 2.2.: Segmentation result sample.

this case, a ground truth is available, and it is possible to compare clustering methods based on the similarity between the observed labels and the produced clusters. The clustering methods performances are, as in the previous section, measured by the ARI score. Renault's dataset is composed of 150 time series, with a duration varying from 13 to 52 seconds and length varying from 415 to 573 data points.

An illustration of the segmentation step result is shown in Figure 2.2.

The obtained dictionary, composed of five driving patterns, is shown in Figure 2.4. Using this dictionary, Figure 2.3 shows the encoded sequence.

Two stationary patterns can be recognized (b and c), corresponding to cruise speed phases, as well as two accelerating (d and e) and one decelerating (a).

The scores are obtained in the same conditions than the previous tests on public datasets, displayed in Figure 2.5.

Two versions of $SDLHC$ are tested: $SDLHC - LEV$ and $SDLHC - WLEV$ corresponding to the use of the standard and weighted Levenshtein distance in $SDLHC$'s last step. ARI criterion confirms that the $SDLHC - WLEV$ method slightly improves the score obtained by $SDLHC - LEV$. Among the distance-based methods, the $K - Shapes$ method is the best performer without, however, reaching the ARI threshold of 0.45 regardless of the number of clusters. With high cluster numbers, SAX method nearly reaches the performance of $SDLHC - LEV$. This result seems logical given the proximity between the proposed workflow and the dictionary-based methods.

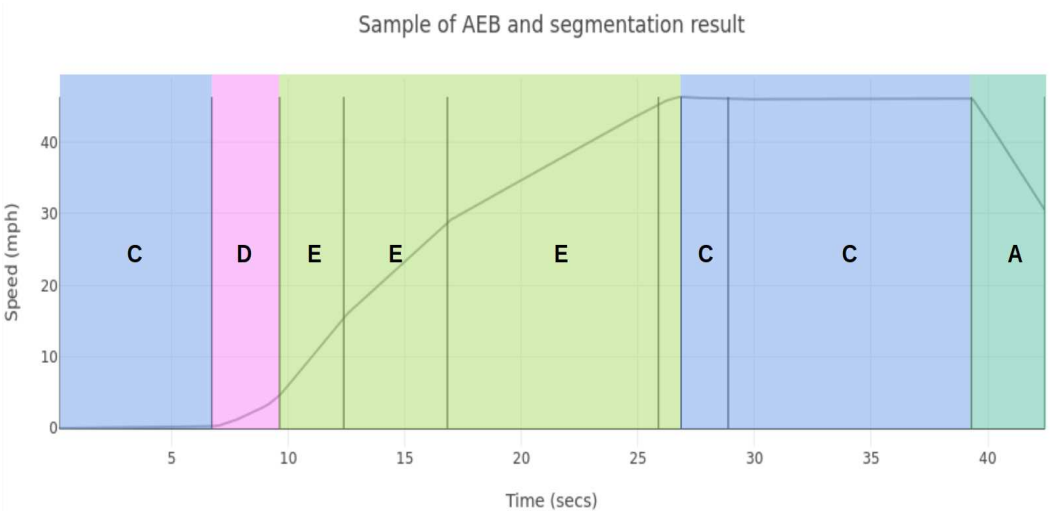


Fig. 2.3.: Segment sequence encoded using the dictionary. Two stationary patterns can be recognized (b and c), corresponding to cruise speed phases, as well as two accelerating (d and e) and one decelerating (a).

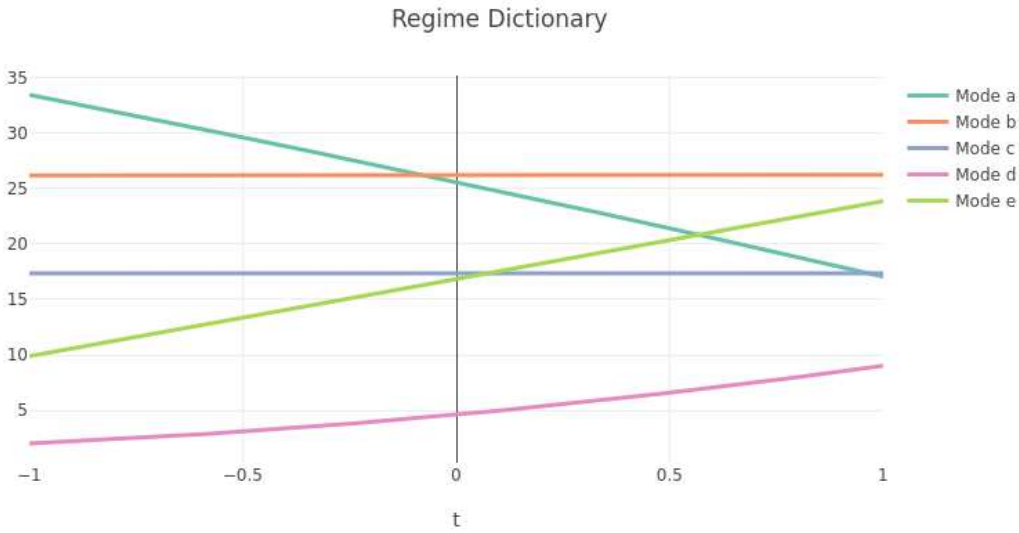


Fig. 2.4.: Dictionary produced in the AEB use case.

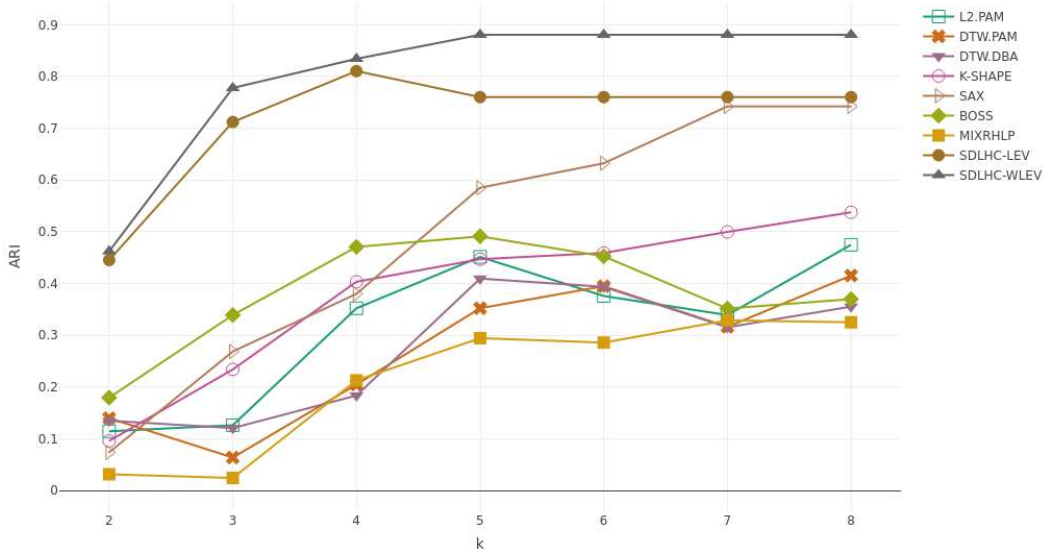


Fig. 2.5.: ARIs scores of various clustering approaches as a function of the number of clusters.

2.4 Conclusion

This first contribution $SDLHC$ performs an univariate time series clustering based on the independent segmentation of time series, a common segment pattern dictionary construction, and a categorical sequences clustering.

This method makes several assumptions: a) the time series are composed of several phases and these phases can be extracted with a piecewise polynomial regression ; b) A small number of pattern can adequately represent all the phases from every time series and these patterns can be obtained with a GMM on the segments representation coefficients; c) a relevant partition of driving behaviors can be obtained with a weighted levenshtein hierarchical clustering of the time series recoded as categorical sequences.

It is our strong belief that $SDLHC$ can be applied to other domains complying with the latent scenario hypothesis (e.g., human activity recognition).

One drawback of this contribution, however, is that it only addresses univariate time series dataset, while the ADAS validation use cases are mainly composed of multivariate data. Because the direct extension of $SDLHC$ is not straightforward (c.f. perspective discussions in Sect. 6.1), we considered a different strategy for these use cases. This strategy, the model-based block-clustering framework, is the topic of the next three chapters.

3 Multivariate Time Series Clustering With the Functional Conditional Latent Block Model

This chapter details the first block clustering method contribution of this thesis, which consists in a model-based parametric multi-clustering model.

3.1 Introduction

Coclustering methods infers simultaneously one row partition and one column partition (c.f. Sect. 1.5.1). However, this class of method makes one strong structural assumption: in addition of the variable partition, there is only one observation partition. This assumption means that the observation partition is shared among every column cluster. When the true latent block structure is composed of several row-partitions, inferring a coclustering forces the creation of a unique consensus row-partition made of the crossing of the multi-clustering row-partitions. This effect leads to the over-estimation of the consensus partition cluster number, and to the inappropriate split of block components. This effect is illustrated on Fig. 3.1, with three variable clusters, each containing containing 3 row-clusters.

When the multi-clustering structure contains an important number of variable clusters, this over-estimation can severely hinder the interpretation.

In the following we present a new model, called Functional Conditional Latent Block Model (*FunCLBM*), a multivariate time series method that extends the Functional Latent Block Model with a dependency structure that allows each column-cluster to be associated with a specific row-partition.

As opposed to existing model-based multi-clustering from [GS07] or [MV19], that handle univariate cells elements and where each variable follow a different distribution, in CLBM the cells belonging to the same block follow independently

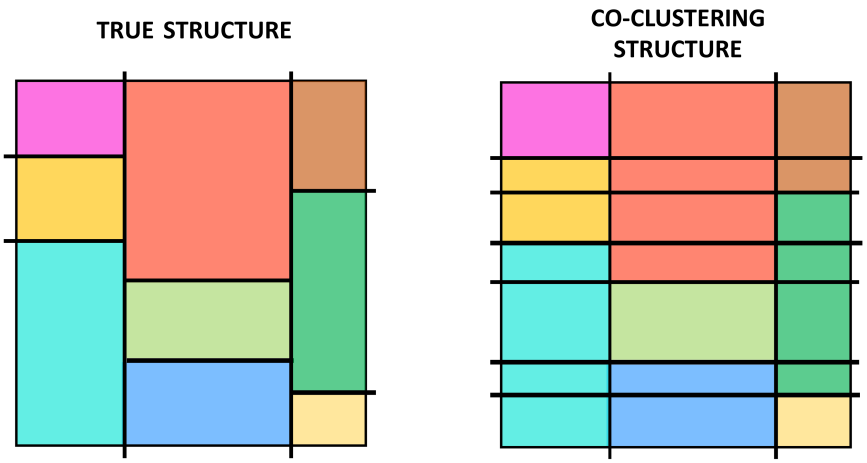


Fig. 3.1.: The true multi-clustering partition and a block partition that can be produced with a coclustering method. The coclustering inadequation produces more row-clusters than necessary and several blocks are wrongly splitted.

the same multivariate distribution, as is the case in the LBM [GN03; GN08] or FunLBM [SAJ18]. This difference is outlined in Fig. 3.2. This choice of model combines the efficiency of the LBM parameterization sparsity with the multi-clustering framework.

The output of this method is a dataset structure that highlights the variables that are linked together and the multiple partitions, from which it is easy to discard groups of uninformative (i.e., the one that are associated with a one-component row-partition) or uninteresting groups of variables.

As a model-based method, this approach allows to detect outlier observation from several point of view, and to precisely understand which group of variable is associated to the outlier values.

3.2 Functional Conditional Latent Block Models

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.

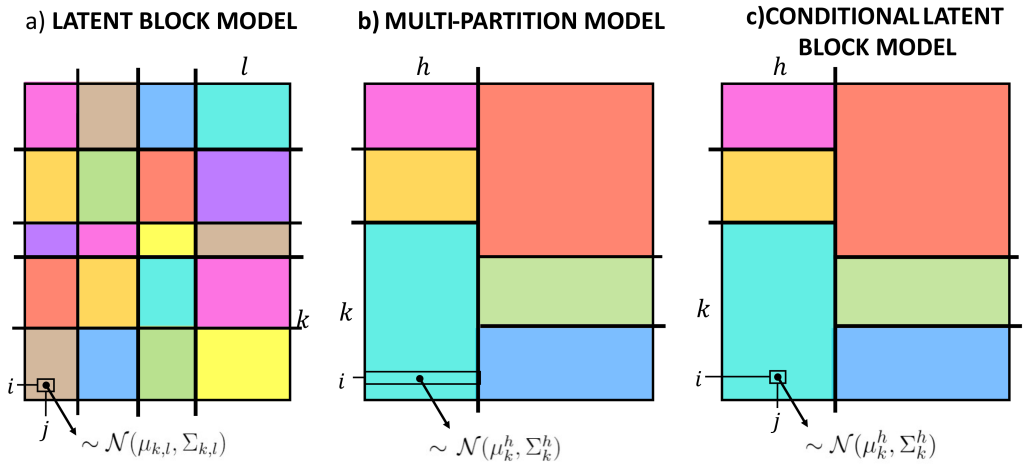


Fig. 3.2.: Differences between the LBM, MPM and CLBM. In CLMB c), every cells belonging to the same block follow the same block component distribution, as is the case in the LBM case a)

3.2.1 Representation with Principal Components of Interpolated Periodogram

As described in the Related Work section, there is a important number of existing time series representations (c.f., Sect. 1.1.4). In this chapter we chose to use an interpolated log-scaled Fourier periodogram representation, as it proves its ability to compactly represent the time series obtained in our driving simulation context.

Definition Each time series $S = (s_{i,j})_{n \times p}$ is first independently represented as a periodogram with a Fourier Transform. Given a time series $s_{i,j}$ with length T_i, this transformation outputs a decomposition of the time series in a weighted combination of trigonometric functions, with $f_{i,j}$ the associated frequency basis. Each frequency basis has the same dimension than the original time series, T_i, and the associated frequencies are different from one time series to the other (i.e., the time series are represented in different basis).

In order to compare the time series, it is mandatory to construct a common representation basis $\widehat{f}$, based on $(f_{i,j})_{n \times p}$.

A possible solution consists in computing the sample average gap $\widehat{\Delta f}$ between two consecutive frequencies over all time series, and to build an "average" frequency basis $\widehat{f} = \langle 0, \widehat{\Delta f}, 2\widehat{\Delta f}, \ldots, (\widehat{l} - 1)\widehat{\Delta f} \rangle$ that best represents the set of time series.

The periodogram values in this common basis can be estimated using linear or cubic interpolation techniques [CCP09]. After this step, as advocated in [CCP09], we take the logarithm transformation of the normalized periodograms, which eases the detection of the dependency structure and reduces the importance of the scale.

As a final transformation, the obtained log-normalized-interpolated periodograms are linearly projected with a PCA, which produces the dataset X, such that each cell $x_{i,j}$ contains the representation of time series $S_{i,j}$. Every cell has the same length d, corresponding to the number of PCA axis kept. The number of PCA axis, and the periodogram dimension are both hyper-parameter of these representations, that are studied in the next chapter.

This preprocessing method is a close variant of the fPCA used in [SAJ18; Bou+18], with a trigonometric basis of function and an additional log normalization transformation.

Interpretation As stated in this thesis introduction, it is important for the user to understand the underlying transformations that compose the clustering algorithms. Therefore, it is important to detail the characteristics of the produced representations, and the link between the similarity in the final representation space and the original time series similarity.

Because the time series are represented in a frequency basis, this transformation is suited to capture any seasonality information, but also the regime change similarity. Because the periodogram are normalized, this transformation also reduces the importance of the time series scale.

These elements can be observed in practice in the use cases: some variables grouped together may have different shapes, but they always exhibit an important correlation and contains synchronous evolution and events. For instance, a variable that records a braking system activation (which is a temporal binary variable exhibiting a rectangular shape) is often seen grouped with the braking strength.

3.2.2 Model definition

Let denote K_h the number of row-clusters associated to column-cluster h, $1 \leq h \leq H$. We also denote by $\mathbf{v} = (v_j)_p$ the column membership vector and η the column proportions. Given a column-cluster h, the associated row-clusters partition is denoted as $\mathbf{z}^h = (z_i^h)_{n \times K_h}$. We denote z_i^h the row-cluster membership of observation

i in the column cluster h. The complete set of row-partitions is denoted as $Z = (z_h)_H$. The row proportions are denoted $\pi = (\pi^h)_H$, with $\pi^h = (\pi^h_k)_{0 \leq k \leq K_h}$. Finally, the joint model density can be described by:

$$\begin{aligned} \mathrm{p}(X) &= \sum_{\mathcal{V}} \mathrm{p}(\mathbf{v})\mathrm{p}(x \mid \mathbf{v}) \\ &= \sum_{\mathcal{V}} \prod_j \eta_{v_j} \prod_H \mathrm{p}(X^h), \end{aligned}$$

where $\mathcal{V}$ is the set of all possible column-partition in H clusters, and $\mathrm{p}(X_h)$ is the density of the sub-matrix containing only the variable belonging to column-cluster h. This density can, in turn, be expressed as:

$$\begin{aligned} \mathrm{p}(X^h) &= \sum_{\mathcal{Z}_h} \mathrm{p}(\mathbf{z}_h)\mathrm{p}(X^h \mid \mathbf{z}_h) \\ &= \sum_{\mathcal{Z}_h} \prod_i \eta_{z^h_j} \prod_k \prod_{x \in X^h_k} \mathrm{p}(x \mid v_j = h, z^h_i = k), \end{aligned}$$

where $\mathcal{Z}_h$ is the set of all possible row-partition in K_h clusters, and $\mathrm{p}(x \mid v_j = h, z^h_i = k) = f(x, \theta^h_k)$ is the block-component density, in our case a d-dimensional multivariate Gaussian density with parameter $\theta^h_k = (\mu^h_k, \Sigma^h_k)$. The complete set of parameter θ=$(\eta, \pi, (\theta^h_k)_{K_h \times H})$ is inferred with a dedicated SEM-Gibbs algorithm.

3.2.3 Inference with SEM-Gibbs algorithm

As detailed in the related work chapter (c.f. Sect 1.2), the SEM algorithm is popular practice in the model-based clustering framework. As is the case for the coclustering, the direct optimization of the likelihood is not straightforward, and requires specific strategies.

In the following we propose an SEM-Gibbs algorithm that alternates the Gibbs sampling of the proportions and the parameter inference. Starting from an initial parameter state θ^0 and an initial column partition w^0, the algorithm alternates between these two steps:

1. SE step:

- Update the column-partition $\mathbf{v}$ given the row-partitions. For each column j, the update of the column membership v_j consists in sampling from $\mathrm{p}(v_j = h \mid \mathbf{x}_{.,j}, \theta, Z) \propto$

$$\frac{\eta_h \mathrm{p}\left(x_{.,j} \mid v_j = h, \mathbf{z}^h; \theta\right)}{\sum_{r=1}^{H} \eta_r \mathrm{p}\left(x_{.,j} \mid v_j = r, \mathbf{z}^r; \theta\right)},$$

  where $\mathrm{p}\left(x_{.,j}|z^h;\theta\right)$ is the density of column j in cluster-column h given the row-cluster memberships, i.e., $\mathrm{p}\left(x_{.,j}|z^h;\theta\right) = \prod_{i=1}^{n} f(x_{i,j}, \theta^h_{z_i^h})$.

- Update the row-partitions Z given the column-partition. For each column-cluster h and each row i, the row-membership $\mathbf{z}_i^h$ update consists in sampling a new value from $\mathrm{p}(\mathbf{z}_i^h = k \mid \mathbf{x}_{i,.}, \mathbf{v}, \theta) \propto$

$$\mathrm{p}\left(x_{i,h} \mid z_i^h = k, \theta)\right) = \frac{\pi_k^h \mathrm{p}\left(x_{i,h} \mid z_i^h = k, \theta\right)}{\sum_{s=1}^{H} \pi_i^s \mathrm{p}\left(x_{i,h} \mid z_i^h = k, \theta\right)},$$

  where $x_{i,h}$ designates the content of row $x_{i,.}$ restricted to column-cluster h, and $\mathrm{p}\left(x_{i,h} \mid z_i^h = k, \theta\right) = \prod_{j:v_j=h} f(x_{i,j}, \theta_k^h)$ is the density of the row i in block-cluster (k, h).

2. M Step: given the sampled block partition, and denoting by X_k^h the observations belonging to block (k, h), the mixture proportions are updated by:

$$\pi_k^h = \frac{1}{n}\sum_i \mathbb{1}_k(z_i^h),\ \eta_h = \frac{1}{p}\sum_j \mathbb{1}_h(v_j),$$

$$\mu_k^h = \frac{1}{n_k^h}\sum_{i,j} z_{ik}^h w_{jl} v_{ij},\ \Sigma_k^h = \frac{1}{n_k^h}\sum_{i,j} z_{ik}^h w_{jl}\left(v_{ij} - \mu_k^h\right)\left(v_{ij} - \mu_k^h\right)^T$$

The SEM algorithm alternates these two steps for a given number of iterations, and output a set of sample $(\hat{v}, \hat{Z})$.

As in the mixture model and coclustering case (c.f. Sect. 1.2.2, the initialization choice is crucial to ensure the good behavior of the algorithm.

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 sect. 3.3, these different initializations are experimented.

column component	row-cluster number					
1	1	1	1	2	2	3
2	1	2	3	2	3	3

Tab. 3.1.: Every model combinations for $H = 2$ and $K_m = 3$, where each row-cluster in each column component.

3.2.4 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 H and a set of row clusters numbers $K = (K_h)_H$. 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) [BCG00]. Initially developed for GMM Selection, extended by [Lom12] to coclustering and in [Bou+18] to functional coclustering, we propose the following extension to functional Multi-clustering:

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

where ν_k^h is the component parameter number of block (k, h). 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 coclustering case, finding the best structure can be done by an exhaustive grid search. This strategy cannot be applied in the Multi-Clustering case, where the number of possible model is huge, even for a small number of blocks.

With a fixed number of column-clusters H, the number of possible model with a maximum of K_m row-clusters is the number of possible combinations of H elements taken from a collection of K_m objects, with repetition (because several column cluster can share the same number of row-cluster -e.g., $(2 \times 2 \times 2)$ is a valid model choice-), and without order (because the model likelihood is invariant to label switching -e.g., choosing the model dimensions $(1 \times 2 \times 3)$ is equivalent to choosing $(3 \times 2 \times 1)$-). The exhaustive combination list for $H = 2$ and $K_m = 3$ is given in Table 3.1.

H_m \ K_m	2	3	4	5	6	7	8	9	10
2	5	9	14	20	27	35	44	54	65
3	9	19	34	55	83	119	164	219	285
4	14	34	69	125	209	329	494	714	1000
5	20	55	125	251	461	791	1286	2001	3002
6	27	83	209	461	923	1715	3002	5004	8007
7	35	119	329	791	1715	3431	6434	11439	19447
8	44	164	494	1286	3002	6434	12869	24309	43757
9	54	219	714	2001	5004	11439	24309	48619	92377
10	65	285	1000	3002	8007	19447	43757	92377	184755

Tab. 3.2.: Number of possible models with respect to K_m and H_m

The number of possible combinations is the number of multisets $\left(\binom{K_m}{h}\right) = \binom{K_m+h-1}{h}$. For a maximum number of column cluster H_m and a maximum number of row-cluster K_m, the total number of model is therefore given by:

$$\sum_{h=1}^{H_m} \left(\binom{K_m}{h}\right) = \sum_{h=1}^{H_m} \binom{K_m + h - 1}{h}.$$

This number grows quite fast, and becomes huge for low values of H_m and K_m (e.g., for $H_m = K_m = 10$, it amounts to 184755 combinations, and, for $H_m = K_m = 20$, $1.37E12$ combinations). Estimations of this number are given in Table 3.2 for a range of low values.

As a consequence of this important number of models, it is not possible to test the combinations exhaustively, and alternative model selection strategies must be considered. Our proposal of heuristics consist in estimating the column-partition with a coclustering model selection, and then to estimate independently the best model for each column-cluster.

In the next section, two of these approaches are tested on a simulated dataset, based on an LBM greedy search.

3.3 Experiments on Synthetic Data

In order to test the capabilities of FunCLBM, several experiments are first conducted on a simulated dataset. These experiments help us verify that the model is suited to the use case and that the SEM-Gibbs algorithm behaves as expected in a controlled

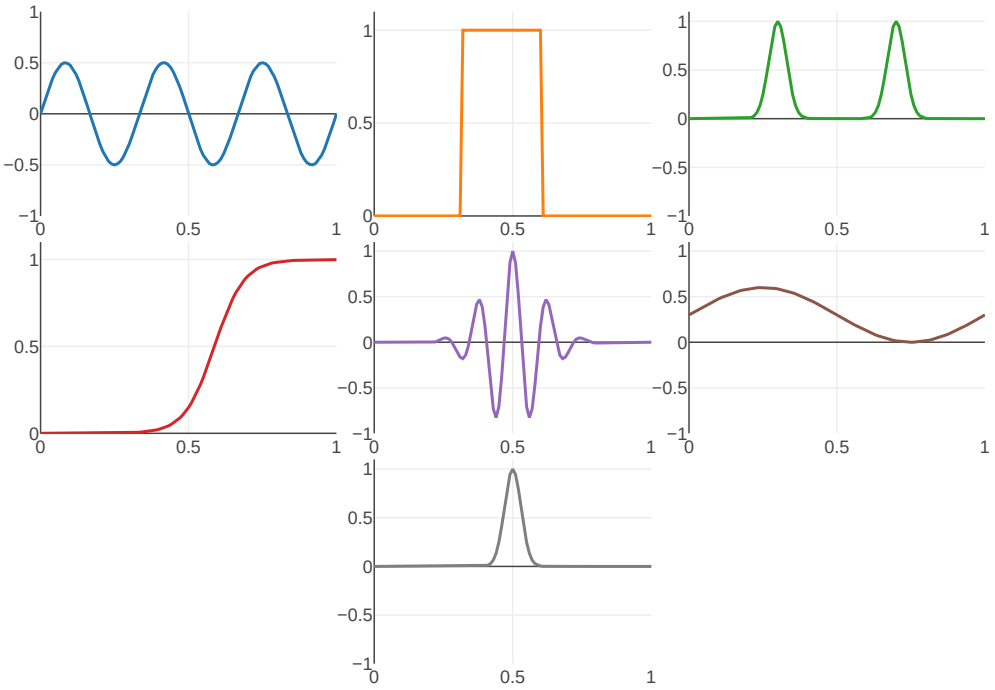


Fig. 3.3.: Prototypes used as block mode for the simulations

environment. The Scala source code is available at https://tinyurl.com/FunCLBM, along with the data simulation script.

3.3.1 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^h) and which represents the components modes in the original space. For each block (k, h), several time series are drawn following $\mathcal{N}\left(\phi_{kh}(t + t_s), s^2\right)$ with $s = 0.02$ and t_s a random shift $\sim \mathcal{N}\left(0, s^2\right)$. These modes are depicted in Fig. 3.3 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

Fig. 3.4.: ARI versus Log-Likelihood in 100 launches of SEM-Gibbs on the simulated dataset

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 $(40, 50)$.

3.3.2 Model Adequacy

As a preliminary test, we first verify the ability of FunCLBM to regroup time series after the pre-processing step and with the latent block structure hypothesis. To do so, we compare the model Log-likelihood produced after 100 launches of SEM-Gibbs inference to the corresponding ARI scores. The results, shown in Fig. 3.4, seem to indicate a correlation between ARI and likelihood, which tends to confirm the model suitability.

We verify this relationship with the Pearson's correlation coefficient between the two scores, and Kendall's correlation test. We use this latter test on scores ranks 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).

	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

Tab. 3.3.: Kendall's correlation test p-value (confidence level: 0.95)

The results (with 95 % confidence level for the Kendall test) are displayed in Table 3.3. For this dataset, the the suitability of the method seems attested by the strong Pearson's correlation for every partition dimension and by Kendall's correlation test p-value at 95% confidence level.

3.3.3 Initialization

The next experiments aim at evaluating the different initialization strategies. The clustering quality is again evaluated with the ARI criterion after 30 runs of SEM-Gibbs. We compare four strategies:

- Populate blocks with samples: the model is initialized by sampling a small number of variables for each column-cluster, and then a small number of rows for row-clusters.
- Random shuffle of the column partition, then of the row partition.
- One k-means algorithm run on the variables (i.e., on the transposed dataset), then one k-means to estimate each row-partition.
- Initialize $\mathbf{w}$ with the column-partition obtained after an LBM run, then the row-partitions at random.

The results distributions are described in Fig. 3.5.

The figure shows important differences between row and column cluster results, as expected since the model does not treat rows and column symmetrically. The k-means algorithm has an unexpected behavior: while performing well on average, the results show a high dispersion for column cluster ARI. Its row cluster ARI however is slightly better than the other methods.

The FunLBM initialization strategy gives the best results on average, closely followed by the randomPartition strategy. For real-life applications, we suggest to use FunLBM when the datasets dimensions allows it, and otherwise the randomPartition strategy, which is cheaper alternative.

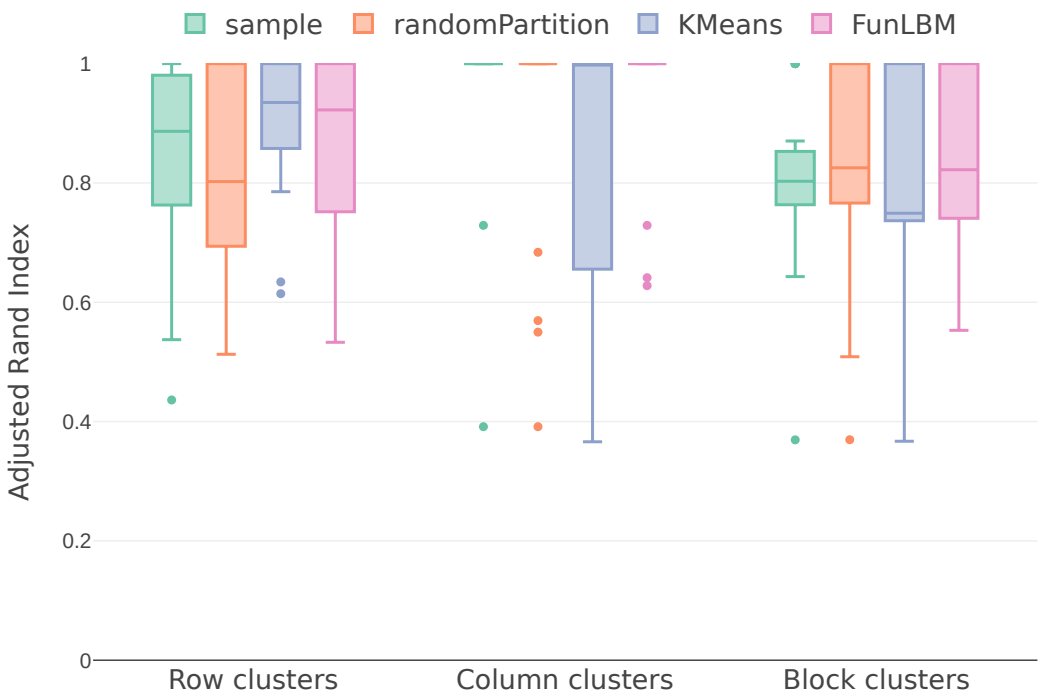


Fig. 3.5.: Distribution (median and quantiles 0.1,0.9) of Row, Cluster and Block partition ARI obtained after 30 SEM-Gibbs runs with different initialization methods.

Whether with the FunLBM or randomPartition initialization, an additional strategy can be to select the best result (w.r.t. model likelihood) among several runs. The following experiment highlights the interest of this approach, by comparing the ARI score increase with respect to the number of candidate runs. The results are displayed in Fig. 3.6.

The results shows that, whichever initialization strategy is applied, the concurrent run of several methods allows to stabilize the results, and finding the perfect structure is an easy task whenever the number of concurrent launches is higher than 4.

3.3.4 Model selection

The previous experiments compared the initialization methods for a given choice of model. In the following, we experiment two approaches of model selection, which is a challenging aspect of this method, and an important feature for the field expert. In [Bou+18], the coclustering model selection is performed with a grid search strategy, which requires inferring $K \times H$ models. As stated in Sect. 3.2.4, this exhaustive

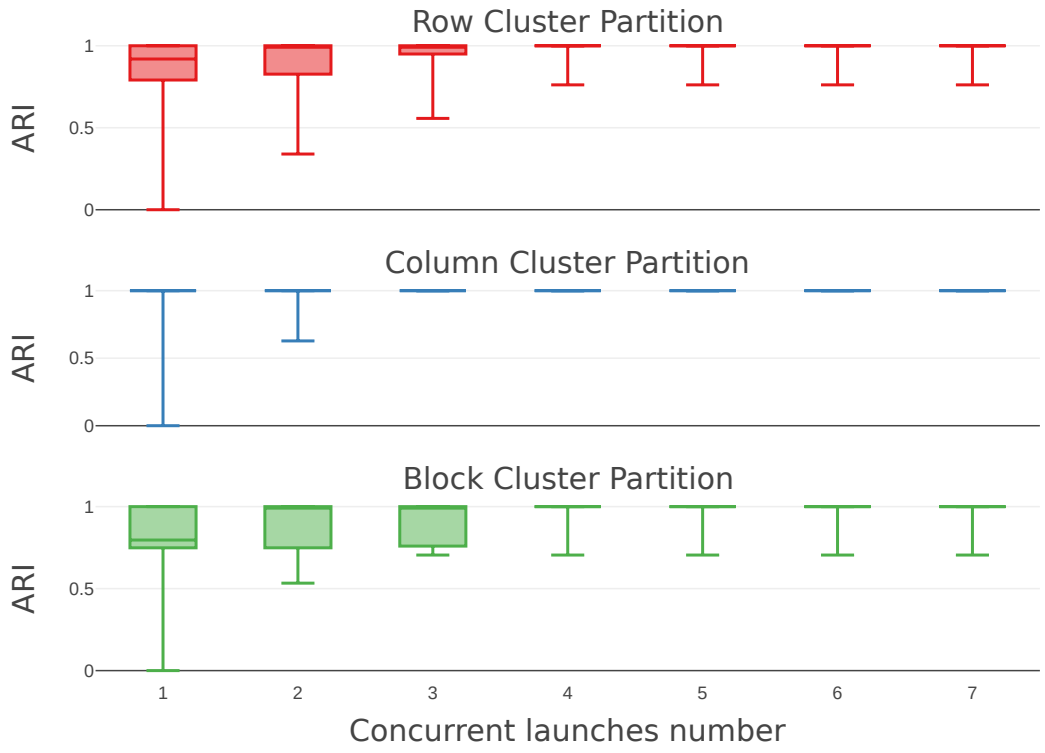


Fig. 3.6.: Best ARI obtained among several SEM-Gibbs runs (median, quantile 0.9; with variable number of concurrents)

search is not feasible in the multi-clustering case, because of the prohibitively high number of model possibility.

To overcome this limitation, we propose and compare two strategies that both avoids a direct grid-search of every multi-clustering combinations. These two strategies are based on a different estimation of the column-cluster number $\widehat{H}$ and then on a column-wise grid search.

In the first case, $\hat{\mathbf{w}}$ and $\widehat{H}$ are estimated from an LBM exhaustive grid search, which implies an exhaustive search of $K_M \times H_M$ LBM models and then $\widehat{H} \times K_M$ to estimate the best number of row-clusters per column-cluster. The total number of model to infer is bounded by $2K_mH_m$ (K_mH_m for the LBM grid search, then K_m possible models for each of the selected column number bounded by H_m).

The second strategy is an iterative algorithm that chooses, at each iteration, the best functional Latent Block Model between the one with an added row cluster and the one with an added column cluster, based on the ICL criterion [Lom12; Bou+18]. The search stops when the ICL stops increasing. In this LBM search, the number of model to infer is bounded by $K_m + H_m$, and the total model number is $K_mH_m + K_m + H_m$.

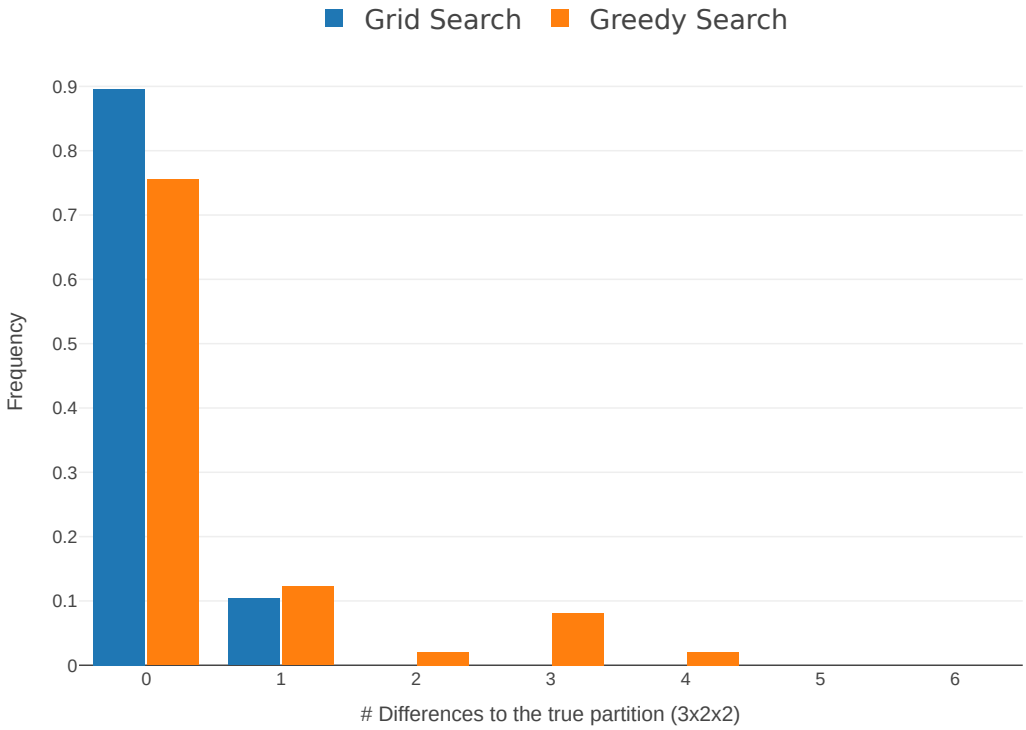


Fig. 3.7.: Results of 50 runs of each selection model strategy, in terms of differences to the generative model structure.

Each methods are run 50 times, and the model selection is evaluated with the model dimension difference to the ground-truth model (in this case, a $(3 \times 2 \times 2)$ model). These differences, displayed in Fig. 3.7, show that the LBM grid search performs well on that case, and is a valid strategy for low model dimensions candidates.

3.4 Application to autonomous driving system validation

3.4.1 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. 3.8.

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.

3.4.2 Results

The experiments on simulated datasets lead us to choose the following setup for the real case analysis: the 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.

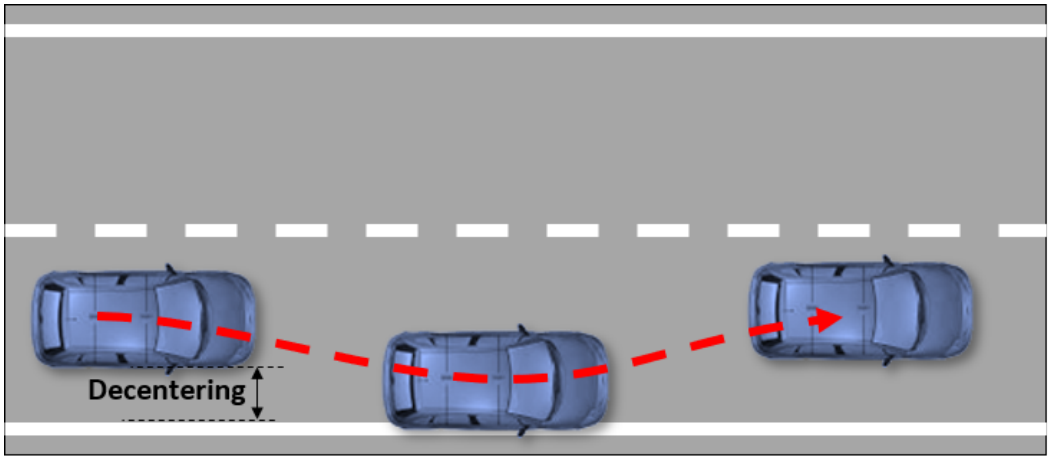


Fig. 3.8.: Use case illustration: Ego drifts from its lane and cross the white line on the side of the road, before being put back in the runway center

The final clustering structure is presented in Fig.3.9. It consists of 4 column clusters, each one with a different number of row clusters: $(6 \times 5 \times 4 \times 4)$.

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. 3.10, 3.11. The clusters adequately gather simulations that share the same behavior. In Fig. 3.10 case, the control logic is activated and the car is recentered in its lane, and then repeatedly

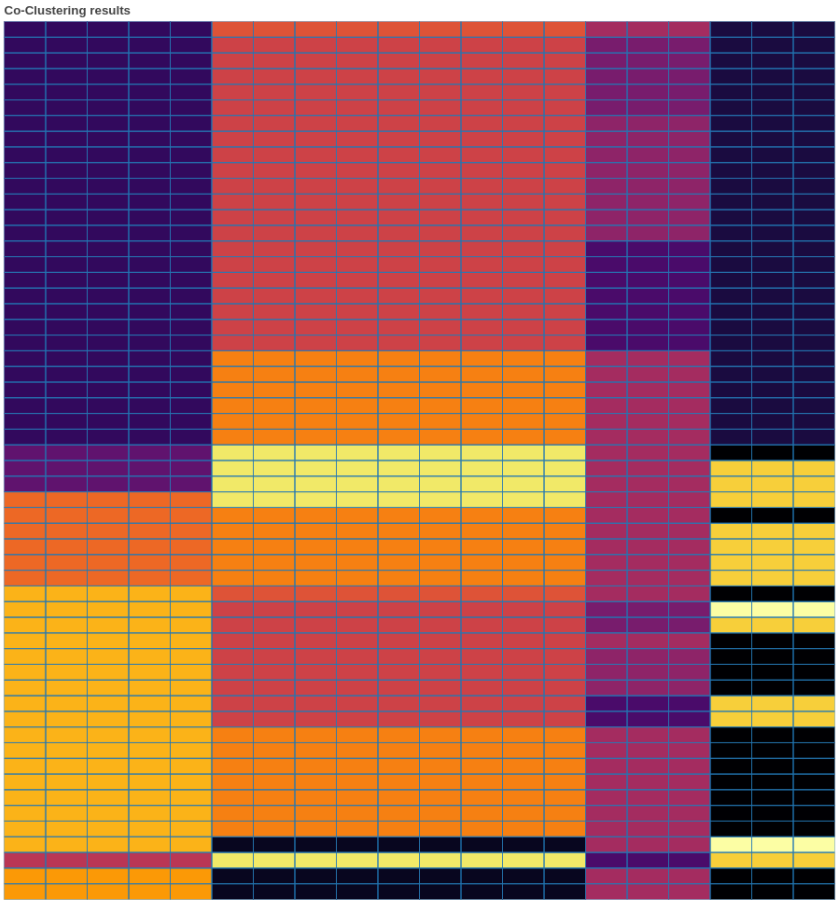


Fig. 3.9.: Final structure obtained on real case dataset

bounces back on the exact same road markings. In Fig. Fig 3.12 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 3.12 scenario, the car bounces only once and either goes to other side of the road of comes back after a large drift.

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. 3.14 and Fig. 3.15 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.3.13 shows the content of subcluster (3, 3) which illustrates

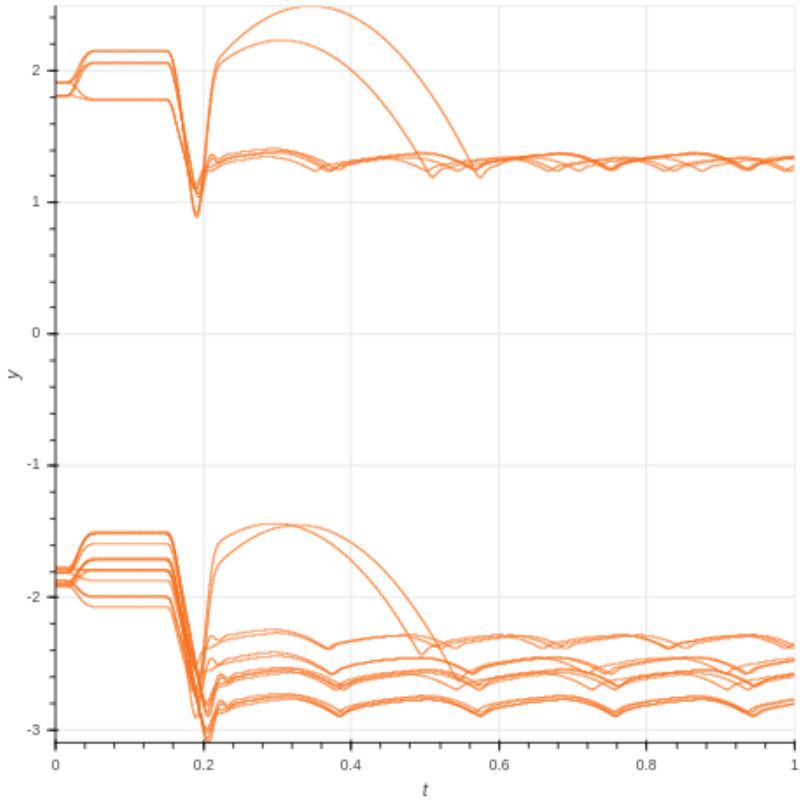


Fig. 3.10.: Ego lateral position in Block Cluster (5,1)

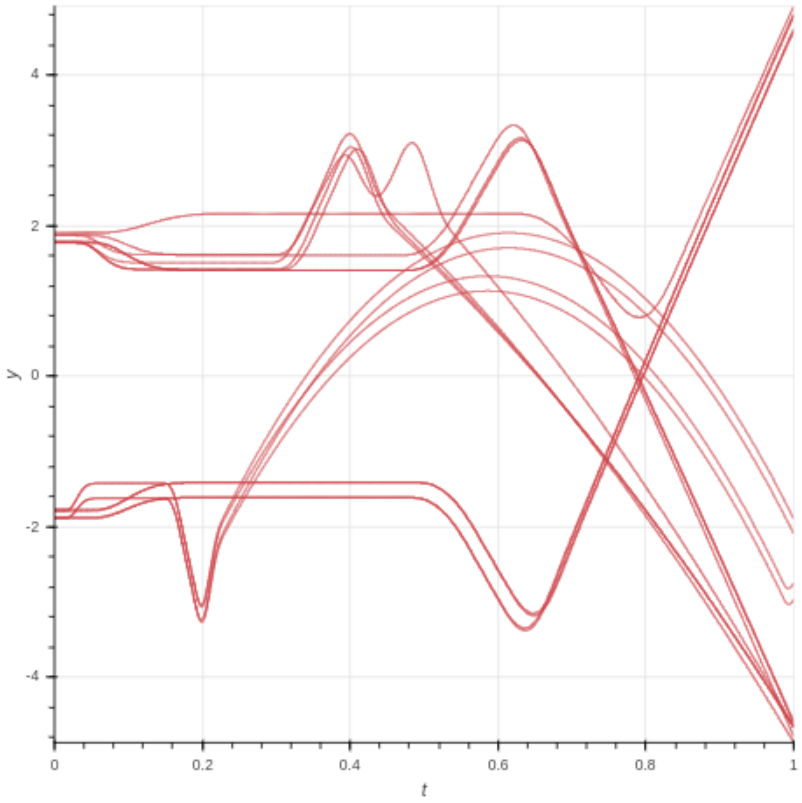


Fig. 3.11.: Ego lateral position in Block Cluster (2,1)

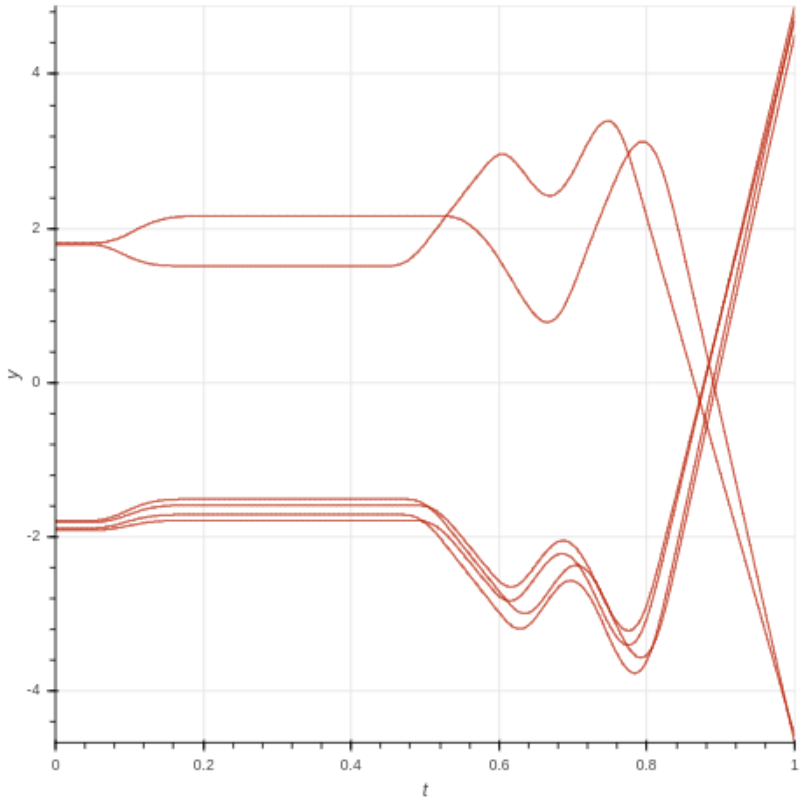


Fig. 3.12.: Ego lateral position in Block Cluster (3,1)

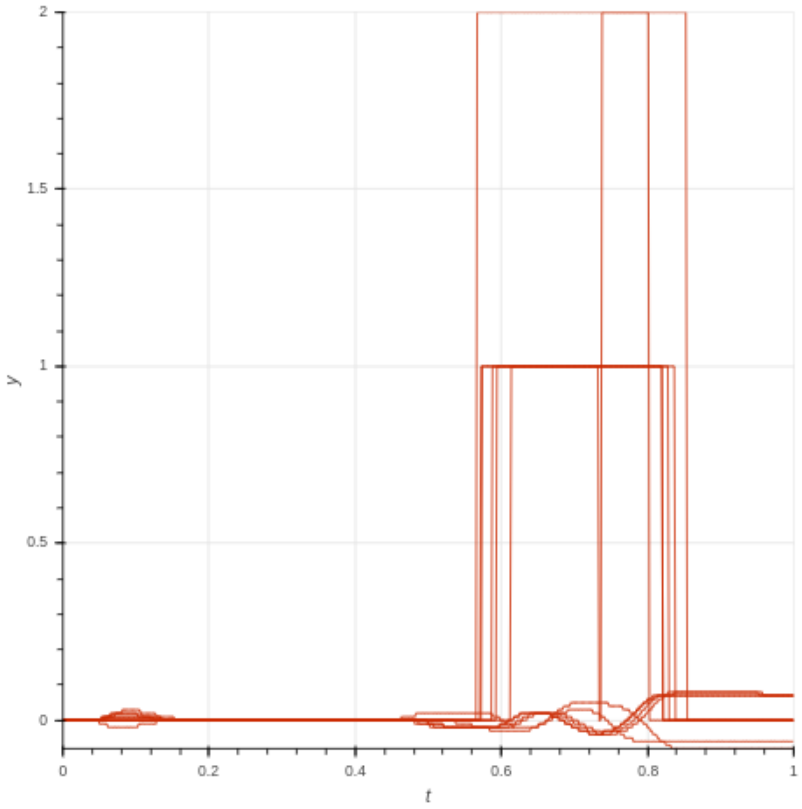


Fig. 3.13.: Control logic activation and changes in Ego's heading in Block Cluster (3,3)

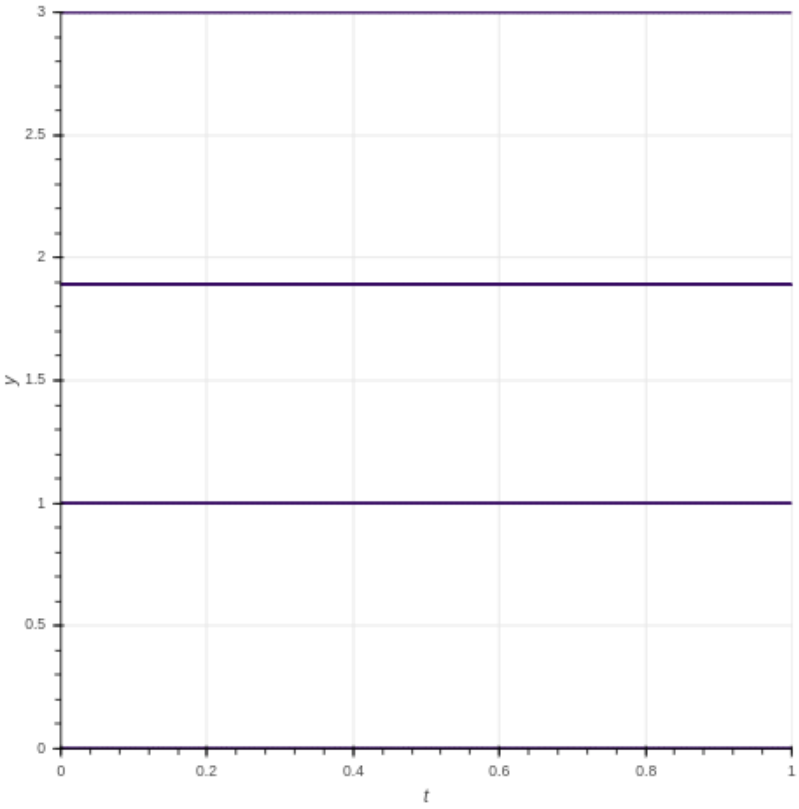


Fig. 3.14.: Uninformative signals in Block Cluster (1,1): linearly increasing feature (vehicle's width, length, headlights activation..)

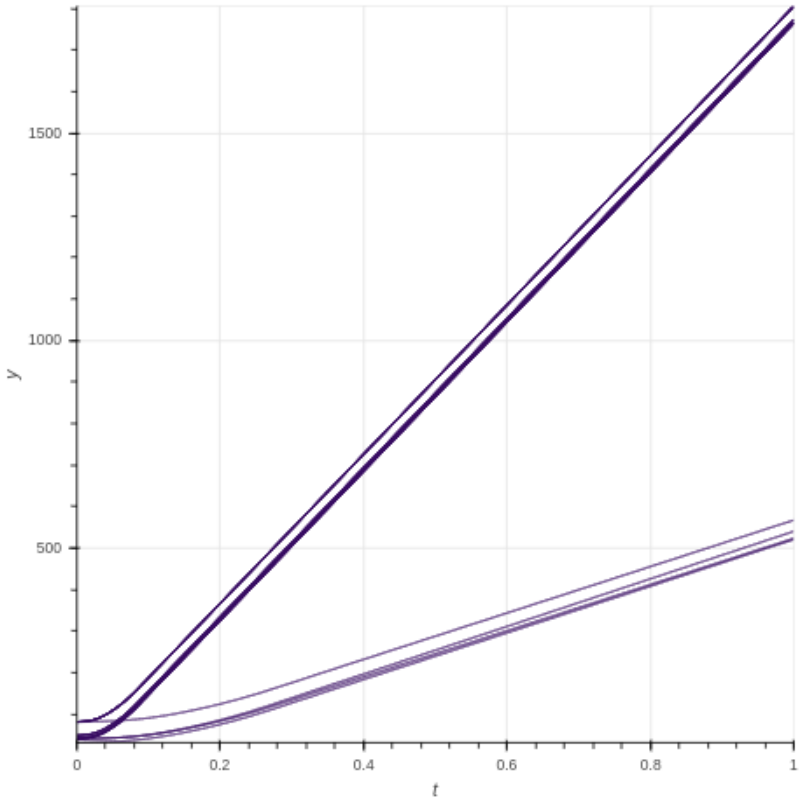


Fig. 3.15.: Uninformative signals in Block Cluster (1,1): linearly increasing feature (distance to origin)

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 observation clusters are relevantly regrouped and provide insights of the dataset content.

3.5 Conclusion

This chapter describes FunCLBM, a model-based method that 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.

Several initialization methods and model selection strategies are proposed and experimented on a simulated dataset. The experiments show 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.

A major drawback of this method is the model selection aspect, that cannot be performed without the help of specific strategies. The proposed greedy search algorithms allow to explore a part of the model space but at the cost of additional structural assumptions.

For instance, the LBM grid search-based model selection assumes that the LBM with the ICL can adequately estimate the true number of cluster column (which, by assumption, have different row-partitions).

In addition, for higher numbers of clusters (e.g., $H_m, K_m >= 20$), these strategies still involves the inference of hundreds of models.

One solution to circumvent this model selection limitation is to use the BNP framework. By adding a BNP prior to the existing block structure, it is possible to estimate simultaneously the model dimensions and parameters.

4 Bayesian Nonparametric Coclustering

This chapter describes a Bayesian Non Parametric Functional Latent Block Model, which is probabilistic coclustering that extends the recent Functional Latent Block Model [SAJ18] in a Bayesian Non Parametric framework. This model also serves as basis component of the next chapter contribution.

4.1 Introduction

As stated in Sect. 1.5.2, LBM methods for time series coclustering, where each cell is a temporally-indexed vector, have only been introduced recently. The existing model-based methods [CB17; SAJ18; Bou+18; Sch+19; Bou+20; Cas+21] are parametric model, and assume that the number of blocks is known a priori. This assumption is rarely true in practice, and these dimensions must be estimated with an additional model selection step. This selection is usually performed either with a grid-search or with a greedy strategy that hierarchically explore the coclustering solutions. These strategies present several drawbacks:

- In the grid-search selection context, 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
- The greedy optimization heuristic is sub-optimal, by picking iteratively local optima and assuming a hierarchical structure of the mixture components
- Whether with the greedy optimization or grid-search, choosing the model selection criterion [CFR18; For+21] is not an easy task and influences the final results

The Dirichlet Process Mixture Model (DPMM), described in 1.3.2, is a Bayesian BNP 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 property makes it extremely interesting for massive dataset

exploration, especially when the user can augment the dataset content at will. This is precisely our situation in the driving validation context, in which it is possible to allocate additional resources to explore specific areas of the observation space.

Non-parametric approaches to LBM (NPLBM) have been studied in few works [MR07; WF12], but, to the best of our knowledge, never applied to multivariate time series coclustering use cases. This paper proposes the *functional non-parametric Latent Block Model* (FunNPLBM) method, the first non-parametric model-based method applied to time-series coclustering, which closes the gap between FunLBM and NPLBM. In addition, our contribution includes a practical use case illustrating the method's interest, a more compact definition of the NPLBM, a detailed study of the method's capacities based on benchmark and experiments, and Scala source code put at disposal for reproducibility.

In the following, Sect. 4.2.1 describes the model FunNPLBM and the proposed inference algorithm. Benchmark and experiments are studied in Sect. 4.3 and, finally, a real-case application on an industrial dataset is presented in Sect. 4.4.

4.2 Functional Bayesian Non-Parametric Latent Block Model

This section introduces our proposal of functional Bayesian non-parametric latent block model, FunNPLBM.

The next sections, Sect. 4.2.1 and Sect. 4.2.2 give the formal model definition and the associated stochastic inference process. Finally, Sect. 4.2.3 and Sect. 4.2.4 dive deeper into the algorithm implementation details: multivariate Gaussian observation model, initialization strategy, final inference, hyperparameter specification, and complexity.

In the following, we recall that the dataset X is an interpolated periodogram fPCA representation of the vectorized multivariate time series dataset as in the previous chapter (c.f. Sect. 3.2.1) .

4.2.1 Model Definition

In another context than the multivariate time series analysis, [MR07] proposed a definition of the NPLBM. This work assumes Pitman-Yor Process (PYP) priors for

the row-memberships and column-membership. However, PYP distributions (as DP) are distributions over parameters and not on memberships. Using PYP, the model from [MR07] implicitly defines different sets of parameter distributions that are not linked to the block distributions.

In the following, we propose a model definition of the NPLBM. Instead of generating each cell independently, this definition models the overall dataset X. Given fixed row-partition and column-partition $\mathbf{z}, \mathbf{w}$ and given the block parameter matrix (with dimension $1 \leq K \leq n$ and $1 \leq L \leq p$, $\Theta = (\theta_{k,l})_{K \times L}$, the element $x_{i,j}$ follows the distribution associated with its block membership. In our application, the block distribution $F(theta_{z_j,w_j})$ are multivariate Gaussian.

The overall process is defined by the following expressions:

$$
\begin{aligned}
x_{i,j} \mid \{\mathbf{z}, \mathbf{w}, \Theta\} &\sim F(\theta_{z_i,w_j}), \\
z_i \sim Mult(\pi),\ w_j &\sim Mult(\rho), \\
\pi \sim SB(\alpha),\ \rho \sim SB(\beta),\ \theta_{k,l} &\sim G_0.
\end{aligned}
$$

With this definition, the generation of the matrix X is done by generating the proportions π and ρ, sampling the row-memberships $\mathbf{z} \sim Mult(\pi)$ and column-memberships $\mathbf{w} \sim Mult(\rho)$ separately, then sampling the block parameter matrix Θ (in practice, only the ones associated with observed block defined by 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 $\mathrm{p}(X \mid \mathbf{z}, \mathbf{w}, \Theta) = \prod_{i,j} \mathrm{p}(x_{i,j} \mid \theta_{z_i,w_j})$, and the joint prior distribution of the hidden variables by:

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

In the next section, we describe the two-steps collapsed Gibbs sampling used to simulate draws from the posterior.

4.2.2 Inference

The objective is to obtain good estimates of $\mathbf{z}, \mathbf{w}$, based on the prior (α, β, G_0) and the observed dataset X, by sampling from the joint posterior distribution $\mathrm{p}(\mathbf{z}, \mathbf{w} \mid X, G_0, \alpha, \beta)$. Sampling directly from this joint distribution is not feasible, but can be approximated with a Gibbs sampler that iteratively update the values of the memberships $\mathbf{z}$ given $\mathbf{w}, X, G_0, \alpha$, and then $\mathbf{w}$ given $\mathbf{z}, X, G_0, \beta$. In our case we use a *collapsed* version of the Gibbs sampler, which uses directly 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 $\mathrm{p}(z_i = k \mid \mathbf{z}_{-i}, X, \mathbf{w}^{(m)}, \alpha, G_0) \propto$

$$\begin{cases} \dfrac{n_k}{n-1+\alpha}\ \mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}^{(m)}, X_{-i}, \mathbf{z}_{-i}, G_0), & \text{existing row-cluster } k, \quad (4.1) \\ \dfrac{\alpha}{n-1+\alpha}\ \mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}^{(m)}, G_0), & \text{new row-cluster,} \quad (4.2) \end{cases}$$

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

In eq. (4.2), the joint predictive distribution of the row $\mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}^{(m)}, G_0)$ is the product of the joint predictive distributions in the block components:

$$\mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}^{(m)}, G_0) = \prod_l \mathrm{p}(\mathbf{x}_{i,l}^{(m)} \mid G_0),$$

with $\mathbf{x}_{i,l}^{(m)}$ the elements of row i in column-cluster l at iteration m. The joint prior predictive distribution of $\mathbf{x}_{i,l}^{(m)}$ is obtained by integrating over the component's parameter:

$$\mathrm{p}(\mathbf{x}_{i,l}^{(m)} \mid G_0) = \int_\theta \mathrm{p}(\mathbf{x}_{i,l}^{(m)} \mid \theta)\ \mathrm{p}(\theta \mid G_0)\ d\theta.$$

Because G_0 is a prior conjugate to F, this integral is analytically tractable (cf. Sect. 4.2.3 for the detail in the multivariate Gaussian case).

The posterior predictive distribution in eq. (4.1) has a similar definition, with G_0 updated with the observations inside the blocks cf. Sect. 4.2.3).

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. 4.2.4 - §3). In the next subsection we detail how eq. (4.1) and (4.2) simplify with our choice of G_0.

4.2.3 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 $(\mu_0, \kappa_0, \Psi_0, \nu_0)$.

The next paragraphs details the joint predictive distributions closed forms that are necessary to compute the memberships probabilities (cf. proof and computation details in Appendix A.3).

Joint Prior Predictive Distribution The joint prior predictive distribution, used in Eq. (4.2) to compute $\mathrm{p}(\mathbf{x}_{i,\cdot} \mid \mathbf{w}^{(m)}, G_0) = \prod_l \mathrm{p}(\mathbf{x}_{i,l} \mid G_0)$, has the following expression:

$$\mathrm{p}(\mathbf{x}_{i,l} \mid G_0) = \pi^{\frac{-n_{i,l}d}{2}} \frac{\kappa_0^{d/2}}{\kappa_{i,l}^{d/2}} \cdot \frac{\Gamma_d(\nu_{i,l}/2)}{\Gamma_d(\nu_0/2)} \cdot \frac{|\Psi_0|^{\nu_0/2}}{|\Psi_{i,l}|^{\nu_{i,l}/2}}$$

where $n_{i,l}$ is the size of $\mathbf{x}_{i,l}$ and where the updated hyper-parameters values are obtained with:

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

Joint Posterior Predictive Distribution Given $X_{k,l}$, the observations in block (k,l), the block parameters posterior distribution is formally defined by $\mathrm{p}(\mu, \Sigma \mid X_{k,l}, G_0) = NIW(\mu, \Sigma \mid \mu_{k,l}, \kappa_{k,l}, \Psi_{k,l}, \nu_{k,l})$, with $(\mu_{k,l}, \kappa_{k,l}, \Psi_{k,l}, \nu_{k,l})$ the block posterior distribution parameters, updated from the prior:

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

With these parameters, the joint posterior predictive distribution needed in eq. (4.1) has the following expressions:

$$\mathrm{p}(\mathbf{x}_{i,l} \mid G_{k,l}) = \pi^{\frac{-n_{i,l}d}{2}} \frac{\kappa_{k,l}^{d/2}}{\kappa_{i,l}^{d/2}} \cdot \frac{\Gamma_d(\nu_{i,l}/2)}{\Gamma_d(\nu_{k,l}/2)} \cdot \frac{|\Psi_{k,l}|^{\nu_{k,l}/2}}{|\Psi_{i,l}|^{\nu_{i,l}/2}}$$

4.2.4 Implementation

In this section we give more details on the inference implementation: hyper-parameters specification, initializatin strategy, inference complexity

G_0 hyper-parameters 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, [SGH16] 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. We discuss the specification of the other hyperparameters in the experiments (c.f., Sect. 4.3).

Initialization strategy.

Initializing a Bayesian non-parametric MCMC inference algorithm is usually performed with single-component partition [NBA13; RM18]. However, [HLR15] shows that, for high-dimensional datasets with a high number of components, this strategy can give poor performances and recommends initializing the algorithm with a random partition with more components than the actual number of clusters. However, this number is unknown, and our objective is precisely to avoid its specification. A tempting workaround is to initialize the partitions with one component for each observation. However, this choice is computationally expensive because the membership update has linear complexity in the number of blocks. In practice, as a safety measure, we propose an heuristic, consisting in 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. If this maximal number is aberrantly high, it can be replaced by an upper quantile of the sampled cluster numbers.

Inferring the final partitions.

In the Bayesian Non-Parametric context, the MCMC-based inference process outputs samples of partitions that are drawn from an approximation of the posterior distribution. These sampled must, in turn, be aggregated over the iterations (usually after a given number of burnin iterations) with a consensus partition estimation (c.f. Sect. 1.3.3 for the clustering case).

In our application, the objective is to estimate the row and column partitions modes $\hat{\mathbf{z}}$, $\hat{\mathbf{w}}$. Because the row-partition and column-partition are independent, it is possible to resolve the consensus partition estimation problem independently on each dimension. With $(\hat{z}_m)_M$ and $(\hat{w}_m)_M$ the row and column-memberships sampled during the MCMC inference, the problem is equivalent to solve:

$$\hat{z} = \arg\min_{\mathcal{Z}} \sum_m d(\hat{z}_m, z),$$
$$\hat{w} = \arg\min_{\mathcal{W}} \sum_m d(\hat{w}_m, w).$$

Consequently, the problem can be solved by applying separately the same method [GOK18] than in the clustering case. The complete inference is summarized in algorithm 6.

Algorithm Complexity

During the row membership update step, the inference process updates the memberships one at a time. With $K^{(m)}$ and $L^{(m)}$ respectively the current number of row-cluster and column-cluster at iteration m, the update of row-membership z_i involves the computations of $(K^{(m)} + 1)$ membership probabilities ($(K^{(m)}$ existing clusters and the new cluster), and, for each membership probability, $L^{(m)}$ block parameters posterior parameters updates (based on Sect. 4.2.3) and predictive distribution density estimations (one for each block in the row-cluster).

In practice, instead of updating the block posterior parameters $(\mu_{k,l}, \kappa_{k,l}, \Psi_{k,l}, \nu_{k,l})$ with the full block contents $X_{k,l}$ at each iteration and at each membership update, it is more efficient to cache these parameters, and only remove $\mathbf{x}_{i,\cdot}$ from its previous

Algorithm 6: FunNPLBM Inference

input : Dataset X, $n \times p \times d$ tensor
α, β, G_0 (cf. specification strategies in Sect. 4.2.4 - §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. 4.2.4 - §2)

for $m \leftarrow 1$ **to** M **do**
  for $i \leftarrow 1$ **to** n **do**
    Remove $\mathbf{x}_{i,\cdot}$ from the blocks composing its current row-cluster.
    Compute $\mathrm{p}(z_i \mid \mathbf{z}_{-i}, X, \mathbf{w}^{(m)}, \alpha, G_0)$ as defined by eq. (4.1) and (4.2)
    Sample $z_i^{(m+1)}$
    Add $\mathbf{x}_{i,\cdot}$ to its new row-cluster.
  for $j \leftarrow 1$ **to** p **do**
    Remove $\mathbf{x}_{\cdot,j}$ from the blocks composing its current column-cluster.
    Compute $\mathrm{p}(w_j \mid \mathbf{w}_{-j}, X, \mathbf{z}^{(m+1)}, \beta, G_0)$ as defined by eq. (4.1) and (4.2)
    Sample $w_j^{(m+1)}$
    Add $\mathbf{x}_{\cdot,j}$ to its new column-cluster.

Compute the partitions averages $\hat{\mathbf{z}}$ and $\hat{\mathbf{w}}$ on the last samples (c.f. averaging methods in Sect. 4.2.4 - §3).

block-clusters before the membership probabilities computation, and add it to its new block-clusters after sampling.

With this caching, updating the block posterior distribution parameters of every row-clusters has a complexity reduced from $\mathcal{O}(npd^2)$ (dominated by the covariance matrix estimation cost), to two operations (removing and adding) with complexity $\mathcal{O}(pd^2)$.

Because of the matrix determinant computation cubic cost, the complexity of computing the joint predictive distributions for one of the $L^{(m)}$ column clusters is $\mathcal{O}(d^3)$, and the membership probabilities computations in eq. (4.1) and eq. (4.2) have a $\mathcal{O}(L^{(m)}d^3)$ complexity. This computation must be repeated $K^{(m)}$ times, once for each row-cluster. Overall, sampling one row membership has a complexity $\mathcal{O}(pd^2 + K^{(m)}L^{(m)}d^3)$.

This row-membership update is performed n times for the row memberships, and the symmetrical operation p times for the column memberships, which sums up to a complexity of $\mathcal{O}(npd^2 + (n+p)KLd^3)$.

With $\bar{K} = \max_M K^{(m)}$ and $\bar{L} = \max_M L^{(m)}$, the global complexity is

$$\mathcal{O}\left(M\left(npd^2 + (n+p)\bar{K}\bar{L}d^3\right)\right).$$

The complexity is therefore linear in the datasets dimensions, the number of block components and the iteration number. This expression also highlights the interest to keep the projection space dimension d as low as possible.

4.3 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 function and $s = 0.02$. 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.

The quality of the estimated block partition 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 that estimates the quantity of knowledge a partition gives on another. In addition, we also compare the number of co-clusters estimated by the different algorithms to the true number of co-clusters ($5 \times 5 = 25$).

Benchmark and experiments were run on a 64-bit Ubuntu 18.04 LTS with 32GiB RAM and 12 processors Intel(R) Core(TM) i7-8850H CPU @ 2.60GHz. The studied datasets, data generation scripts and the Scala code are available at the following github repository https://tinyurl.com/funNPLBM, along with the data simulation method.

4.3.1 Baselines and compared methods

As detailed in the introduction, coclustering methods have been the subject of numerous works and applied to various data types. However, most of the existing models and applications concern unidimensional datasets (i.e., $d = 1$). The coclustering on datasets containing multidimensional cells ($d > 1$) has been considered only recently in the literature. Apart from the model-based method from [CB17] which does not include a dimension reduction aspect and cannot be applied to large datasets, LBM

[Bou+18] is the only existing model-based parametric method dealing with this use case.

In addition to FunLBM and FunNPLBM (which we denote LBM and $NPLBM$ for brevity in the following), we consider two coclustering that infer the row-partition and column-partition independently: a bi-dimensional Gaussian Mixture Model (B-GMM) and a bi-dimensional DPMM (B-DPM).

In the parametric methods cases, the selection step is performed with a grid-search and the ICL selection criterion [BCG00; Ker+15]. This model selection tries every model possibility with a maximum of seven row clusters and seven column clusters, which represents respectively 2×7 models for the B-GMM, and 7×7 for the LBM. We denote B-GMM_{MS} and LBM_{MS} these parametric methods with model selection. For comparison, we also recorded the performances when directly considering the true block number (the methods are then simply denoted B-GMM and LBM). In each case, we also tested the effect of keeping the best result among several runs, corresponding to the same number of models tested in the model selection (respectively denoted B-GMM_{14} and LBM_{49}). In the non-parametric case, the model selection is natively performed. The concentration hyper-parameters are set to $\alpha = 0.1, \beta = 0.1$ and G_0 specified as described in 4.2.4. The baselines are summarized in Table 4.1.

Tab. 4.1.: Baselines summary

Method	Description	Grid	# runs
B-GMM	Decoupled clustering	True cluster numbers	2
B-GMM_{14}	- best among several runs	True cluster numbers	14
B-GMM_{MS}	- with model selection	$\{1,\dots,7\} \times \{1\}$, $\{1\} \times \{1,\dots,7\}$	14
LBM	Coclustering	True block number	1
LBM_{49}	- best among several runs	True block number	49
LBM_{MS}	- with model selection	$\{1,\dots,7\} \times \{1,\dots,7\}$	49
B-DPM	Decoupled DPM	-	2
$NPLBM$		-	1

For each method, the results are the average scores and median block number over ten runs. These performances are displayed in Table 4.2. The method B-GMM_{14} and LBM_{49} both give perfect scores for this dataset. Their scores show that when the true number of blocks is known and given enough re-trys, finding the true block partition with the parametric approaches is likely. On the contrary, the methods B-GMM and LBM have the worst performances, which proves that using re-trys is mandatory when using random initialization. This phenomenon comes presumably from a known sensibility of parametric model-based methods to initialization [BCG03]. However, a high number of re-trys increases the computation

time, as illustrated by comparing the computation times of LBM and LBM_{49}. We also note that the performances associated to $B\text{-}GMM$ are higher than the ones of LBM, which suggests that the LBM convergence might be harder to reach than the one of $B\text{-}GMM$.

Performing a model selection also increases the computation time, as is illustrated by the computation times of $B\text{-}GMM_{MS}$ and LBM_{MS}. The scores of these methods illustrate their capacities to find the true block partitions. However, when the grid-search spectrum is wide, when it is necessary to launch several re-trys, and when the dataset dimensions are high (or a combination of these three conditions), the associated computation cost becomes prohibitively high. In comparison, the $NPLBM$ method only runs once for an equivalent result and in less time. We acknowledge that these scores and runtimes depend on appropriate hyper-parameter specifications, which we discuss in the next sections.

Tab. 4.2.: Performance comparison of different coclustering methods

Method	ARI	RI	NMI	K	runtime (s)
$B\text{-}GMM$	0.825	0.982	0.946	25	18.6
$B\text{-}GMM_{MS}$	0.942	0.994	0.9803	30	85.9
$B\text{-}GMM_{14}$	0.979	0.997	0.994	25	89.3
LBM	0.823	0.913	0.887	25	17.5
LBM_{MS}	0.940	0.994	0.979	25	494.4
LBM_{49}	1	1	1	25	603.2
$B\text{-}DPMM$	0.670	0.958	0.906	16	25.3
$NPLBM$	**1**	**1**	**1**	25	**40**

4.3.2 Hyperparameters specification study

The complete set of the method's hyperparameters is composed of the base NIW distribution G_0, the concentration parameters (α, β), the number of iterations M and the preprocessing parameters: the Fourier basis dimension and the number of PCA axes.

In the unsupervised analysis context, hyperparameters specification remains an active research topic because there is no label to support hyperparameter inference. Consequently, it is not possible to give definitive good choices of values for the following hyper-parameters, 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 NPLBM's behavior, which we illustrate in the following. In each case, we compare NPLBM'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 NPLBM setting (as in the DPM) the prior distribution of the number of components is an increasing function of the concentration parameters: the higher the concentration parameters the higher the probability of producing high numbers of components, without data knowledge. 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. 4.1, the concentration parameters effects are negligible for this dataset and only have an impact for extreme values. When the concentration is extremely high ($> 10^{12}$), the number of components is highly overestimated. When its value is on the contrary extremely small ($< 10^{-10}$), only one-cluster partition is 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 $10^{-6} \leq \alpha \leq 10^{10}$) and the high stability of the MCMC chain.

For a given dataset, if the values of α and β seem to have a strong influence on the inference results, we advise to add a Gamma hyper-prior assumption for α and β and estimate their values during the inference algorithm. This strategy is a common practice in the DPM setting [Wes92]. However, it implies to study specification strategies for the Gamma distribution parameters.

Preprocessing parameters.

The Fourier Basis dimension and the PCA axes numbers influence both the performance of NPLBM and the trade-off between sparsity and quality of time series representation.

Fig. 4.2 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. If the number of axes is too high (>5), the high dimensionality

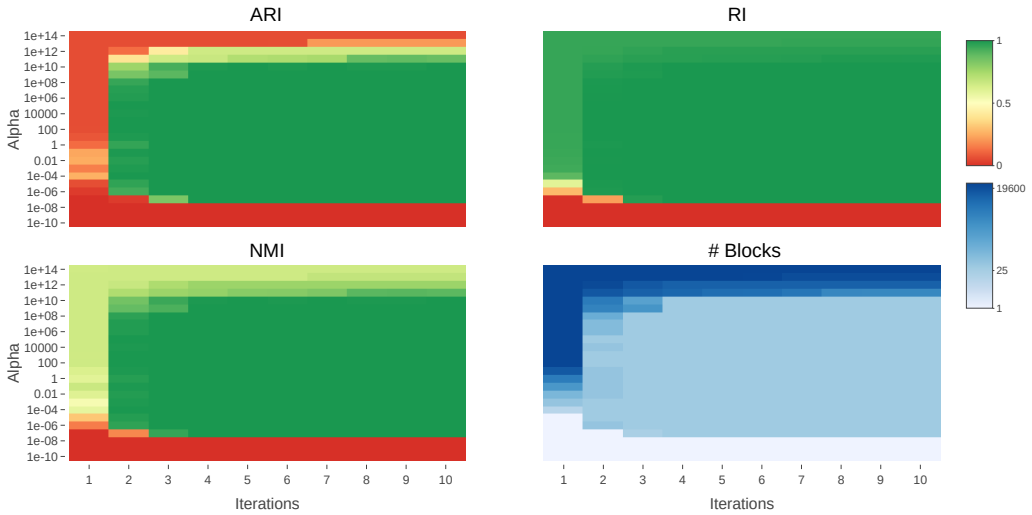


Fig. 4.1.: Scores versus Alpha and Iterations - with # Block in log scale

exaggerates the time series separation and the component number is overestimated, which explains the sharp decrease of the ARI and NMI.

In our use cases, we observed that three PCA axes lead to the most interesting results. In Fig. 4.3, 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 datasets, data generation scripts, and the Scala code used for the benchmark and the experiments are available at the following github repository https://tinyurl.com/funNPLBM, 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.

4.4 ADAS validation

Validating an ADAS is a complicated task, that is partly addressed with on-track tests. This practice has been gradually replaced with numerical simulations. The first motivation is the reduced testing cost compared to real physical tests which requires specific track infrastructure, equipment management, and significant human

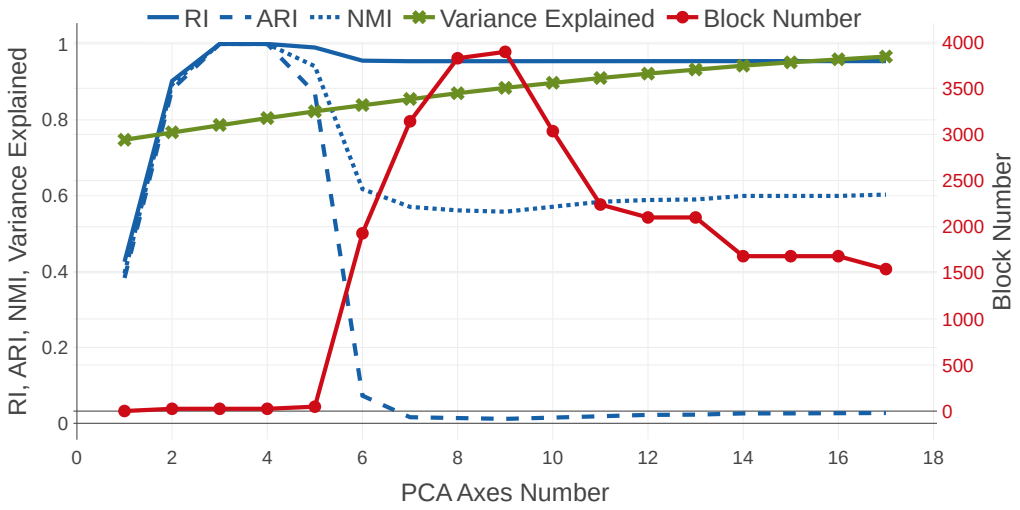


Fig. 4.2.: Scores versus number of PCA axes.

intervention. One digital simulation is estimated 10,000 times cheaper than its physical counterpart. The savings achieved through the use of digital simulation add up to millions of euros.

The second motivation comes from the fact that numerical validation remove uncertainties due to physical conditions (e.g. sensor error, initial conditions, weather conditions, ...). As a consequence, we are able to test the driving software independently from uncontrolled external factors as .

Another major disadvantage of physical tests is the prohibitive number of test sample to check for high reliability of the system. A validation objective may be the assessment of vehicle incident odds (e.g. $< 10^{-8}$ casualties per hour). With a classical sampling method, estimating such probability would require real driving sessions of billions of kilometers.

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 of the data quality or value. In our case, this value lies in the specific driving situation in which to test the system behavior. Indeed, some situations are rarely observed in reality such as emergency procedures, like the one considered in this numerical application.

4.4.1 Use case description

This section illustrates the use of the coclustering approach to the Emergency Lane Keeping (ELK) assistance system validation. In a straight lane scenario, the vehicle

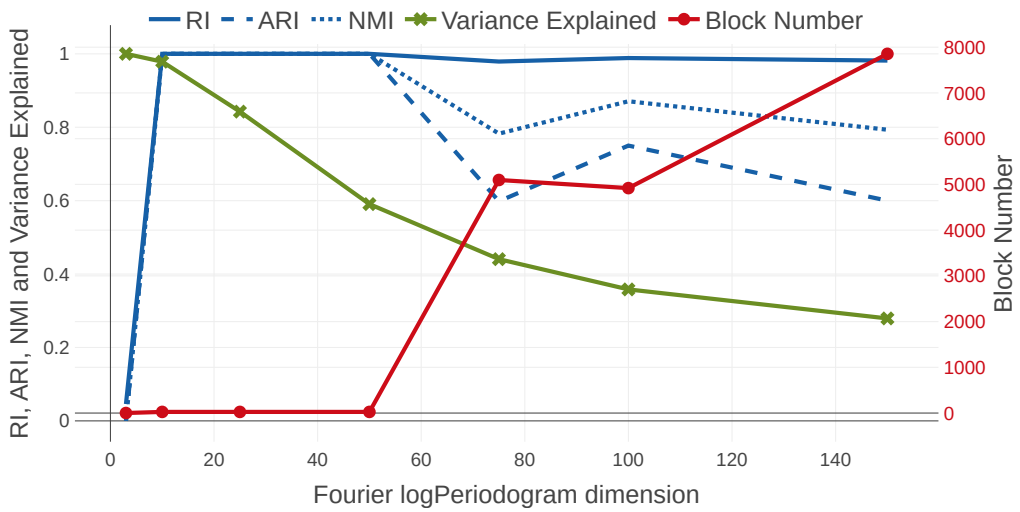


Fig. 4.3.: Scores versus logPeriodogram dimension.

under test (called *ego*) is drifting towards an oncoming car on the other lane (c.f. Fig. 4.4). The ELK system is expected to detect the drifting, the oncoming car, and to put the vehicle back to its lane center with an emergency maneuver.

For our dataset, we consider $n = 400$ simulations that were generated by simulating the ADAS according to a particular design of experiments (varying ego speed, the drift angle, ...). The objective of this analysis is to find characteristic operating modes of the intelligent system described as $p = 22$ time series and to discriminate relevant groups of correlated variables.

The time series are represented in a common log-periodogram with 40 coefficients and then projected on the 3 dominant PCA axes. The concentration parameters are both set to 10^{-2} and the NIW parameters to the default values discussed in Sect. 4.2.4.

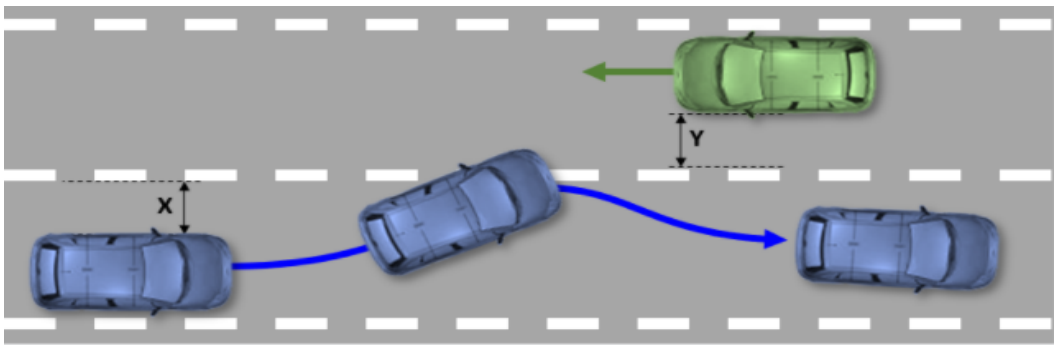


Fig. 4.4.: Use case illustration: ego drifts from its lane, crosses the center line and heads toward an oncoming vehicle. The system detects the target and change ego's direction.

4.4.2 Results

The final coclustering is the block partitions mean (c.f. averaging methods in Sect. 4.2.4) of 10 samples obtained after a burn-in of 10 iterations and is composed of 6 row-clusters and 13 column-clusters. With color indicating block membership, Fig. 4.5 shows the global coclustering structure and Fig. 4.6 an extract of the block contents.

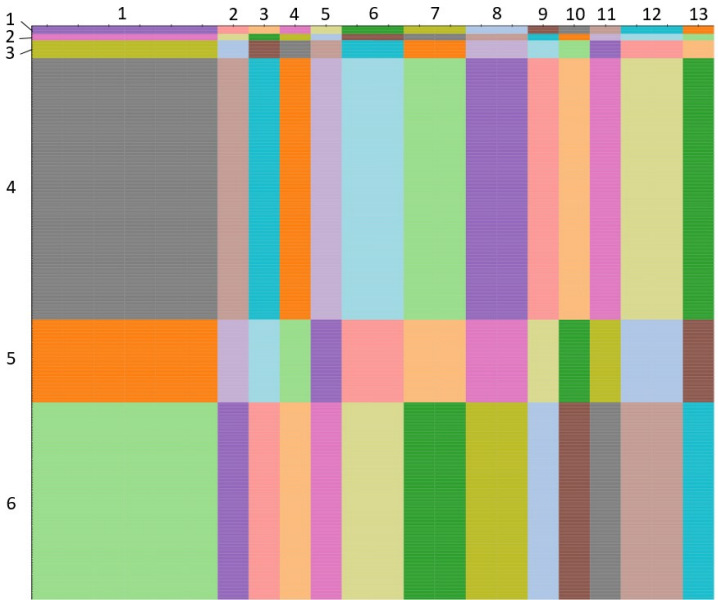


Fig. 4.5.: Resulting coclustering on Emergency Lane Keeping (ELK) dataset. The result consists of 6 row-clusters and 13 column-clusters

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. 4.6 top-left, top-right and bottom-left respectively. The 12-th cluster regroups two duplicated variables describing the covered distance.

The row-clustering is also insightful: each row-cluster correspond to well-separated driving behaviors. This separation is best seen in Fig. 4.6 (top-right) that shows the following driving behaviors: 1) ego is drifting left and the ELK system fails (light green); 2) the symmetric behavior on the right (dark green); 3) the ELK system corrects 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. 4.6 (bottom-right). In this situation, the oncoming car is correctly detected, ego heading is changed accordingly, but still the collision cannot be prevented. In conclu-

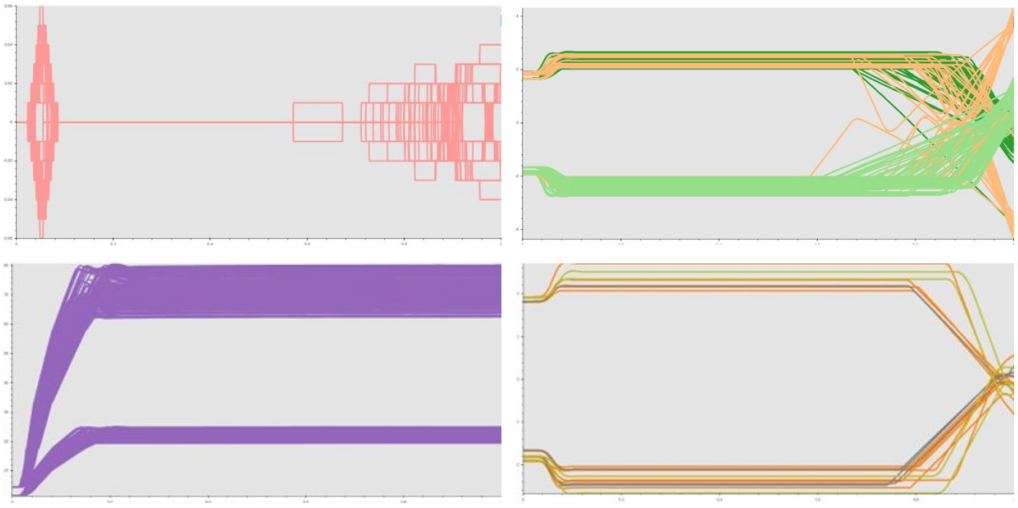


Fig. 4.6.: Top-left: two highly negatively correlated direction change signals; top-right: ego lateral position in the 3 biggest observation clusters; bottom-left: 2 correlated speed variables; bottom-right: 3 outlier driving patterns in the 3 smallest observation clusters.

sion, this coclustering partition shows that the approach has correctly discriminated uninformative signals while creating meaningful clusters of features and clusters of simulations. From this information, we can visualize the variety of driving behaviors that compose the datasets. Moreover, we can 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, optimize them to reach the performance objectives.

4.5 Conclusion

This chapter describes FunNPLBM, a Bayesian non-parametric based method which addresses the problem of coclustering multivariate time series without prior specification of the model dimension. This work offers a novel definition of the NPLBM model, the description of an adapted collapsed Gibbs sampling, and overall the first BNP coclustering method dedicated to multivariate time series data analysis. This model enables regrouping redundant signals and discriminating uninformative ones.

Moreover, the methodology provides a two-dimensional overview of a multivariate time series dataset to the users.

The specification of the different hyperparameters (concentrations, NIW, pre-processing) is discussed and experimented on a simulated dataset, which shows the model performance and gives insights on the most interesting hyper-parameter specification strategies.

Finally, the method is applied to a real-case dataset from the advanced driving-assistance system validation domain. In this application, FunNPLBM proved its ability to simultaneously discriminate groups of signals and to produce meaningful driving behavior clusters. These results show the usefulness of the model and of the model selection strategy in concrete modern challenging use cases.

The FunNPLBM approach was applied here to an autonomous driving context, however we are confident that it can be used in many other domains. A first interesting perspective would be the extension of a variational inference algorithm, a strategy that has been proven interesting in [BJ+06]. Another extension would be to define similar BNP concept to datasets of higher dimension and develop tensor coclustering methods for higher-dimensional functional problems.

Finally, the combination of FunNPLBM and CLBM from Chapt. 3 brings us to the next contribution titled Bayesian Non Parametric Multi-Clustering, that models each column-cluster with a specific FunNPLBM component.

5 Multivariate Time Series Multi-Coclustering

This chapter introduces a block structure model that combines two layer: a Multi-Clustering structure that groups together the variables with common row-partition, and a coclustering layer that associate one specific NPLBM structure to each column-cluster.

5.1 Introduction

In a multivariate context, the possible relations between variables introduces additional difficulties (c.f. Sect. 1.4.3). In real-life use cases, it is likely that each variable is associated to a different partition of the observation space (called *row-partition*).

In the following, we assume the presence of a variable partition (called *redundant column-partition*), such that variables in a redundant column-cluster share the same row-partition. In our use cases, this assumption seems rational, as we know that many variables are related (e.g., position and acceleration, ADAS activation and braking). The *Multi-Clustering* [HP18] designates a set of methods that infers this multiple-partitions representation of a dataset, as illustrated in Fig. 5.1 - b).

When in a *parametric* model-based context [MV19; Van20], these approaches requires to know the partitions sizes a priori (rarely true in practice), or to perform a model selection step. However, the combinatorial nature of Multi-Clustering makes the model selection a complex task (c.f. Sect. 3.2.4). Heuristic approaches could be considered at the cost of assumptions on model structure (e.g. with greedy strategies). The *non-parametric* approach circumvents this issue by natively performing a model selection during inference.

A non-parametric Bayesian Multi-Clustering model has been developed by [Gua+10] for continuous datasets. In this model, the rows belonging to a block follow an independent multivariate distribution (c.f. Fig. 5.1-b). It must be emphasized that, in this work, the block component distributions models the rows and not the cells as in the coclustering case. The reason for this is that each variable follows a different

distribution, which means that the overall model to estimate has a $p \times d$ dimension, and is invariant to the number of column-cluster.

This Multi-Clustering method groups variables according to their row partition but cannot regroup variables with similar distributions. Moreover, estimating the block distributions parameters becomes problematic when the column cluster sizes are important (e.g., in our case, high dimensional covariance matrices estimation). We propose to model each redundant column cluster with a coclustering structure to deal with this limitation. In each coclustering structure, the column-partition is called the *correlated* variable partition, and regroups variables with common distribution..

The model-based coclustering [GN13] (Fig. 5.1 - a)) infers one row-partition and one column partition. This model assumes that all variables share the same row partition and that there exists groups of variables with common distribution. We propose to model each redundant variable cluster with a coclustering structure, which assumes that some variables sharing the same row-partition can also share the same distribution. This assumption seems also natural, because some of the simulated variables are physically correlated (e.g., car speed and wheel rotation speed).

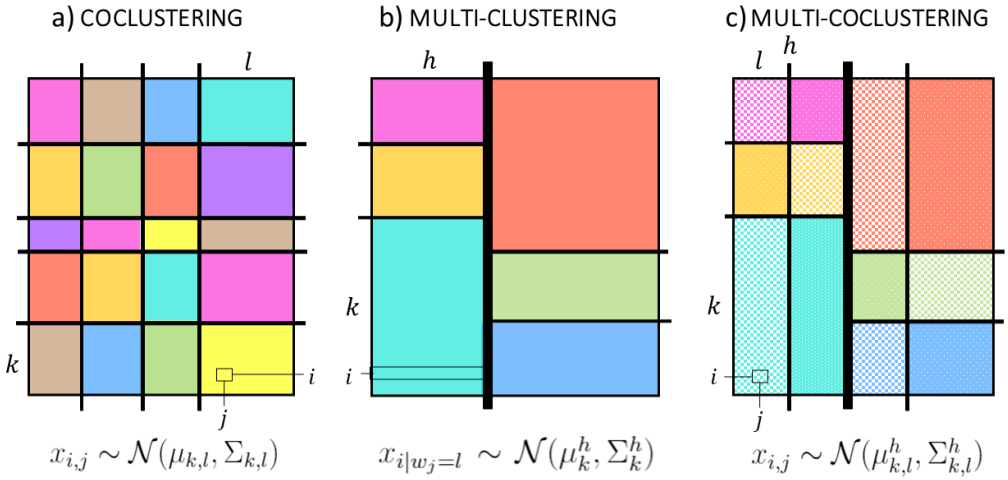


Fig. 5.1.: Coclustering, Multi-Clustering and Multi-Coclustering, with k, l, h the row-cluster, correlated cluster and redundant cluster indices (respectively). Color and pattern designate block membership.

In this paper, we propose the Functional Non-Parametric Multi Coclustering (Fun-MCC) model (c.f. Fig. 5.1-c) that combines the Multi-Clustering and coclustering approaches and produces the best of both worlds: the Multi-Clustering layer infers the redundant variable partition and extracts the true sparse number of row parti-

tion while the coclustering layer reduces the parameter set dimension and regroup correlated variables. To the best of our knowledge, there is only one comparable work by [Tok+17] in the bayesian non-parametric setting, which does not apply to multivariate time series.

5.2 Multiple Coclustering of multivariate time series

5.2.1 FunMCC model definition

In the following, the dataset X is obtained with interpolated periodogram fPCA representation (c.f. Sect. 3.2.1) as in the previous chapters.

As in Chapt. 3, $\mathbf{v}$ denotes the Multi-Clustering partition, i.e., the redundant partition, and Z the $n \times H$ row-partitions indicator matrix, with H the number of redundant clusters. As in Chapt. 4, the column-partitions inside each coclustering structure (i.e., the correlated partitions) are denoted $(\mathbf{w}^h)_H$ (with an additional superscript denoting the coclustering structure index). With this notation, the model is formally defined by:

$$
\begin{aligned}
x_{i,j} \mid \{v_j = h, w_j^h = l, z_i^h = k, \theta_{k,l}^h\} &\sim \mathcal{N}(\theta_{k,l}^h), \\
\theta_{k,l}^h \sim G_0,\ v_j \sim Mult(\eta),\ w_j^h &\sim Mult(\rho_h),\ z_i^h \sim Mult(\pi_h), \\
\eta_j(\mathbf{r}) = r_j \prod_{j'=1}^{j-1} (1 - r_{j'}), \quad r_j &\overset{\text{i.i.d.}}{\sim} Beta(1, \gamma) \ , \gamma \sim Gamma(a_\gamma, b_\gamma), \\
\rho_j^h(\mathbf{s}^h) = s_j^h \prod_{j'=1}^{j-1} \left(1 - s_{j'}^h\right), \quad s_j^h &\overset{\text{i.i.d.}}{\sim} Beta(1, \beta_h) \ , \beta_h \sim Gamma(a_\beta, b_\beta), \\
\pi_j^h(\mathbf{t}^h) = t_j^h \prod_{j'=1}^{j-1} \left(1 - t_{j'}^h\right), \quad t_j^h &\overset{\text{i.i.d.}}{\sim} Beta(1, \alpha_h),\ \alpha_h \sim Gamma(a_\alpha, b_\alpha),
\end{aligned}
$$

where each memberships proportions vector η, $(\rho^h)_h$ and $(\pi^h)_h$ follows a *stick-breaking* construction scheme [Set94]. Each concentration parameter follows a specific Gamma hyperprior.

The role of this additional hyper-prior is to grant the ability to infer and adapt the concentration parameter value to each coclustering structure, which are likely to contain different numbers of block components.

The block distributions are multivariate normal and G_0 is the associated conjugate Normal-inverse-Wishart prior. The complete hyperparameter set is noted χ.

It can be noted that, with this definition, constraining the correlated partitions to contain only one column-cluster makes the model equivalent to a Bayesian Non Parametric extension of the CLBM method described in Sect. 3. In addition, constraining the redundant partition to contain only one column-cluster makes the model equivalent to the FunNPLBM method described in Sect. 4. For these reasons, the FunMCC

5.2.2 Inference

We propose a stochastic iterative inference based on the following three-steps Gibbs sampler: 1) during the *Multi-Clustering layer* inference step, the redundant column partition $\mathbf{v}$ is updated given Z; 2) in the *Coclustering layer* inference step, the coclustering partitions Z and $(\mathbf{w}^h)_H$ are updated given $\mathbf{v}$ 3) Each concentration parameter is updated.

First step: Multi-Clustering layer

In the first step, the memberships $(v_j)_p$ are sampled sequentially according to:

$$\mathrm{p}(v_j \mid \mathbf{v}_{-j}, Z, \mathbf{x}_{.,j}, \chi) \propto \begin{cases} \dfrac{p_h}{p-1+\gamma}\ \mathrm{p}(\mathbf{x}_{.,j} \mid \mathbf{z}^h, \chi), & \text{existing cluster } h, \quad (5.1) \\ \dfrac{\gamma}{n-1+\gamma}\ \mathrm{p}(\mathbf{x}_{.,j} \mid \chi), & \text{new cluster,} \quad (5.2) \end{cases}$$

with $\mathbf{v}_{-j} = \{v_i : i \neq j\}$. In Eq. (5.1), the joint prior predictive distribution $\mathrm{p}(\mathbf{x}_{.,j} \mid \mathbf{z}^h, \chi)$ reduces to a product of multivariate t-student densities [Gel+13] thanks to an appropriate choice of conjugate prior G_0 (c.f. Sec. 4.2.3). In Eq. (5.2), $\mathrm{p}(\mathbf{x}_{.,j} \mid \chi)$ is estimated with an univariate Dirichlet Process Mixture (once per variable, prior to the inference of the FunMCC), which also produces the row-partition $\hat{\mathbf{z}}^j$ associated with variable j and to a possible new cluster.

Second step: Coclustering layer

In the second step, an NPLBM [MR07] is inferred on each sub-dataset $(X^h)_h$ defined by the partition $\mathbf{v}$ with a Gibbs algorithm that simulates $\mathrm{p}(\mathbf{z}^h, \mathbf{w}^h \mid \mathbf{v}, X^h, \chi)$. The inference is performed in two symmetrical steps: first, $\mathbf{z}^h$ is updated with fixed partition $\mathbf{w}^h$, then $\mathbf{w}^h$ is updated with fixed $\mathbf{z}^h$. Because these steps are symmetrical,

we only detail the column membership update in the following. For each variable j in sub-dataset X^h, w_j^h is updated by sampling from $\mathrm{p}(w_j^h \mid \mathbf{w}_{-j}^h, \mathbf{z}^h, X, \chi) \propto$

$$\begin{cases} \dfrac{p_l^h}{p_h - 1 + \beta} \displaystyle\prod_{k=1}^{K^h} \mathrm{p}(\mathbf{x}_{k,j}^h \mid \mathbf{x}_{k,l}^h, \chi), & \text{existing cluster } l, \quad (5.3) \\ \dfrac{\beta}{p_h - 1 + \beta} \displaystyle\prod_{k=1}^{K^h} \mathrm{p}(\mathbf{x}_{k,j}^h \mid \chi), & \text{new cluster,} \quad (5.4) \end{cases}$$

with $\mathbf{w}_{-j}^h$ and $\mathbf{z}^h$ the column and row partition indicator vectors without variable j, p_l^h the size of the correlated column cluster l, $\mathbf{x}_{k,j}^h = \{\mathbf{x}_{i,j} : \mathbf{v}_j = h, z_i^h = k\}$, and $\mathrm{p}(\mathbf{x}_{k,j}^h \mid \mathbf{x}_{k,l}^h, \chi)$, $\mathrm{p}(\mathbf{x}_{k,j}^h \mid \chi)$, respectively, the joint posterior and prior predictive distributions in block (k, l). After the row and correlated partition updates, the concentration parameters α_h and β_h are updated.

Third step: Concentration parameters update

Following [Wes92], the concentrations parameters γ, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are updated based on a Gamma prior. This update process is the same for every concentration parameter with their respective Gamma prior parameters. We also emphasize that the coclustering structures share the same hyperprior values: the concentration parameters $\boldsymbol{\alpha} = (\alpha_h)_H$ follow the prior $Gamma(a_\alpha, b_\alpha)$ and the concentration parameters $\boldsymbol{\beta} = (\beta_h)_H$ follow the same prior $Gamma(a_\beta, b_\beta)$. In the following we give the detail for γ's update.

Given H the current number of coclustering structures, γ the current value of the column-partition concentration partition, p the number of variables, and (a_γ, b_γ) the Gamma prior parameters, the update process is:

$$\begin{aligned} &(i) && \text{Sample } y \sim Beta(\gamma + 1, p) \\ &(ii) && \text{With probability proportional to } (a_\gamma + H + 1), \text{ draw} \\ & && \gamma\text{'s new value from } Gamma(a_\gamma + H, b_\gamma - log(y)) \\ &(iii) && \text{With probability proportional to } \mathrm{p}(b_\gamma - log(y)), \text{ draw} \\ & && \gamma\text{'s new value from } Gamma(a_\gamma + H - 1, b_r - log(y)) \end{aligned} \quad (5.5)$$

The same process is applied to $(\beta_h)_H$ and $(\alpha_h)_H$, with corresponding priors, number of elements and number of clusters. In the following we detail how the appropriate choice of conjugate prior G_0 prior enables the tractable computation of the prior and posterior predictive distributions used in Sect. 5.2.2 and Sect. 5.2.2.

5.2.3 Implementation

The implementation details are similar to the ones of the FunNPLBM discussed in Sect. 4.2.4:

- We also use a conjugate NIW prior with a zero-valued mean μ_0, a precision matrix Ψ_0 inferred from the dataset, and minimal values of κ_0 and ν_0
- The initial state is a one-cluster partition, and a second inference may be performed with block numbers higher than the one sampled in the first run.

The final partitions inference step is different than for the FunNPLBM method.

Inferring the final partitions.

In the Bayesian Non-Parametric context, the MCMC-based inference process outputs samples of partitions that are drawn from an approximation of the posterior distribution. These sampled must, in turn, be aggregated over the iterations (usually after a given number of burnin iterations).

In the FunNPLBM model, the row and column final partitions can be independently obtained by applying a consensus partition estimation [GOK18] separately on each dimension.

In our cases, the objective is to obtain the final sets of partitions modes $\hat{\mathbf{v}}$, $(\hat{\mathbf{z}}^h)_H$, and $\hat{\mathbf{w}}^h)_H$. Although it is straightforward to obtain the consensus redundant partition $\hat{\mathbf{v}}$ with [GOK18], the consensus on the other dimension partitions are much complex to estimate. This is caused by the label switching effect that makes the model likelihood invariant against component indices re-ordering.

In other words, if one wants to find the consensus correlated partition corresponding to redundant cluster h in $\hat{\mathbf{v}}$, one must first find, in every sampled FunMCC results, the FunNPLBM structure that match with the one associated with the redundant cluster h.

This problem can be seen as a consensus partition estimation with a hierarchical partitioning constraint.

Here are several workarounds solutions:

- A first method consists simply in keeping the last sample, which assumes that the MCMC has converged.

- It is also possible to select the sample that maximizes the model likelihood. This method seems the most natural, but is not a consensus estimation.

- A decoupled solution consists in first running the SEM-Gibbs to estimate the consensus redundant partition, then infer separately the FunNPBLM structure with fixed redundant clusters, and finally to estimate the consensus coclustering solutions of each FunNPLBM separately.

- As a coupled extension of the previous solution, a consensus partition estimation can be performed after each step (Multi Clustering and Coclustering layer) at each iteration. In that case, each step is based on the consensus partition(s) estimated in the previous.

Including this final partition estimation step, the complete inference process is summarized in Algorithm 7.

Algorithm 7: Hierarchical Inference Process For BNP Functional Multi-Coclustering

Input : dataset X, concentration parameters priors $(a_\gamma, b_\gamma), (a_\alpha, b_\alpha), (a_\beta, b_\beta)$, number of iterations M

Output : Redundant column partition $\hat{\mathbf{v}}$, correlated column partition $\hat{\mathbf{w}}$, Multiple row partitions $\hat{Z}$

$\mathbf{v}, \mathbf{w}, Z \leftarrow$ Initialize columns and row partitions

$\gamma, \boldsymbol{\alpha}, \boldsymbol{\beta} \leftarrow$ Initialize concentration parameters

for $j \leftarrow 1$ **to** p **do**

  Estimate variable j-th prior predictive distribution and univariate row-memberships $\hat{\mathbf{z}}_j$

for $m \leftarrow 1$ **to** M **do**

  for $j \leftarrow 1$ **to** p **do**

    Sample $\mathbf{v}_j$ based on Eq. (5.1), Eq. (5.2)

  for $h \leftarrow 1$ **to** H **do**

    for $i \leftarrow 1$ **to** n **do**

      Sample z_i^h based on equations (5.3) and (5.4).

    for $j \leftarrow 1$ **to** p_h **do**

      Sample w_j^h based on the same process as (5.3) and (5.4).

  Update concentration parameters γ, $\boldsymbol{\alpha}, \boldsymbol{\beta}$ based on equation (5.5)

Aggregate the sampled partitions (c.f. Sect. 5.2.3 - §3) to obtain the final consensus partitions $\hat{\mathbf{v}}$, $\hat{\mathbf{w}}$ and $\hat{\mathbf{Z}}$.

5.2.4 Algorithm Complexity

Before the main FunMCC inference, one DPM is inferred on each variable $\mathbf{x}_{.,j}$ independently, in order to estimate the expression $\mathrm{p}(\mathbf{x}_{.,j} \mid \chi)$ used in Eq.(5.2) and to obtain the row-partition $\hat{\mathbf{z}}^j$ associated with variable j "alone", i.e., the partition associated to a newly discovered cluster containing only the variable j. Each DPM has a complexity $\mathcal{O}(Mnd^2)$, with M the iterations number (with same value than for the main inference), n the observation number, and d the time series representation space dimension. This inference is performed on each variable, therefore, which amounts to a $\mathcal{O}(Mnpd^2)$ complexity. These estimations are only performed once, then can be cached.

After this preparation step, the main FunMCC inference process begins, alternating the two phases: the Multi-Clustering layer update and the coclustering layer updates.

During the Multi-Clustering layer inference, the redundant cluster memberships are updated sequentially following 5.2.2, with a fixed value of Z. The update of one membership v_j involves the computation of $H+1$ membership probabilities (with H the number of existing redundant clusters plus one new cluster). For a given existing redundant cluster h, computing the redundant membership probability of column $\mathbf{x}_{.,j}$ involves, in turn, to compute one joint prior predictive for each of the K blocks (one for each row-cluster) that composes the column-cluster h. These updates, detailed in Sect. 4.2.3, consists in combining the n observations by groups into K means, and K covariance matrices, with an overall $\mathcal{O}(nd^2)$ complexity (dominated by the covariance matrix estimation cost).

As opposed to the FunNPLBM case (cf. Sect. 4.2.4), it is not possible to cache the updated hyper-parameters values, because they are only computed once at each iteration (for this particular membership distribution computation) given a fixed value of Z that changes during the coclustering update set.

After having updated the prior hyper-parameters, the membership probability is finally obtained by computing $\mathrm{p}(\mathbf{x}_{.,j} \mid \mathbf{z}^h, \chi)$ the joint predictive distribution of variable j. This involves K^h computations (with K^h the current number of row-clusters in redundant cluster h), each with complexity d^3. This operation is performed once for each of the H existing cluster, and for each of the p variables, which amounts to a complexity

$$\mathcal{O}(p\bar{H}(nd^2 + \bar{K}d^3)), \tag{5.6}$$

where $\bar{H} = \max_m H^{(m)}$ and $\bar{K} = \max_{m,h} K^{(m),h}$ are the maximum of the cluster number values sampled during inference.

The coclustering layer inference consists in updating H FunNPLBM structures, therefore with complexity (derived in Sect. 4.2.4) $\mathcal{O}\left(\bar{H}\left(npd^2 + (n+p)\bar{K}\bar{L}d^3\right)\right)$, with $\bar{L} = \max_{m,h} L^{(m),h}$.

The multi-clustering and coclustering layer updates are repeated for M iterations. The final complexity is, after simplification:

$$\mathcal{O}\left(M\bar{H}\left(npd^2 + (n+p)\bar{K}\bar{L}d^3\right)\right),$$

with M the number of iterations. This complexity is linear in the datasets dimensions, the number of block components and the iteration number.

It may be noted that the coclustering structures can be updated in parallel, which would cancel the linear complexity in $\bar{H}$. It is also possible to parallelize the independent DPM that are inferred before the main FunMCC inference to estimate the distributions $\mathrm{p}(\mathbf{x}_{.,j} \mid \chi)$.

5.3 Experiments

In the following section we highlight the interest of using FunMCC by comparing the impact of uninformatives or misleading variables on several block clustering baselines. The Scala source code of FunMCC is available at https://tinyurl.com/FunMCC, along with the data simulation script.

The experiments dataset is composed of observations generated from a ground truth row-partition, that we seek to estimate. These observations are described by several groups of variables sharing common distribution. This ground-truth dataset is therefore a coclustering structure.

This ground-truth information is "corrupted" by adding a proportion of other variables: uninformatives (generated from a one-cluster partition) or misleading (generated from another row-partition than the ground truth). These proportions are respectively denoted p_u and p_m. In the ground-truth coclustering structure, 30 variables and 150 rows are distributed in 5 row-clusters and 3 column-clusters. The uninformatives variables are simulated with one-partition variables, and the misleading structure with another coclustering structure with 4 row-clusters and 3 column clusters. The ground-truth and misleading row-partitions are randomly

and independently generated. With the proportions p_u and p_m varying from 0 to 200%, this setup generates 7 datasets, each one corresponding to the ground truth and given uninformative or/and misleading proportions. The objective is to find the ground-truth without supervision, and to discriminate and separate the uninformative and misleading information.

The elements belonging to a given block (i.e., to one particular redundant cluster h, and one block (k, l) in the coclustering structure h) are 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 $s = 0.02$. An instance of such dataset is displayed in Fig. 5.2, with equal proportions of ground-truth, uninformative and misleading variables. This figure also represents three prototypes used to generate the associated block contents.

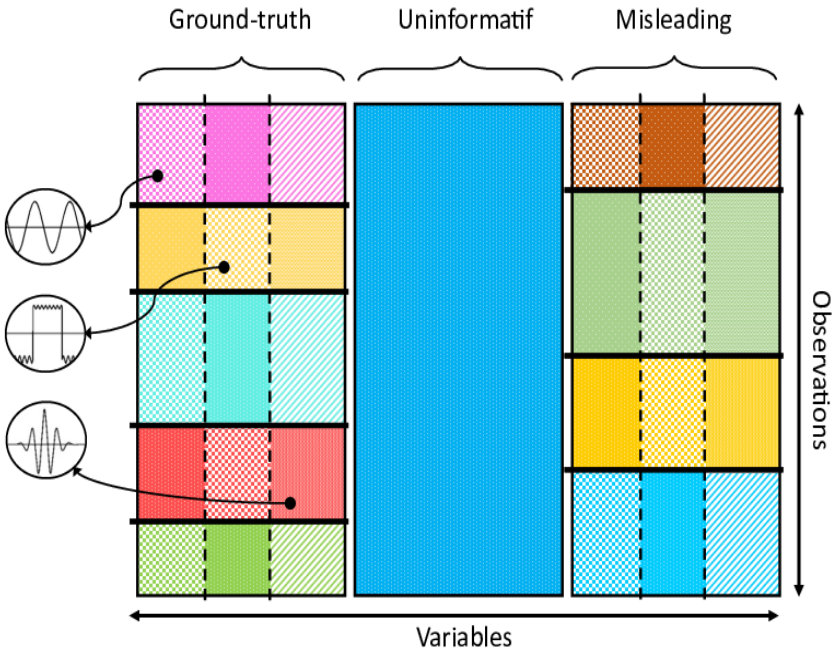


Fig. 5.2.: Illustration of the ground truth coclustering structure, the uninformative variables and the misleading variables. Color and pattern indicates block memberships. The time series profiles displayed on the left represent a sample of the prototypes used for block generation.

We compare FunMCC's performances to baselines non-parametric model-based approaches, that match to our specifications. We consider a Dirichlet Process Mixture model (DPM) that infers a row-partition without a column-partition and represents the default strategy without variable discrimination. The second baseline is a non-parametric coclustering method (CC) [MR07] displayed in Fig. 5.1, that adds a column-partition. As alternative Multi-Clustering solutions, we consider three methods derivated from our proposal FunMCC:

- Decoupled Multi-Clustering based on DPM (DPDM) consisting of one DPM on the columns, then one on the rows.
- Decoupled Multi-Clustering based on Coclustering (DCC) that first infers the redundant partition with a single global coclustering, then infer a coclustering structure in each of the obtained redundant cluster.
- A non-parametric Multi-clustering (MC), equivalent to FunMCC without the coclustering layer.

The performances comparison is based on two popular scores: the Adjusted Rand Index (ARI), and the Normalized Mutual Information (NMI). The ARI is a corrected-for-chance version of the Rand Index that evaluates the proportions of correctly grouped and separated elements in a partition. The NMI is an entropy-based criterion from the information theory literature estimating the quantity of knowledge a partition gives on another.

In addition, we also compare the estimated number of clusters, $\hat{K}$, as a complementary information. Because the selected baselines are stochastic inference processes, we always consider the scores mean obtained over ten launches (or the median in the case of the cluster number). When several row-partitions are produced (i.e., Multi-Clustering methods), the closest row-partition is considered, with respect to the ARI.

The results, displayed in Table. 5.1, show a gradual performance increase ranging from DPM to FunMCC. The DPM, which only creates one row-partition, is the most impacted by the presence of uninformative and misleading variables. The DPM performances are gradually worsened when the p_m (the misleading proportion) increases. On the contrary, even a small proportion p_u (the uninformative proportion) reduces DPM's performances dramatically. This is presumably induced by the non-parametric setting that selects the most likely number of row-clusters, and favours one-cluster partitions when adding uninformative signals. On the contrary, the presence of misleading information does not reduce the number of row-clusters, but corrupts their contents, resulting in lower performances with an approximatively correct number of row-clusters.

The CC method is not impacted by the presence of uninformative signal, but tends to overestimate the row-cluster number in presence of misleading variables. This overestimated $\hat{K}$ is associated to the row-partition that results from crossing the ground-truth row-partition (5 row-clusters) with the misleading row-partition (4 row-clusters), resulting in a 8-clusters row-partition, as approximated by CC. The Decoupled methods (DDPM and DCC) have similar performances than CC, with a

Tab. 5.1.: Ground-truth estimation quality described by ARI, RI and row-cluster number $\hat{K}$, of the Dirichlet Process Mixture (DPM), Bayesian Non-Parametric Coclustering (CC), symmetrical decoupled Dirichlet Process Model (DDPM), Multi-Clustering based on a coclustering structure (DCC), Multi-Clustering without the coclustering layer (MC) and our proposal (FunMCC).

Score	p_u	p_m	DPM	CC	DDPM	DCC	MC	FunMCC
ARI	0 %	0 %	**1.00**	**1.00**	**1.00**	0.98	**1.00**	**1.00**
	0 %	50 %	0.78	0.88	0.48	0.52	**1.00**	**1.00**
	0 %	100 %	0.61	0.81	0.43	0.43	0.96	**1.00**
	0 %	200 %	0.36	0.88	0.53	0.53	**1.00**	**1.00**
	50 %	0 %	0.00	**1.00**	**1.00**	0.99	**1.00**	**1.00**
	100 %	0 %	0.00	**1.00**	0.99	0.98	0.96	**1.00**
	200 %	0 %	0.00	**1.00**	**1.00**	0.97	**1.00**	**1.00**
NMI	0 %	0 %	**1.00**	**1.00**	**1.00**	0.99	**1.00**	**1.00**
	0 %	50 %	0.91	0.93	0.66	0.70	**1.00**	**1.00**
	0 %	100 %	0.77	0.89	0.60	0.61	0.98	**1.00**
	0 %	200 %	0.55	0.93	0.71	0.71	**1.00**	**1.00**
	50 %	0 %	0.00	**1.00**	**1.00**	0.99	**1.00**	**1.00**
	100 %	0 %	0.00	**1.00**	0.99	0.98	0.98	**1.00**
	200 %	0 %	0.00	**1.00**	**1.00**	0.97	**1.00**	**1.00**
$\hat{K}$	0 %	0 %	**5.00**	**5.00**	**5.00**	**5.00**	**5.00**	**5.00**
	0 %	50 %	4.00	7.00	**5.00**	**5.00**	**5.00**	**5.00**
	0 %	100 %	6.00	8.00	4.00	**5.00**	**5.00**	**5.00**
	0 %	200 %	3.00	7.00	4.00	4.00	**5.00**	**5.00**
	50 %	0 %	1.00	**5.00**	**5.00**	**5.00**	**5.00**	**5.00**
	100 %	0 %	1.00	**5.00**	**5.00**	**5.00**	**5.00**	**5.00**
	200 %	0 %	1.00	**5.00**	**5.00**	6.00	**5.00**	**5.00**

nearly null impact of p_u, but with a greater impact of p_m. This effect is linked to a low-quality redundant column-clustering, due to the decoupled aspect, that in turn impacts the row-partition quality. With the decoupled methods, the Multi-Clustering is the result of two steps: a column-partition inference, then one row-partition inference in each column-cluster. However, the row-partitions information, obtained in this second step, cannot be used in the column-cluster construction.

On the contrary, the pure non-parametric Multi-Clustering uses the row-partitions information to construct the redundant column-partition, which explains a better column-partition quality, and, as a consequence, a stronger stability against the presence of misleading variables. We observe a light performance increase with the addition of a coclustering structure layer (our proposal FunMCC), that regroups variables with similar distributions. Our intuition is that the coclustering layer acts as a shrinkage method that reduces the parameters number and the global model complexity while keeping the same overall density representation capacity. This

phenomenon presumably increases the Multi-Clustering layer quality, in addition to enhancing the model interpretability.

5.4 Applications

5.4.1 Emergency lane Keeping Assist

After experimenting FunMCC on a synthetic dataset, we apply it on a real ADAS validation dataset. In a straight lane scenario, the vehicle under test is drifting towards an oncoming car on the other lane. The ELK system is expected to detect the drifting, the oncoming car and put it back to its lane center with an emergency maneuver. The simulation generates a dataset of $n = 500$ described by $p = 150$ temporal variables, totalling 75000 time series. We emphasize that these simulations are generated with a simulation black-box that faithfully recreates the real-life driving conditions, and are not produced by the generative model proposed in this article.

The objective is to infer a synthetic structure that highlights interesting driving patterns and discriminate relevant groups of sensors, in order to isolate and understand the ADAS activation contexts.

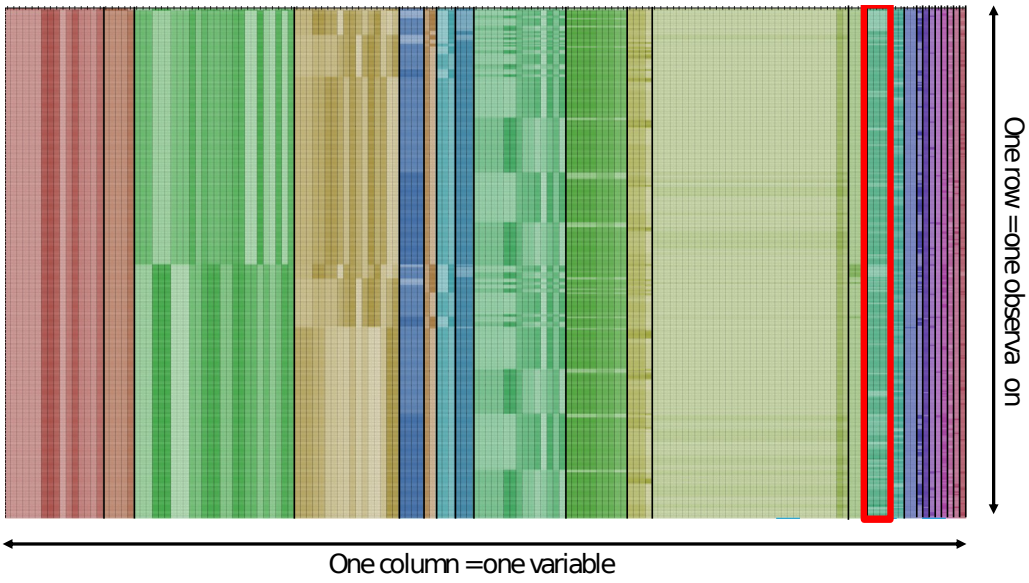


Fig. 5.3.: Multiple Coclustering structure overlaid on the ELK dataset (one cell = one time series). The red rectangle outlines the trajectory variables depicted in Fig. 5.4.

The Multi-Coclustering structure is displayed on Fig. 5.3, which represents the row partitions overlaid on the dataset, with color representing redundant column clusters, and transparency indicating block partitions.

A first interesting results is that the non-informative variables variables are grouped together in the red and orange column-clusters. This group contains normally uninformative variable such as the car length, the time from start, but also use case specific (e.g., road radius in a straight lane scenario, headlights activation while in broad daylight, ...).

Several other column-clusters are particularly interesting:

- The leftmost green column-cluster contains orientation and car direction variables (e.g., the car orientation, the wheel angle, the steering wheel angle, ...)
- The next ochre column-cluster on the figure contains position variables: lateral position, index and type of the current lane, distance to the road center, ...
- The next small blue column-cluster contains ADAS activation variables, which are categorical time series. These variables can also indicates the activation "mode" (for instance, activation on the left or right).

The column-clusters composition exhibits one outstanding feature of this approach: the column-cluster acts as *meta*-variables, that regroup the meaning of several others. This process acts simultaneously as a variable selection and dimension reduction step, and helps the interpretation.

Another interesting information that appears on Fig. 5.3 is the hierarchical structure between the row partitions. Because the simulations are sorted by row-partitions, and based on the order of the row-partitions, the successive sorting creates a hierarchical clustering. The specification of the row-partitions order must be specified by the user. This problematic is specific to the Multi-Clustering case, and is discussed in the perspectives (Sect. 6.2.2).

The Fig. 5.4 illustrates the content of a column-cluster, and shows the car trajectories (surrounded by a red rectangle in Fig. 5.3) distributed in the row-clusters, where several scenarios are represented: the sub-graph I shows a late ADAS activation scenario, sub-graph II regroups cases when ELK did not activate at all. In sub-graph III, the systems correctly change the car trajectory and prevent the collision. The last graph IV shows a very small cluster of anormal driving behaviors detected and grouped by FunMCC.

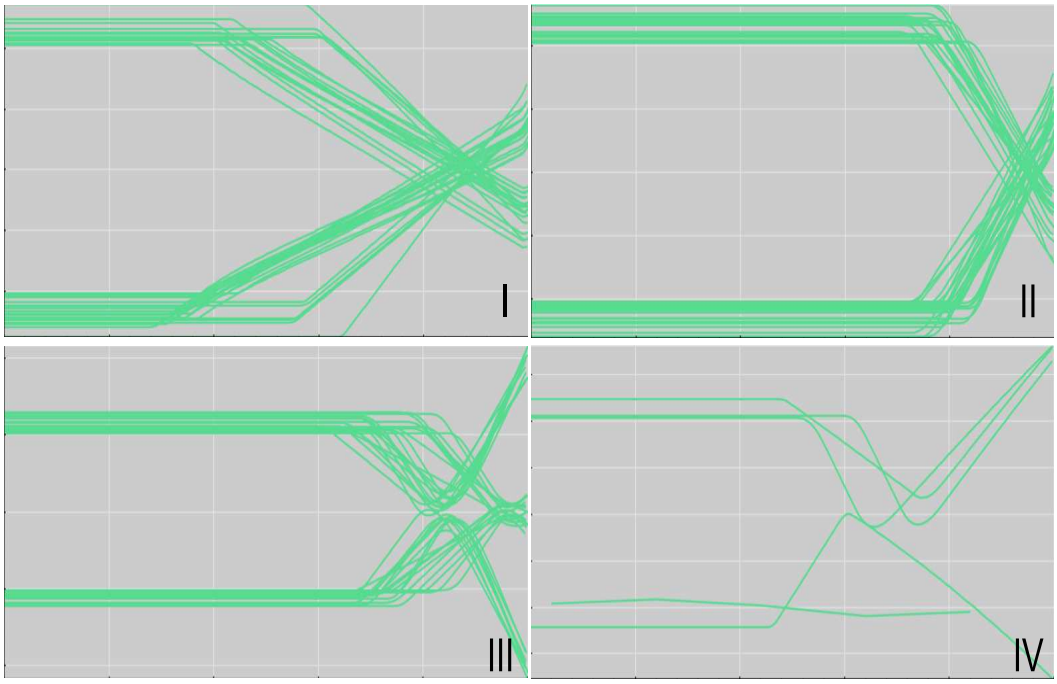


Fig. 5.4.: Car trajectories distributed in the clusters, corresponding to the content of the red rectangle outline in Fig. 5.3.

Finally, the understanding of the Multi-Coclustering result structure can be enhanced by representing the variables in their own space and in their column-partition. This representation is given in Fig. 5.5, that shows an MDS projection based on both row-partitions distance and variable distribution. This figure outlines the presence of well separated groups of variables.

Based on these results, the next steps for the ADAS system developer consists in relating the partitions to the input simulation parameters and assessing the ADAS compliance with its specifications.

5.5 Conclusion

This chapter describes a new Bayesian non-parametric based method designed for the exploration of multivariate time series datasets produced by driving simulations. This solution infers a multi-level dependency structure that highlights the relationships between sensors and discriminates driving simulations patterns. This probabilistic method can natively be used for anomaly detection based on probabilistic predictive intervals.

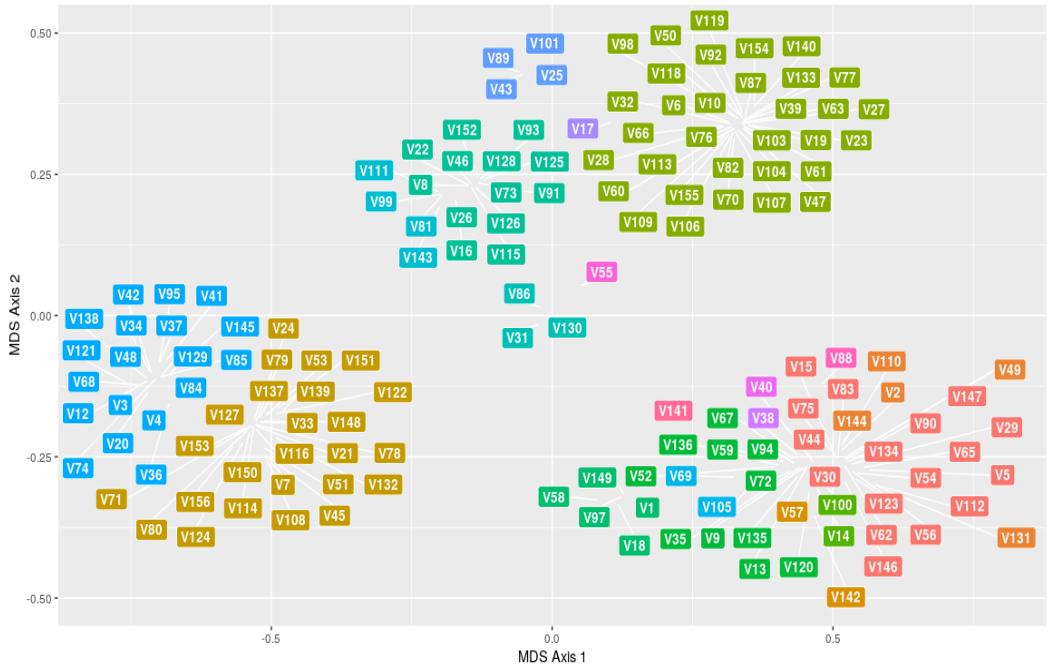


Fig. 5.5.: Redundant variable partition projected with MDS

The next step in the ADAS validation pipeline is to use the multivariate time-series multi-coclustering results as labels for a supervised analysis of the simulation parameters, in order to understand and explain ADAS failure and success conditions.

Conclusion And Perspectives

6

6.1 Dictionary-based time series clustering

In the context of unsupervised classification of regime-changing time-series, this chapter proposes a dictionary-based method that consists in three steps: automatic segmentation of each time-series, regime dictionary construction, and clustering of produced categorical sequences. $SDLHC$ shows good results when applied to time-series complying to the regime construction assumption, and is competitive with other state-of-the-art methods in this case. The ability to handle unequal-length time-series, a-synchronized time-series, and time-series exhibiting asynchronous regimes are its best assets.

One weak point of this method is the complexity associated to the third step, which is $\mathcal{O}(n^3)$. This complexity makes the method quite slow when the number of time series becomes important. In that case, alternatives methods should be considered, e.g., using a partition-based clustering on histograms, at the cost of losing the time-dependency information.

One other weak point is that this method can only be applied to univariate time series, which limits its application to specific use cases. In the ADAS validation context, this means that the user is limited to the exploration of one simulated variable. The multivariate extension can be addressed in several ways.

6.1.1 Joint Multivariate Segmentation

The easiest extension to multivariate clustering consists in assuming that, for a given MTS observation, every dimension share the same segment cut-points (i.e., the same univariate hidden logistic process). Under this assumption, it is possible to perform a multivariate joint segmentation, similar to [Cha+09b] with multidimensional segments. This method has been described in [Cha+13] by the same authors.

In this case, the dictionary construction step can be extended in a multivariate GMM that regroups multivariate segments, and the produced driving patterns are also multivariate. The MTS are, in turn, recoded as univariate symbol sequences, which

leaves the final step (univariate symbol sequence clustering) unchanged. The overall transformation and clustering can be considered a dependent approach, as every dimensions are considered jointly.

The joint multivariate segmentation can be suited for dataset composed of a small set of correlated variables (e.g., position and speed variables). However, this method is not suited to deal with a large number of variables, when the synchronous cutpoint hypothesis becomes more and more unrealistic.

6.1.2 Independent Univariate Segmentation

A short-cut to the previously described limitation is to independently segment every dimensions. In this case, three alternatives can be considered.

Cut-points Intersection

A first solution consists in taking the union of the cutpoints cut-points of every dimensions, and segments the MTS in smaller sub-sequence.

After this step, the dictionary can perform, as in the previous variant, a multi-dimensional segment clustering that produces multivariate patterns. The MTS are recoded in univariate symbol sequences, and the final hierarchical clustering remains unchanged.

However, intersecting the cut-points of every dimensions can lead to an excessive sub-sampling of the segments, and to an important loss of information.

For instance, given a p-dimensional MTS with length T, and assuming that the p-dimensions have, on average, K distinct cut-points, the union of these cutpoints generates a pK cutpoints. As a consequence, the average length of the d-dimensional sub-sequences is $\frac{T}{pK}$. With plausible values ($T = 1000$, $p = 100$, $k = 10$), the sub-sequences becomes singleton sub-sequences, which cancels the dimension reduction effects of the segmentation step, and makes the dictionary construction step (a GMM on a $\mathbb{R}^d = \mathbb{R}^{100}$ space) unfeasible.

Symbol Sequences Intersection

Instead of taking the union of the cut-points, which may greatly over-estimate the number of segments, it is possible to apply the union operation on the uni-dimensional symbol sequences.

In this variant, after the univariate segmentation of each dimension, the dictionary reconstruction step is applied independently on each dimension. As a results, each MTS is recoded as a multivariate symbol sequence.

After this phase, each unique combination of univariate state is associated to a new multivariate symbol. For instance, given an MTS $T_1(t)$ represented with a two-dimensional symbol sequence $((a, a, a, b, b, b), (d, d, e, e, f, f))$, the resulting representation would be (A, A, B, C, D, D), with $A = (a, d), B = (a, e), C = (b, e), D = (b, f)$.

This method implies to construct a new dictionary that associates a unique symbol to each multidimensional symbol state. As a result, each MTS can be represented with a univariate symbol sequence, and the final clustering step remains unchanged.

The drawback of this method is the same as the previous cut-points union method: when the number of variable is high, the length of the sub-sequences associated to each multivariate state decreases, and the number of possible multivariate state increases sharply.

The problem is the same if the dictionary construction is performed on the overall set of segments from each dimension rather than dimension-wise (with D replaced by the size of the overall dictionary).

Independent Symbol Sequence Comparison

Instead of combining the multi-dimensional symbol sequences to obtain univariate strings, this variant proposes to keep a multi-dimensional string representation and to perform the last clustering step based on this multi-dimensional representation.

In that case, the dictionary method remains unchanged and deals with univariate segments, either applied dimension per dimension or to every segments from every dimensions), as in Sect. 6.1.2. As a result, each MTS is recoded as a tuple of symbol sequences.

Based on this representation, the MTS are compared by summing their dimension-wise difference string dissimilarity, e.g., the Levenshtein distance. This setting implies

to compute the dimension-wise sub-sequences dissimilarities and the Levenshtein distance matrix, which adds a linear complexity in d to the step complexity. This solution can, up to the final hierarchical clustering, be parallelized or distributed per dimension, which makes it scalable and interesting for industrial applications.

A major drawback of this solution is that each dimension has the same weight, which makes the overall method sensible to uninformative or misleading variables. Moreover, summing the distances in this high-dimensional space produces strongly separated observations.

In order to solve this problem, a perspective would be to consider weighted version of this multi-dimensional Levenshtein Distance.

6.1.3 Coclustering extension

Another approach to extend the method to the multivariate case is to consider a coclustering extension. Using the fact that the block component are uni-dimensional distribution (shared by every variables inside a variable cluster), our first idea was to develop a model-based version of $SDLHC$ that could be extended to the multivariate case with the coclustering framework.

In order to obtain this model-based extension, it is necessary to replace the last step of $SDLHC$ (the symbol sequence clustering), with a model-based clustering method. After this replacement, the objective was to collapse the entire method and to perform a joint inference of the segmentation, dictionary construction and symbol sequence clustering in a single inference process.

However, the development of a model-based categorical sequence clustering has proven difficult, partly because of the complexity of estimating a consensus categorical sequence. This limitation led us to modify our strategy, and adopt a specific representation for the multivariate case.

6.2 Multi-Coclustering

The Multi-Coclustering method can be extended in several ways. As a model-based method, it can natively be used for anomaly detection (e.g., by considering a component-wise likelihood ratio). It can also serve as a supervised classification method, by simply constraining the redundant, correlated and row partition values. In that case, the inference would lead to a multiple coclustering density

estimation, that would enable a multiple classification of a given observation. In a semi-supervised application, the method can be used to infer missing value in one coclustering structure based on the others.

FunNPLBM can be useful in every domains that deal with datasets containing multiple row-partitions and correlated temporal variables. For instance in industrial contexts for predictive maintenance based on multiple sensors, in health for ECG and biological signals data analysis, in finance for stock trade data analysis.

Another natural extension would be the *online* multiple coclustering applied to the real-time detection of driving states and the online anomaly detection.

6.2.1 Method Scalability

One drawback of the inference method is that, as is the case for the DPM, the memberships (of every dimensions) are updated one by one given the other observation and other memberships. Because of this sequential inference, it is not straightforward to parallelize the inference at the individual level. This can cause some inference issues when the datasets dimensions are too important.

However, following recent advances in the field [Meg+19], it is possible to distribute the computation when the dataset is important. The DPMM distribution consists in running independent DPMM on sub-datasets, and then regroup the obtained clusters. In the next works we suggest to develop an extension of the work of [Meg+19] by distributing the FunNPLBM, and the FunMCC.

6.2.2 Representing the variables relationships

The MCC method outputs several row-partitions, but also a nested column-partition: based on the observation clustering, then on the distributions. An interesting perspective for the user is to visualize the relationships between the variables, which can help the results interpretation.

A Partition of variables

This relationship can be represented with a partition-based clustering, as illustrated in Fig. 5.5. In this example, an MDS projection is used based on the combination of two dissimilarities: the row-partition dissimilarity, and the time series distance.

This representation can help interpreting the contents of the column-clusters (e.g., associate a column-cluster to Ego's direction, or lateral position).

However, the weights of this combination have been set manually, which is not a straightforward task. In the following we consider developing alternative representations of the variable.

One interesting candidate can be the *dandelion* plot, that allows a multi-level categorical grouping based on hierarchical dissimilarities. It is also possible to simply ignore the partition similarity information and construct the row-partitions on the only basis of the distributions.

Choosing the row-partition ordering

In the coclustering framework, the user is interested in visualizing the inferred block structure, which is obtained by re-ordering the dataset matrix by group. For instance, this re-ordering produces the grid visualization shown in Fig. 1.13 b), or in Fig. 3.2 a).

With MCC (and more generally with the Multi-Clustering methods), the re-ordering of the rows is performed hierarchically: based on $\mathbf{z}^1$, then $\mathbf{z}^2$ ... until $\mathbf{z}^H$. Because it is hierarchical, this re-ordering depends on the redundant cluster order (e.g., swapping $\mathbf{z}^1$ and $\mathbf{z}^2$ will modify it). However, these different visualisations are not equivalent to the user, because some row-partitions are more interesting than the others. In addition, several partitions are *included* into others, which means that they are the result of a hierarchical subdivision, similar to to a hierarchical clustering dendrogram. Choosing this ordering manually is not straightforward (especially when the number of redundant clusters H is large) and, in the current implementation of FunMCC, depends on expert knowledge. An interesting extension would consist in inferring automatically this ordering. For instance, assuming that some row-partitions are nested into each others, a tree representation might be able to represent the dependencies between row-partitions. A score that measures the mutual partition information (e.g., NMI, ARI, ...) or the partition inclusion seems an interesting candidate for this construction.

This hierarchical representation adds an interesting layer of information: the dependency between row-partitions.

6.2.3 Time Series Representation Alternatives

In this thesis contributions, the time series are transformed with a PCA on the coefficient of an Interpolated Log-Periodogram representations. Alternative time series representations can be considered, with a strong incentive on keeping a final representation dimension low (cf. inference complexity description in Sec. 5.2.4).

First, it is possible to replace the trigonometric function basis by another polynomial basis (e.g., Legendre, ..). It is also possible to represent the time series with a wavelet basis function decomposition, and to apply a PCA on the resulting coefficients. This method, presented in [RW13], has the advantage to represent the time series in both frequency and time domain, and could be an interesting alternative.

It is also possible to avoid the functional basis decomposition approach. If the user is interested in grouping time series on the basis of a given similarity measure, the spectral dimension reduction is an interesting candidate. This approach consists in computing the Laplacian of the time series similarity matrix and use its eigenvectors as time series features. With this approach, any distance can be considered (e.g., DTW, Mahalanobis, ...). Other ways to make use of the dissimilarity matrix for time series representation include the Multi-Dimensional Scaling, which is similar to performing a PCA on the dissimilarity matrix. The results is a linear transformation in a smaller space, that conserves the similarity information (i.e., two elements close in the final space are also close in the original space). This transformation can also be performed with auto-encoder, or t-SNE. However, this set of approaches may be computationally expansive, as they require to compute $\frac{np(np-1)}{2}$ similarities before the block-clustering inference.

Inspired from [Kat16], it is also possible to represent a time series with its dissimilarity to other *reference* elements. By using a small number of elements (for instance a ratio specified by the user), this method is equivalent to sub-sampling the dissimilarity matrix, and results in a lower dimensional feature basis representation. Again, a dimension reduction step may be useful if this feature space remains excessively large. This approach seems promising, but requires to correctly choose the reference time series.

6.2.4 Alternative Bayesian Non Parametric Prior

The DPMM framework used throughout this thesis is an handy solution to mixture model indeference and is one of the most popular BNP mixture model.

However, the DPMM has several drawbacks:

- The DP prior tends to create partitions with few large clusters and several small ones, which is not always desirable.
- Even though the DPMM consistently estimates the mixture density [GV07] and the mixing distributions [Ngu13], it has been shown in [MH13] that the method does not consistently converges to the true number of latent components.

The first limitation can be addressed with the introduction of a Pitman-Yor Process prior in place of the DP. This process uses a discount parameter to leverage the creation of equal-size clusters. However, it has also been shown in [MH14] that this method still inconsistently estimates the number of components.

A recent proposal [Mil19] introduces the Mixture of Finite Mixtures model, that is consistent for the number of components. This model is similar to the DPMM, but adds an explicit definition of a prior directly on the component number K. This prior distribution is only a consequence of the stick-breaking construction with the DP prior (cf. Eq. 1.6 in Sect. 1.3.2). The same paper presents the associated inference methods, that are similar to the existing DPMM inference methods.

Based on this thesis contributions, extending the Mixture of Finite Mixtures to a co-clustering, multi-clustering and multi-coclustering framework seems straightforward, and would allow a better control of the number of components.

6.3 Conclusion

In the broad and wide domain of time series clustering, this thesis presented several contributions specifically designed for the exploration of time series datasets generated for ADAS validation.

The first contribution addresses univariate time series datasets, and assumes the presence of latent scenario. This method uses the mixture model framework for time series segmentation and driving pattern detection, and the hierarchical clustering heuristic for the final clustering of time series re-coded as symbol sequences.

The next three contributions are block clustering methods that make full use of the model-based coclustering and multi-clustering methods. These methods address the clustering of MTS based on a Fourier representation of the time series. The use

of the BNP framework allow to integrate a model selection step while keeping the anomaly detection and prediction interval features.

These methods are experimented on simulated datasets, but also applied on real-life datasets generated for ADAS validation. These applications proved the contributions interest for industrial purposes. The methods source code is open and made available at github repositories for reproducibility.

This thesis, staged in convention between the LIPN at Université Sorbonne Paris Nord and Groupe Renault, opens several perspectives of extension, both for the dictionary-based and the block-clustering-based methods.

Glossary

ADAS	Advanced Driver Assistance System	AEB	Autonomous Emergency Braking
AIC	Akaike Information Criterion	ARI	Adjusted Rand Index
BIC	Bayesian Information Criterion	BNP	Bayesian Non Parametric
CLBM	Conditional LBM	CRP	Chinese Restaurant Process
DP	Dirichlet Process	DPM	Dirichlet Process Mixture
DTW	Dynamic Time Warping	DWT	Discrete WaveletTransform
Ego	Vehicle under test	ELK	Emergency Lane Keeping
-/C/S EM	-/Classification/Stochastic Expectation Maximization	fPCA	Functional Principal Component Analysis
GMM	Gaussian Mixture Model	HMM	Hidden Markov Model
ICL	Integrated Completed Likelihood	LBM	latent Block Model
LKA	Lane Keeping Assist	MCC	Multi-Coclustering
MCMC	Markov Chain Monte-Carlo	MPM	Multiple Partitions Model
MTS	Multivariate Time Series	NIW	Normal Inverse Wishart
NMI	Normalized Mutual Information	NN	Neural Network
NPLBM	Non-Parametric LBM	PAA	Piecewise Aggregate Approximation
PCA	Principal Component Analysis	PRM	Polynomial Regression Model
PYP	Pitman-Yor Process	SAE	Society of Automotive Engineers
SB	Stick-Breaking	SDLHC	Segmentation-Dictionary-Levenshtein-Hierarchical-Clustering
SOM	Self Organizing Map	SVD	Singular Value Decomposition
SAX	Symbolic Aggregate approXimation	RHLP	Regression with Hidden Logistic Process
RI	Rand Index	SNE	Stochastic Neighbor Embedding
UMAP	Uniform Manifold Approximation and Projection	WLD	Weighted Levenshtein Distance

List of Figures

List of Tables

Appendix

A

A.1 Prior Predictive Distribution

The proof that the predictive distribution densities are equivalent to a multivariate t-distribution is rarely presented. The most cited source is [Mur07] (e.g., cited in [VDR14; Das14]) proves the equivalence in the univariate case. The predictive distribution closed-form being a key feature of our Collapsed Gibbs Sampler inference processes, we propose a detailed derivation of the multivariate distribution.

Lemma 1. *The probability density function of the multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$ can be written as*

$$\mathrm{p}(y \mid \eta) = h(y) \exp\left(\eta^T T(y) - \mathbb{1}^T g(\eta)\right),$$

Proof. First, observe that $y^T\Sigma^{-1}y = \mathrm{Tr}(\Sigma^{-1}yy^T) = \mathrm{vec}(\Sigma^{-1})^T\,\mathrm{vec}(yy^T)$. As a consequence, using the usual definition of the density function yields

$$\begin{aligned}
\mathrm{p}(y \mid \mu, \Sigma) &= (2\pi)^{-\frac{d}{2}}|\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(y-\mu)^T\Sigma^{-1}(y-\mu)\right) \\
&= (2\pi)^{-d/2} \exp\left(\begin{bmatrix}\Sigma^{-1}\mu \\ -\frac{1}{2}\,\mathrm{vec}(\Sigma^{-1})\end{bmatrix}^T \begin{bmatrix} y \\ \mathrm{vec}(yy^T)\end{bmatrix} - \frac{1}{2}\mu^T\Sigma^{-1}\mu - \frac{1}{2}\log|\Sigma|\right) \\
&= h(y)\ \exp\left(\eta^T T(y) - \mathbb{1}^T g(\eta)\right)
\end{aligned}$$

where

$$h(y) = (2\pi)^{-\frac{d}{2}}, \eta = \begin{bmatrix}\Sigma^{-1}\mu \\ -\frac{1}{2}\,\mathrm{vec}(\Sigma^{-1})\end{bmatrix}, g(\eta) = \begin{bmatrix}\frac{1}{2}\mu^T\Sigma^{-1}\mu \\ \frac{1}{2}log|\Sigma|\end{bmatrix}, T(y) = \begin{bmatrix} y \\ \mathrm{vec}(yy^T)\end{bmatrix},$$

and vec is the vectorization operator. □

Lemma 2. *The probability density function of the Normal Inverse Wishart distribution $NIW(\mu, \Sigma \mid \xi)$, with $\xi = (\mu_0, \kappa_0, \Sigma_0, \nu_0)$ can be written as*

$$\mathrm{p}(\mu, \Sigma \mid \Psi, \nu) = f(\Psi, \nu) \exp\left(\eta^T\Psi - \nu^T g(\eta)\right).$$

Proof.

$$
\begin{aligned}
&\mathrm{p}(\mu, \Sigma \mid \xi) \\
&= \frac{\kappa_0^{d/2}|\Psi_0|^{\frac{\nu_0}{2}}|\Sigma|^{-\frac{\nu_0+d+2}{2}}}{2^{\frac{\nu_0 d}{2}}\pi^{d/2}\Gamma_d(\frac{\nu_0}{2})} \exp\left(-\frac{\kappa_0}{2}(\mu-\mu_0)^T\Sigma^{-1}(\mu-\mu_0) - \frac{1}{2}\operatorname{Tr}(\Psi_0\Sigma^{-1})\right) \\
&= \frac{\kappa_0^{d/2}|\Psi_0|^{\frac{\nu_0}{2}}}{2^{\frac{\nu_0 d}{2}}\pi^{d/2}\Gamma_d\left(\frac{\nu_0}{2}\right)} \exp\left(\eta^T\begin{bmatrix}\kappa_0\mu_0 \\ \operatorname{vec}(\kappa_0\mu_0\mu_0^T + \Psi_0)\end{bmatrix} - \begin{bmatrix}\kappa_0 \\ \nu_0+d+2\end{bmatrix}^T g(\eta)\right) \\
&= f(\Psi,\nu)\ \exp\left(\eta^T\Psi - \nu^T g(\eta)\right)
\end{aligned}
$$

with $f(\Psi,\nu) = \frac{\kappa_0^{d/2}|\Psi_0|^{\frac{\nu_0}{2}}}{2^{\frac{\nu_0 d}{2}}\pi^{d/2}\Gamma_d(\frac{\nu_0}{2})}$, $\Psi = \begin{bmatrix}\kappa_0\mu_0 \\ \operatorname{vec}(\kappa_0\mu_0\mu_0^T + \Psi_0)\end{bmatrix}$ and $\nu = \begin{bmatrix}\kappa_0 \\ \nu_0+d+2\end{bmatrix}$. □

Theorem 1. *The prior predictive distribution of a random variable y drawn according to*

$$y \sim \mathcal{N}(\mu,\Sigma), \quad (\mu,\Sigma) \sim NIW(\xi), \quad \xi = (\mu_0, \kappa_0, \Sigma_0, \nu_0),$$

is the multivariate t-distribution

$$y \mid \xi \sim t_{\nu_0-d+1}\left(y|\mu_0, \frac{(\kappa_0+1)\Psi_0}{\kappa_0(\nu_0-d+1)}\right).$$

Proof. According to Lemmas 1 and 2, we find that

$$
\begin{aligned}
\mathrm{p}(y \mid \Psi,\nu) &= \int_{\mu,\Sigma} \mathrm{p}(y \mid \mu,\Sigma)\mathrm{p}(\mu,\Sigma \mid \Psi,\nu)d(\mu,\Sigma) \\
&= \int_\eta h(y)\exp\left(\eta^T T(y) - \mathbb{1}^T g(\eta)\right) f(\Psi,\xi)\exp\left(\eta^T\Psi - \nu^T g(\eta)\right)d\eta \\
&= \int_\eta h(y)\exp\left(\eta^T T(y) - \mathbb{1}^T g(\eta)\right) f(\Psi,\xi)\exp\left(\eta^T\Psi - \nu^T g(\eta)\right)d\eta \\
&= h(y)f(\Psi,\nu)\int_\eta \exp\left(\eta^T(\Psi+T(y)) - (\nu+\mathbb{1})^T g(\eta)\right)d\eta.
\end{aligned}
$$

The integral part is obtained by considering the Normal Inverse Wishart density with hyper-parameter $(\Psi+T(y), \nu+\mathbb{1})$:

$$
\begin{aligned}
&\mathrm{p}(\mu,\Sigma \mid \Psi+T(y),\nu+\mathbb{1}) = f(\Psi+T(y),\nu+\mathbb{1})\exp\left(\eta^T(\Psi+T(y)) - (\nu+\mathbb{1})^T g(\eta)\right) \\
\Rightarrow \quad &\int f(\Psi+T(y),\nu+\mathbb{1})\exp\left(\eta^T(\Psi+T(y)) - (\nu+\mathbb{1})^T g(\eta)\right)d\eta = 1 \\
\Rightarrow \quad &\int \exp\left(\eta^T(\Psi+T(y)) - (\nu+\mathbb{1})^T g(\eta)\right)d\eta = \frac{1}{f(\Psi+T(y),\nu+\mathbb{1})} \qquad \text{(A.1)}
\end{aligned}
$$

This last expression yields the following equality

$$\mathrm{p}(y \mid \Psi, \nu) = h(y) \frac{f(\Psi, \nu)}{f(\Psi + T(y), \nu + \mathbb{1})},$$

where $\Psi + T(y) = \begin{bmatrix} \kappa_1 \mu_1 \\ \mathrm{vec}(\kappa_1 \mu_1 \mu_1^T + \Psi_1) \end{bmatrix} = \begin{bmatrix} \kappa_0 \mu_0 \\ \mathrm{vec}(\kappa_0 \mu_0 \mu_0^T + \Psi_0) \end{bmatrix} + \begin{bmatrix} y \\ \mathrm{vec}(yy^T) \end{bmatrix}$, and $\nu + \mathbb{1} = \begin{bmatrix} \kappa_1 \\ \nu_1 + d + 2 \end{bmatrix} = \begin{bmatrix} \kappa_0 \\ \nu_0 + d + 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Introducing these parameters into the last expression of $\mathrm{p}(y \mid \xi)$ gives the result of the lemma.

$$\mathrm{p}(y \mid \Psi, \nu) = \pi^{-d/2} \cdot \frac{\kappa_0^{d/2}}{\kappa_1^{d/2}} \cdot \frac{\Gamma_d(\nu_1/2)}{\Gamma_d(\nu_0/2)} \cdot \frac{|\Psi_0|^{\nu_0/2}}{|\Psi_1|^{\nu_1/2}}, \tag{A.2}$$

On one hand, $\Gamma_d(\nu_1/2)/\Gamma_d(\nu_0/2)$ simplifies using the definition of the multivariate Gamma function $\Gamma_d(a) = \pi^{d(d-1)/4} \prod_{j=1}^{d} \Gamma(a + (1-j)/2)$ yielding

$$\frac{\Gamma_d\left(\frac{\nu_1}{2}\right)}{\Gamma_d\left(\frac{\nu_0}{2}\right)} = \frac{\Gamma_d\left(\frac{\nu_0+1}{2}\right)}{\Gamma_d\left(\frac{\nu_0}{2}\right)} = \prod_{j=1}^{d} \frac{\Gamma\left(\frac{\nu_0+2-j}{2}\right)}{\Gamma\left(\frac{\nu_0+1-j}{2}\right)} = \frac{\Gamma\left(\frac{\nu_0+1}{2}\right)}{\Gamma\left(\frac{\nu_0-d+1}{2}\right)}.$$

On the other hand, $|\Psi_0|^{\nu_0/2}/|\Psi_1|^{\nu_1/2}$ reduces because

$$\begin{aligned}
\Psi_1 &= \Psi_0 + yy^T + \kappa_0 \mu_0 \mu_0^T - \kappa_1 \mu_1 \mu_1^T \\
&= \Psi_0 + yy^T + \kappa_0 \mu_0 \mu_0^T - (\kappa_0 + 1) \left(\frac{\kappa_0 \mu_0 + y}{\kappa_0 + 1}\right) \left(\frac{\kappa_0 \mu_0 + y}{\kappa_0 + 1}\right)^T \\
&= \Psi_0 + \frac{\kappa_0 + 1}{\kappa_0 + 1} yy^T + \frac{\kappa_0(\kappa_0 + 1)}{\kappa_0 + 1} \mu_0 \mu_0^T - \left(\frac{1}{\kappa_0 + 1}\right) \left(\kappa_0^2 \mu_0 \mu_0^T + \kappa_0 \mu_0 y^T + \kappa_0 y \mu_0^T + yy^T\right) \\
&= \Psi_0 + \frac{\kappa_0}{\kappa_0 + 1} (y - \mu_0)(y - \mu_0)^T \\
&= \Psi_0 \left(1 + \frac{\kappa_0}{\kappa_0 + 1} (y - \mu_0)^T \Psi_0^{-1} (y - \mu_0)\right)
\end{aligned}$$

and, using the matrix determinant lemma (i.e. $|A + uv^T| = (1 + v^T A^{-1} u)|A|$) gives

$$|\Psi_1| = \left(1 + \frac{\kappa_0}{\kappa_0 + 1} (y - \mu_0^T) \Psi_0^{-1} (y - \mu_0^T)\right) |\Psi_0|.$$

Substituting these expressions into Eq. (A.2) gives the density

$$\mathrm{p}(y \mid \Psi, \nu) = \frac{\Gamma\left(\frac{\nu_0+1}{2}\right)}{\left(\frac{\kappa_0+1}{\kappa_0}\right)^{d/2} \pi^{d/2} \Gamma\left(\frac{\nu_0-d+1}{2}\right) |\Psi_0|^{1/2}} \left(1 + \frac{\kappa_0}{\kappa_0 + 1} (y - \mu_0)^T \Psi_0^{-1} (y - \mu_0)\right)^{-(\nu_0+1)/2},$$

which is the density of a multivariate t-distribution $t_{\nu_0-d+1}\left(y | \mu_0, \frac{(\kappa_0+1)\Psi_0}{\kappa_0(\nu_0-d+1)}\right)$. □

A.2 Posterior Predictive Distribution

With the same notation, and denoting $\mathbf{x} = (x_i)_n$ a set of observations, the posterior distribution follows a NIW distribution [Ben+19; Mur07]:

$$\mathrm{p}(\mu, \Sigma \mid \mathbf{x}, \mu_0, \kappa_0, \Sigma_0, \nu_0) = NIW(\mu, \Sigma \mid \mu_n, \kappa_n, \Sigma_n, \nu_n),$$

with the updated hyper-parameter values obtained by:

$$\begin{aligned}
\mu_n &= \frac{\kappa_0 \mu_0 + n\bar{x}}{\kappa_0 + n}, \\
\kappa_n &= \kappa_0 + n, \nu_n = \nu_0 + n, \\
\Psi_n &= \Psi_0 + \sum_i x_i x_i^T + \kappa_0 \mu_0 \mu_0^T - \kappa_n \mu_n \mu_n^T.
\end{aligned}$$

With $\Psi' = \begin{bmatrix} \kappa_n \mu_n \\ \mathrm{vec}(\kappa_n \mu_n \mu_n^T + \Psi_n) \end{bmatrix}$ and $\nu' = \begin{bmatrix} \kappa_n \\ \nu_n + d + 2 \end{bmatrix}$, the posterior predictive distribution can be written as:

$$\begin{aligned}
\mathrm{p}(y \mid \Psi', \nu') &= \int_{\mu,\Sigma} \mathrm{p}(y \mid \mu, \Sigma) \mathrm{p}(\mu, \Sigma \mid \psi', \nu') d(\mu, \Sigma) \\
&= \int_\eta h(y) \exp\left(\eta^T T(y) - \mathbb{1}^T g(\eta)\right) f(\Psi', \nu') \exp\left(\eta^T \Psi' - \nu'^T g(\eta)\right) d\eta \\
&= h(y) f(\Psi', \nu') \int_\eta \exp\left[\eta^T (\Psi' + T(y)) - (\nu' + \mathbb{1})^T g(\eta)\right] d\eta.
\end{aligned}$$

After integral simplification (c.f. Eq. A.1), the posterior can be written as:

$$\mathrm{p}(y \mid \mathbf{x}, \Psi, \nu) = h(y) \frac{f(\Psi', \nu')}{f(\Psi' + T(y), \nu' + \mathbb{1})}. \tag{A.3}$$

These expressions are similar to the ones of the prior distribution, with updated hyper-parameter values, with $\Psi' + T(y) = \begin{bmatrix} \kappa_1 \mu_1 \\ \mathrm{vec}(\kappa_1 \mu_1 \mu_1^T + \Psi_1) \end{bmatrix} = \begin{bmatrix} \kappa_n \mu_n \\ \mathrm{vec}(\kappa_n \mu_n \mu_n^T + \Psi_n) \end{bmatrix} + \begin{bmatrix} y \\ \mathrm{vec}(yy^T) \end{bmatrix}$, and $\nu' + \mathbb{1} = \begin{bmatrix} \kappa_1 \\ \nu_1 + d + 2 \end{bmatrix} = \begin{bmatrix} \kappa_n \\ \nu_n + d + 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. As in the prior predictive case, injecting the updated hyper-parameters values into Eq. A.3 gives the following expression:

$$\mathrm{p}(y \mid \mathbf{x}, \Psi, \nu) = \pi^{-d/2} \cdot \frac{\kappa_n^{d/2}}{\kappa_1^{d/2}} \cdot \frac{\Gamma_d(\nu_1/2)}{\Gamma_d(\nu_n/2)} \cdot \frac{|\Psi_n|^{\nu_n/2}}{|\Psi_1|^{\nu_1/2}},$$

which is the density of the multivariate t-distribution $t_{\nu_n-d+1}\left(y|\mu_n, \frac{(\kappa_n+1)\Psi_n}{\kappa_n(\nu_n-d+1)}\right)$

A.3 Joint Predictive Distribution

In the Bayesian Coclustering or Multi-Coclustering, the inference algorithm uses the (prior or posterior) predictive distribution of a row, i.e., a joint predictive distribution. This distribution also has a closed-form when the prior is conjugate. In the joint prior predictive case, the distribution of a vector of observations $\mathbf{y} = (y_i)_m$ is:

$$\begin{aligned}
\mathrm{p}(\mathbf{y} \mid \Psi, \nu) &= \int_{\mu,\Sigma} \mathrm{p}(\mathbf{y}, \mu, \Sigma \mid \Psi, \nu) d(\mu, \Sigma) \\
&= \int_{\mu,\Sigma} \mathrm{p}(\mathbf{y} \mid \mu, \Sigma) \mathrm{p}(\mu, \Sigma \mid \psi, \nu) d\,(\mu, \Sigma) \\
&= \int_{\mu,\Sigma} \prod_i \mathrm{p}(y_i \mid \mu, \Sigma) \mathrm{p}(\mu, \Sigma \mid \psi, \nu) d\,(\mu, \Sigma) \\
&= \int_{\eta} \prod_i \left[h(y_i) \exp\left(\eta^T T(y_i) - \mathbb{1} g(\eta)\right)\right] f(\Psi, \xi) \exp\left(\eta^T \Psi - \nu^T g(\eta)\right) d\eta \\
&= \prod_i h(y_i) f(\Psi, \xi) \int_{\eta} \exp\left[\eta^T (\Psi + T(\mathbf{y})) - (\nu + m\mathbb{1})^T g(\eta)\right] d\eta,
\end{aligned}$$

with $T(\mathbf{y}) = \sum_i T(y_i)$. After simplifying the integral (c.f. Eq. A.1), the posterior can be written as:

$$\mathrm{p}(\mathbf{y} \mid \Psi, \nu) = \prod_i h(y_i) \frac{f(\Psi, \nu)}{f(\Psi + T(\mathbf{y}), \nu + m\mathbb{1})}, \tag{A.4}$$

with $\Psi + T(\mathbf{y}) = \begin{bmatrix} \kappa_m \mu_m \\ \mathrm{vec}(\kappa_m \mu_m \mu_m^T + \Psi_m) \end{bmatrix} = \begin{bmatrix} \kappa_0 \mu_0 \\ \mathrm{vec}(\kappa_0 \mu_0 \mu_0^T + \Psi_0) \end{bmatrix} + \begin{bmatrix} \sum_i y_i \\ \mathrm{vec}(\sum_i y_i y_i^T) \end{bmatrix}$,
and $\nu + m\mathbb{1} = \begin{bmatrix} \kappa_m \\ \nu_m + d + 2 \end{bmatrix} = \begin{bmatrix} \kappa_0 \\ \nu_0 + d + 2 \end{bmatrix} + \begin{bmatrix} m \\ m \end{bmatrix}$, which yields the same hyper-parameters update expressions than in the posterior predictive case detailed in the previous section Sect. A.2.

Injecting the values of h, f and the prior hyper-parameters in Eq. A.4 gives the closed-form of the joint prior predictive distribution:

$$\mathrm{p}(\mathbf{y} \mid \Psi, \nu) = \pi^{\frac{-md}{2}} \frac{\kappa_0^{d/2}}{\kappa_m^{d/2}} \cdot \frac{\Gamma_d(\nu_m/2)}{\Gamma_d(\nu_0/2)} \cdot \frac{|\Psi_0|^{\nu_0/2}}{|\Psi_m|^{\nu_m/2}}.$$

Distribution	Expression	Closed-form
prior predictive	$\mathrm{p}(y \mid \Psi, \nu)$	$\pi^{-d/2} \cdot \frac{\kappa_0^{d/2}}{\kappa_1^{d/2}} \cdot \frac{\Gamma_d(\nu_1/2)}{\Gamma_d(\nu_0/2)} \cdot \frac{\lvert\Psi_0\rvert^{\nu_0/2}}{\lvert\Psi_1\rvert^{\nu_1/2}}$
posterior predictive	$\mathrm{p}(y \mid \mathbf{x}, \Psi, \nu)$	$\pi^{-d/2} \cdot \frac{\kappa_n^{d/2}}{\kappa_1^{d/2}} \cdot \frac{\Gamma_d(\nu_1/2)}{\Gamma_d(\nu_n/2)} \cdot \frac{\lvert\Psi_n\rvert^{\nu_n/2}}{\lvert\Psi_1\rvert^{\nu_1/2}}$
joint prior predictive	$\mathrm{p}(\mathbf{y} \mid \Psi, \nu)$	$\pi^{\frac{-md}{2}} \cdot \frac{\kappa_0^{d/2}}{\kappa_m^{d/2}} \cdot \frac{\Gamma_d(\nu_m/2)}{\Gamma_d(\nu_0/2)} \cdot \frac{\lvert\Psi_0\rvert^{\nu_0/2}}{\lvert\Psi_m\rvert^{\nu_m/2}}$
joint posterior predictive	$\mathrm{p}(\mathbf{y} \mid \mathbf{x}, \Psi, \nu)$	$\pi^{\frac{-md}{2}} \cdot \frac{\kappa_n^{d/2}}{\kappa_m^{d/2}} \cdot \frac{\Gamma_d(\nu_m/2)}{\Gamma_d(\nu_n/2)} \cdot \frac{\lvert\Psi_n\rvert^{\nu_n/2}}{\lvert\Psi_m\rvert^{\nu_m/2}}$

Tab. A.1.: Summary of predictive distributions closed forms.

The same derivations yields the following joint posterior predictive distribution expressions:

$$\mathrm{p}(\mathbf{y} \mid \mathbf{x}, \Psi, \nu) = \prod_i h(y_i) \frac{f(\Psi', \nu')}{f(\Psi' + T(\mathbf{y}), \nu' + m\mathbb{1})},$$

that reduces to the following closed-form:

$$\pi^{\frac{-md}{2}} \cdot \frac{\kappa_n^{d/2}}{\kappa_m^{d/2}} \cdot \frac{\Gamma_d(\nu_m/2)}{\Gamma_d(\nu_n/2)} \cdot \frac{\lvert\Psi_n\rvert^{\nu_n/2}}{\lvert\Psi_m\rvert^{\nu_m/2}}.$$

The prior and posterior predictive distributions closed forms are summarized in Tab. A.1.

Bibliography

[AML19] Amaia Abanda, Usue Mori, and Jose A Lozano. “A review on distance based time series classification”. In: *Data Mining and Knowledge Discovery* 33.2 (2019), pp. 378–412 (cit. on p. 28).

[ASW15] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. “Time-series clustering–a decade review”. In: *Information Systems* 53 (2015), pp. 16–38 (cit. on pp. 27, 28, 33).

[AD15] HP Ahmad and Shilpa Dang. “Performance Evaluation of Clustering Algorithm Using different dataset”. In: *International Journal of Advance Research in Computer Science and Management Studies* 8 (2015) (cit. on p. 33).

[Aka74] H. Akaike. “A new look at the statistical model identification”. In: *IEEE Transactions on Automatic Control* 19.6 (1974), pp. 716–723 (cit. on p. 44).

[AT15] Duong Tuan Anh and Le Huu Thanh. “An efficient implementation of k-means clustering for time series data with DTW distance”. In: *International Journal of Business Intelligence and Data Mining* 10.3 (2015), pp. 213–232 (cit. on p. 38).

[Ant74] Charles E Antoniak. “Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems”. In: *The annals of statistics* (1974), pp. 1152–1174 (cit. on pp. 48, 49).

[BJ05] Anthony Bagnall and Gareth Janacek. “Clustering time series with clipped data”. In: *Machine learning* 58.2-3 (2005), pp. 151–178 (cit. on p. 39).

[Bai+15] Adeline Bailly, Simon Malinowski, Romain Tavenard, Thomas Guyet, and Laetitia Chapel. “Bag-of-temporal-sift-words for time series classification”. In: *ECML/PKDD workshop on advanced analytics and learning on temporal data*. 2015 (cit. on p. 29).

[BC15] Jean-Patrick Baudry and Gilles Celeux. “EM for mixtures”. In: *Statistics and computing* 25.4 (2015), pp. 713–726 (cit. on p. 42).

[Beg+15] Nurjahan Begum, Liudmila Ulanova, Jun Wang, and Eamonn Keogh. “Accelerating dynamic time warping clustering with a novel admissible pruning strategy”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, pp. 49–58 (cit. on p. 34).

[Ben+19] Eric Benhamou, David Saltiel, Beatrice Guez, and Nicolas Paris. “Bcma-es ii: revisiting bayesian cma-es”. In: (2019) (cit. on p. 162).

[BB12] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization.” In: *Journal of machine learning research* 13.2 (2012) (cit. on p. 44).

[BCG00] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. “Assessing a mixture model for clustering with the integrated completed likelihood”. In: *IEEE transactions on pattern analysis and machine intelligence* 22.7 (2000), pp. 719–725 (cit. on pp. 44, 89, 116).

[BCG03] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. “Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models”. In: *Computational Statistics & Data Analysis* 41.3-4 (2003), pp. 561–575 (cit. on pp. 42, 116).

[Bis06] Christopher M Bishop. “Pattern recognition”. In: *Machine learning* 128.9 (2006), p. 3 (cit. on pp. 26, 41, 42).

[BJ+06] David M Blei, Michael I Jordan, et al. “Variational inference for Dirichlet process mixtures”. In: *Bayesian analysis* 1.1 (2006), pp. 121–143 (cit. on pp. 49, 124).

[BPZ04] Stefan Bleuler, Amela Prelic, and Eckart Zitzler. “An EA framework for biclustering of gene expression data”. In: *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*. Vol. 1. IEEE. 2004, pp. 166–173 (cit. on p. 60).

[BB13] Johannes Blömer and Kathrin Bujna. “Simple methods for initializing the em algorithm for gaussian mixture models”. In: *CoRR* (2013) (cit. on pp. 42, 75).

[Bol+08] Tim Bollerslev et al. “Glossary to arch (garch)”. In: *CREATES Research paper* 49 (2008), pp. 1–46 (cit. on p. 31).

[Bou12] Marc Boullé. “Functional data clustering via piecewise constant nonparametric density estimation”. In: *Pattern Recognition* 45.12 (2012), pp. 4389–4401 (cit. on p. 39).

[Bou+18] Charles Bouveyron, Laurent Bozzi, Julien Jacques, and François-Xavier Jollois. “The functional latent block model for the co-clustering of electricity consumption curves”. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 67.4 (2018), pp. 897–915 (cit. on pp. 63, 86, 89, 94, 95, 107, 116).

[BJ11] Charles Bouveyron and Julien Jacques. “Model-based clustering of time series in group-specific functional subspaces”. In: *Advances in Data Analysis and Classification* 5.4 (2011), pp. 281–300 (cit. on p. 39).

[Bou+20] Charles Bouveyron, Julien Jacques, Amandine Schmutz, Fanny Simoes, and Silvia Bottini. “Co-Clustering of Multivariate Functional Data for the Analysis of Air Pollution in the South of France”. In: (2020) (cit. on pp. 63, 107).

[Bra14] Vincent Brault. “Estimation et sélection de modèle pour le modèle des blocs latents”. PhD thesis. Université Paris Sud-Paris XI, 2014 (cit. on p. 63).

[BL15] Vincent Brault and Aurore Lomet. “Methods for co-clustering: a review”. In: *Journal de la Société Française de Statistique* 156.3 (2015), pp. 27–51 (cit. on p. 59).

[BNS11] Krisztian Buza, Alexandros Nanopoulos, and Lars Schmidt-Thieme. “Fusion of similarity measures for time series classification”. In: *International Conference on Hybrid Artificial Intelligence Systems*. Springer. 2011, pp. 253–261 (cit. on p. 28).

[CC07] Jorge Caiado and Nuno Crato. “A GARCH-based method for clustering of financial time series: International stock markets evidence”. In: *Recent Advances in Stochastic Modeling and Data Analysis*. World Scientific, 2007, pp. 542–551 (cit. on p. 31).

[CCP09] Jorge Caiado, Nuno Crato, and Daniel Peña. “Comparison of times series with unequal length in the frequency domain”. In: *Communications in Statistics—Simulation and Computation®* 38.3 (2009), pp. 527–540 (cit. on pp. 31, 86).

[CHZ15] Yujie Cao, Minlie Huang, and Xiaoyan Zhu. “Clustering sentiment phrases in product reviews by constrained co-clustering”. In: *Natural Language Processing and Chinese Computing*. Springer, 2015, pp. 79–89 (cit. on p. 59).

[Cas+21] Alessandro Casa, Charles Bouveyron, Elena Erosheva, and Giovanna Menardi. *Co-clustering of time-dependent data via Shape Invariant Model*. 2021. arXiv: `2104.03083 [stat.ME]` (cit. on pp. 63, 107).

[CD85] G. Celeux and J. Diebolt. “The SEM Algorithm: A probabilistic teacher algorithm derived from the EM algorithm for the mixture problem”. In: *Computational Statistics Quarterly* 2 (1985), pp. 73–82 (cit. on p. 42).

[CD92] Gilles Celeux and Jean Diebolt. “A stochastic approximation type EM algorithm for the mixture problem”. In: *Stochastics: An International Journal of Probability and Stochastic Processes* 41.1-2 (1992), pp. 119–134 (cit. on p. 42).

[CFR18] Gilles Celeux, Sylvia Frühwirth-Schnatter, and Christian P Robert. “Model selection for mixture models—perspectives and strategies”. In: *Handbook of mixture analysis* (2018), pp. 121–160 (cit. on p. 107).

[Cha16] Faicel Chamroukhi. “Unsupervised learning of regression mixture models with unknown number of components”. In: *Journal of Statistical Computation and Simulation* 86.12 (2016), pp. 2308–2334 (cit. on p. 39).

[CB17] Faicel Chamroukhi and Christophe Biernacki. “Model-based co-clustering of multivariate functional data”. In: *ISI 2017-61st World Statistics Congress*. 2017 (cit. on pp. 63, 107, 115).

[Cha+13] Faicel Chamroukhi, Samer Mohammed, Dorra Trabelsi, Latifa Oukhellou, and Yacine Amirat. “Joint segmentation of multivariate time series with hidden process regression for human activity recognition”. In: *Neurocomputing* 120 (2013), pp. 633–644 (cit. on pp. 57, 63, 141).

[CN18] Faicel Chamroukhi and Hien D Nguyen. “Model-based clustering and classification of functional data”. In: *arXiv preprint arXiv:1803.00276* (2018) (cit. on p. 39).

[Cha+10] Faicel Chamroukhi, Allou Samé, Gérard Govaert, and Patrice Aknin. “A hidden process regression model for functional data description. application to curve discrimination”. In: *Neurocomputing* 73.7-9 (2010), pp. 1210–1221 (cit. on p. 69).

[Cha+09a] Faicel Chamroukhi, Allou Samé, Gérard Govaert, and Patrice Aknin. “A regression model with a hidden logistic process for feature extraction from time series”. In: *2009 International Joint Conference on Neural Networks*. IEEE. 2009, pp. 489–496 (cit. on p. 73).

[Cha+09b] Faicel Chamroukhi, Allou Samé, Gérard Govaert, and Patrice Aknin. “Time series modeling by a regression approach based on a latent process”. In: *Neural Networks* 22.5-6 (2009), pp. 593–602 (cit. on pp. 69, 71, 141).

[CS08] S Chandrakala and C Chandra Sekhar. “A density based method for multivariate time series clustering in kernel feature space”. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE. 2008, pp. 1885–1890 (cit. on p. 58).

[Cha+18] Marie Chavent, Vanessa Kuentz-Simonet, Amaury Labenne, and Jérôme Saracco. “ClustGeo: an R package for hierarchical clustering with spatial constraints”. In: *Computational Statistics* 33.4 (2018), pp. 1799–1822 (cit. on p. 36).

[CS75] Václáv Chvatal and David Sankoff. “Longest common subsequences of two random sequences”. In: *Journal of Applied Probability* 12.2 (1975), pp. 306–315 (cit. on p. 28).

[CL15] Etienne Côme and Pierre Latouche. “Model selection and clustering in stochastic block models based on the exact integrated complete data likelihood”. In: *Statistical Modelling* 15.6 (2015), pp. 564–589 (cit. on p. 60).

[DUr+14] Pierpaolo D’Urso, Livia De Giovanni, Elizabeth Ann Maharaj, and Riccardo Massari. “Wavelet-based self-organizing maps for classifying multivariate time series”. In: *Journal of Chemometrics* 28.1 (2014), pp. 28–51 (cit. on p. 57).

[Dar+09] Srivatsava Daruru, Nena M Marin, Matt Walker, and Joydeep Ghosh. “Pervasive parallelism in data mining: dataflow solution to co-clustering large and sparse netflix data”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 1115–1124 (cit. on p. 59).

[Das14] Rajarshi Das. “Collapsed Gibbs sampler for dirichlet process Gaussian mixture models (DPGMM)”. In: *Technical report, Carnegie Mellon University, United States* (2014) (cit. on p. 159).

[DSP05] Tamraparni Dasu, Deborah F Swayne, and David Poole. "Grouping multivariate time series: A case study". In: *Proceedings of the IEEE Workshop on Temporal Data Mining: Algorithms, Theory and Applications, in conjunction with the Conference on Data Mining, Houston*. Citeseer. 2005, pp. 25–32 (cit. on pp. 57, 58).

[Dau+18] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, et al. *The UCR Time Series Classification Archive*. 2018 (cit. on p. 79).

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22 (cit. on p. 41).

[Den+13] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. "A time series forest for classification and feature extraction". In: *Information Sciences* 239 (2013), pp. 142–153 (cit. on p. 32).

[Dhi01] Inderjit S Dhillon. "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*. 2001, pp. 269–274 (cit. on p. 59).

[Di+18] Ruitong Di, Hong Wang, Youli Fang, and Ying Zhou. "Fake comment detection based on time series and density peaks clustering". In: *International conference on algorithms and architectures for parallel processing*. Springer. 2018, pp. 124–130 (cit. on p. 34).

[Din+06] Chris Ding, Tao Li, Wei Peng, and Haesun Park. "Orthogonal nonnegative matrix t-factorizations for clustering". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pp. 126–135 (cit. on p. 60).

[DTS07] Nicolas Dobigeon, Jean-Yves Tourneret, and Jeffrey D Scargle. "Joint segmentation of multivariate astronomical time series: Bayesian sampling with a hierarchical model". In: *IEEE Transactions on Signal Processing* 55.2 (2007), pp. 414–423 (cit. on p. 57).

[Du+19] Mingjing Du, Shifei Ding, Yu Xue, and Zhongzhi Shi. "A novel density peaks clustering with sensitivity of local density and density-adaptive metric". In: *Knowledge and Information Systems* 59.2 (2019), pp. 285–309 (cit. on p. 33).

[Dua+19] Haiyang Duan, Xudong Wang, Yu Bai, Man Yao, and Qingtao Guo. "Integrated approach to density-based spatial clustering of applications with noise and dynamic time warping for breakout prediction in slab continuous casting". In: *Metallurgical and Materials Transactions B* 50.5 (2019), pp. 2343–2353 (cit. on p. 33).

[Esc94] Michael D Escobar. "Estimating normal means with a Dirichlet process prior". In: *Journal of the American Statistical Association* 89.425 (1994), pp. 268–277 (cit. on p. 50).

[Est+96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *kdd*. Vol. 96. 34. 1996, pp. 226–231 (cit. on p. 33).

[FRM94] Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos. “Fast subsequence matching in time-series databases”. In: *ACM Sigmod Record* 23.2 (1994), pp. 419–429 (cit. on p. 30).

[Faw+19] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. “Deep learning for time series classification: a review”. In: *Data mining and knowledge discovery* 33.4 (2019), pp. 917–963 (cit. on p. 57).

[FZZ20] Liang Feng, Qianchuan Zhao, and Cangqi Zhou. “Improving performances of Top-N recommendations with co-clustering method”. In: *Expert Systems with Applications* 143 (2020), p. 113078 (cit. on p. 59).

[For+19] Florent Forest, Mustapha Lebbah, Hanane Azzag, and Jérôme Lacaille. “Deep Embedded SOM: Joint Representation Learning and Self-Organization”. In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*. 2019 (cit. on pp. 33, 35).

[For+00] Florent Forest, Mustapha Lebbah, Hanene Azzag, and Jérôme Lacaille. “Deep embedded SOM: joint representation learning and self-organization”. In: *reconstruction* 500 (2000), p. 500 (cit. on p. 34).

[For+21] Florent Forest, Alex Mourer, Mustapha Lebbah, Hanane Azzag, and Jérôme Lacaille. “An Invariance-guided Stability Criterion for Time Series Clustering Validation”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 9296–9303 (cit. on pp. 44, 107).

[FK08] Sylvia Fröhwirth-Schnatter and Sylvia Kaufmann. “Model-based clustering of multiple time series”. In: *Journal of Business & Economic Statistics* 26.1 (2008), pp. 78–89 (cit. on p. 39).

[Fuc+10] Erich Fuchs, Thiemo Gruber, Jiri Nitschke, and Bernhard Sick. “Online segmentation of time series based on polynomial least-squares approximations”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.12 (2010), pp. 2232–2245 (cit. on p. 69).

[GS99] Scott Gaffney and Padhraic Smyth. “Trajectory clustering with mixtures of regression models”. In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. 1999, pp. 63–72 (cit. on p. 39).

[Gaf04] Scott John Gaffney. *Probabilistic curve-aligned clustering and prediction with regression mixture models*. University of California, Irvine, 2004 (cit. on p. 39).

[GMS18] Giuliano Galimberti, Annamaria Manisi, and Gabriele Soffritti. “Modelling the role of variables in model-based cluster analysis”. In: *Statistics and Computing* 28.1 (2018), pp. 145–169 (cit. on p. 65).

[GS07] Giuliano Galimberti and Gabriele Soffritti. “Model-based methods to identify multiple cluster structures in a data set”. In: *Computational statistics & data analysis* 52.1 (2007), pp. 520–536 (cit. on pp. 65, 83).

[Gel+13] Andrew Gelman, John B Carlin, Hal S Stern, et al. *Bayesian data analysis*. CRC press, 2013 (cit. on p. 128).

[GM05] Thomas George and Srujana Merugu. “A scalable collaborative filtering framework based on co-clustering”. In: *Fifth IEEE International Conference on Data Mining (ICDM’05)*. IEEE. 2005, 4–pp (cit. on p. 59).

[GGM14] Shima Ghassempour, Federico Girosi, and Anthony Maeder. “Clustering multivariate time series using hidden Markov models”. In: *International journal of environmental research and public health* 11.3 (2014), pp. 2741–2763 (cit. on p. 58).

[GV07] Subhashis Ghosal and Aad Van Der Vaart. “Posterior convergence rates of Dirichlet mixtures at smooth densities”. In: *The Annals of Statistics* (2007), pp. 697–723 (cit. on p. 148).

[Gio09] Toni Giorgino. “Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package”. In: *Journal of Statistical Software* 31.7 (2009), pp. 1–24 (cit. on p. 28).

[GOK18] Thomas J Glassen, Timo von Oertzen, and Dmitry A Konovalov. “Finding the mean in a partition distribution”. In: *BMC bioinformatics* 19.1 (2018), pp. 1–10 (cit. on pp. 53, 62, 113, 130).

[Goo65] IJ Good. “Categorization of classification. Mathematics and Computer Science in Biology and Medicine”. In: *Her Majesty’s Stationary O ce, London* (1965) (cit. on p. 59).

[Gov83] G Govaert. “Classification croisée”. In: *These d’etat, Universite Paris* 6 (1983) (cit. on p. 59).

[GN08] Gérard Govaert and Mohamed Nadif. “Block clustering with Bernoulli mixture models: Comparison of different approaches”. In: *Computational Statistics & Data Analysis* 52.6 (2008), pp. 3233–3245 (cit. on pp. 60, 84).

[GN07] Gérard Govaert and Mohamed Nadif. “Clustering of contingency table and mixture model”. In: *European Journal of Operational Research* 183.3 (2007), pp. 1055–1066 (cit. on p. 60).

[GN03] Gérard Govaert and Mohamed Nadif. “Clustering with block mixture models”. In: *Pattern Recognition* 36.2 (2003), pp. 463–473 (cit. on pp. 60, 84).

[GN13] Gérard Govaert and Mohamed Nadif. *Co-clustering: models, algorithms and applications*. John Wiley & Sons, 2013 (cit. on pp. 59, 61, 126).

[Gre90] Peter J Green. “On use of the EM algorithm for penalized likelihood estimation”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 52.3 (1990), pp. 443–452 (cit. on p. 66).

[Gua+10] Yue Guan, Jennifer G Dy, Donglin Niu, and Zoubin Ghahramani. "Variational inference for nonparametric multiple clustering". In: *MultiClust Workshop, KDD-2010*. 2010 (cit. on pp. 67, 125).

[Gup+96] Lalit Gupta, Dennis L Molfese, Ravi Tammana, and Panagiotis G Simos. "Nonlinear alignment and averaging for estimating the evoked potential". In: *IEEE transactions on biomedical engineering* 43.4 (1996), pp. 348–356 (cit. on p. 38).

[HNB19] David Hallac, Peter Nystrup, and Stephen Boyd. "Greedy Gaussian segmentation of multivariate time series". In: *Advances in Data Analysis and Classification* 13.3 (2019), pp. 727–751 (cit. on p. 57).

[Han+02] Daniel Hanisch, Alexander Zien, Ralf Zimmer, and Thomas Lengauer. "Co-clustering of biological networks and gene expression data". In: *Bioinformatics* 18.suppl_1 (2002), S145–S154 (cit. on p. 59).

[Har72] John A Hartigan. "Direct clustering of a data matrix". In: *Journal of the american statistical association* 67.337 (1972), pp. 123–129 (cit. on p. 59).

[HLR15] David I Hastie, Silvia Liverani, and Sylvia Richardson. "Sampling from Dirichlet process mixture models with unknown concentration parameter: mixing issues in large data implementations". In: *Statistics and computing* 25.5 (2015), pp. 1023–1037 (cit. on p. 112).

[Hed+01] Ingrid Hedenfalk, David Duggan, Yidong Chen, et al. "Gene-expression profiles in hereditary breast cancer". In: *New England Journal of Medicine* 344.8 (2001), pp. 539–548 (cit. on p. 59).

[HK21] Jens C Hegg and Brian P Kennedy. "Lets Do the Time Warp Again: Non-linear time series matching as a tool for sequentially structured data in ecology". In: *bioRxiv* (2021) (cit. on p. 58).

[HW21] Matthieu Herrmann and Geoffrey I Webb. "Early abandoning and pruning for elastic distances including dynamic time warping". In: *Data Mining and Knowledge Discovery* (2021), pp. 1–25 (cit. on p. 28).

[HT05] Shoji Hirano and Shusaku Tsumoto. "Empirical comparison of clustering methods for long time-series databases". In: *Active Mining*. Springer, 2005, pp. 268–286 (cit. on p. 36).

[Hot33] Harold Hotelling. "Analysis of a complex of statistical variables into principal components." In: *Journal of educational psychology* 24.6 (1933), p. 417 (cit. on p. 31).

[HP18] Juhua Hu and Jian Pei. "Subspace multi-clustering: a review". In: *Knowledge and information systems* 56.2 (2018), pp. 257–284 (cit. on p. 125).

[HA07] John P Huelsenbeck and Peter Andolfatto. "Inference of population structure under a Dirichlet process model". In: *Genetics* 175.4 (2007), pp. 1787–1802 (cit. on p. 53).

[II20] Dino Ienco and Roberto Interdonato. “Deep multivariate time series embedding clustering via attentive-gated autoencoder”. In: *Advances in Knowledge Discovery and Data Mining* 12084 (2020), p. 318 (cit. on pp. 57, 58).

[IK13] Félix Iglesias and Wolfgang Kastner. “Analysis of similarity measures in times series clustering for the discovery of building energy patterns”. In: *Energies* 6.2 (2013), pp. 579–597 (cit. on pp. 27, 28).

[JB18] Julien Jacques and Christophe Biernacki. “Model-based co-clustering for ordinal data”. In: *Computational Statistics & Data Analysis* 123 (2018), pp. 101–115 (cit. on p. 60).

[JP13] Julien Jacques and Cristian Preda. “Funclust: A curves clustering method using functional random variables density approximation”. In: *Neurocomputing* 112 (2013), pp. 164–171 (cit. on p. 39).

[JP14] Julien Jacques and Cristian Preda. “Functional data clustering: a survey”. In: *Advances in Data Analysis and Classification* 8.3 (2014), pp. 231–255 (cit. on p. 31).

[JLR20] Ali Javed, Byung Suk Lee, and Donna M Rizzo. “A benchmark study on time series clustering”. In: *Machine Learning with Applications* 1 (2020), p. 100001 (cit. on pp. 34, 36).

[Kat16] Rohit J Kate. “Using dynamic time warping distances as features for improved time series classification”. In: *Data Mining and Knowledge Discovery* 30.2 (2016), pp. 283–312 (cit. on pp. 33, 147).

[KR09] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Vol. 344. John Wiley & Sons, 2009 (cit. on p. 38).

[Keh04] Ath Kehagias. “A hidden Markov model segmentation procedure for hydrological and environmental time series”. In: *Stochastic Environmental Research and Risk Assessment* 18.2 (2004), pp. 117–130 (cit. on p. 69).

[Keo+01] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. “Dimensionality reduction for fast similarity search in large time series databases”. In: *Knowledge and information Systems* 3.3 (2001), pp. 263–286 (cit. on p. 29).

[Keo+04] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. “Segmenting time series: A survey and novel approach”. In: *Data mining in time series databases*. World Scientific, 2004, pp. 1–21 (cit. on p. 69).

[Keo+07] Eamonn Keogh, Stefano Lonardi, Chotirat Ann Ratanamahatana, et al. “Compression-based data mining of sequential data”. In: *Data Mining and Knowledge Discovery* 14.1 (2007), pp. 99–129 (cit. on p. 28).

[KR05] Eamonn Keogh and Chotirat Ann Ratanamahatana. “Exact indexing of dynamic time warping”. In: *Knowledge and information systems* 7.3 (2005), pp. 358–386 (cit. on p. 28).

[Ker+15] Christine Keribin, Vincent Brault, Gilles Celeux, and Gérard Govaert. “Estimation and selection for the latent block model on categorical data”. In: *Statistics and Computing* 25.6 (2015), pp. 1201–1216 (cit. on pp. 63, 116).

[Kip+18] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. “Neural relational inference for interacting systems”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2688–2697 (cit. on p. 57).

[Kir05] Sergey Kirshner. *Modeling of multivariate time series using hidden Markov models*. University of California, Irvine, 2005 (cit. on p. 58).

[Klu+03] Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. “Spectral biclustering of microarray data: coclustering genes and conditions”. In: *Genome research* 13.4 (2003), pp. 703–716 (cit. on p. 59).

[KLB05] Dmitry A Konovalov, Bruce Litow, and Nigel Bajema. “Partition-distance via the assignment problem”. In: *Bioinformatics* 21.10 (2005), pp. 2463–2468 (cit. on p. 53).

[KM86] Mirko Křivánek and Jaroslav Morávek. “NP-hard problems in hierarchical-tree clustering”. In: *Acta informatica* 23.3 (1986), pp. 311–323 (cit. on p. 53).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105 (cit. on p. 34).

[LN11] Lazhar Labiod and Mohamed Nadif. “Co-clustering for binary and categorical data with maximum modularity”. In: *2011 IEEE 11th international conference on data mining*. IEEE. 2011, pp. 1140–1145 (cit. on p. 60).

[LN17] Charlotte Laclau and Mohamed Nadif. “Diagonal latent block model for binary data”. In: *Statistics and Computing* 27.5 (2017), pp. 1145–1163 (cit. on p. 60).

[Lac+17] Charlotte Laclau, Ievgen Redko, Basarab Matei, Younes Bennani, and Vincent Brault. “Co-clustering through optimal transport”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1955–1964 (cit. on p. 60).

[LM00] Marc Lavielle and Eric Moulines. “Least-squares estimation of an unknown number of shifts in a time series”. In: *Journal of time series analysis* 21.1 (2000), pp. 33–59 (cit. on p. 69).

[LS01] DD Lee and HS Seung. *Algorithms for non-negative matrix factorization Advances in Neural Information Processing 13 (Proc. NIPS 2000)*. 2001 (cit. on p. 60).

[Lee+20] Seulbi Lee, Jaehoon Kim, Jongyeon Hwang, et al. “Clustering of Time Series Water Quality Data Using Dynamic Time Warping: A Case Study from the Bukhan River Water Quality Monitoring Network”. In: *Water* 12.9 (2020), p. 2411 (cit. on p. 58).

[Lev66] Vladimir I Levenshtein. “Binary codes capable of correcting deletions, insertions, and reversals”. In: *Soviet physics doklady*. Vol. 10. 1966, pp. 707–710 (cit. on p. 76).

[Li19] Hailin Li. “Multivariate time series clustering based on common principal component analysis”. In: *Neurocomputing* 349 (2019), pp. 239–247 (cit. on p. 57).

[Li05] Tao Li. “A general model for clustering binary data”. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005, pp. 188–197 (cit. on p. 60).

[Li+13] Zheng-Xin Li, Jian-Sheng Guo, Xiao-Bin Hui, and Fei-Fei Song. “Dimension reduction method for multivariate time series based on common principal component”. In: *Control and Decision* 28.4 (2013), pp. 531–536 (cit. on p. 57).

[Lin+03] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. “A symbolic representation of time series, with implications for streaming algorithms”. In: *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. 2003, pp. 2–11 (cit. on p. 78).

[Lin+07] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. “Experiencing SAX: a novel symbolic representation of time series”. In: *Data Mining and knowledge discovery* 15.2 (2007), pp. 107–144 (cit. on p. 29).

[LL09] Jessica Lin and Yuan Li. “Finding structural similarity in time series data using bag-of-patterns representation”. In: *International conference on scientific and statistical database management*. Springer. 2009, pp. 461–477 (cit. on p. 77).

[Liu+18] Yongli Liu, Jingli Chen, Shuai Wu, Zhizhong Liu, and Hao Chao. “Incremental fuzzy C medoids clustering of time series data using dynamic time warping distance”. In: *Plos one* 13.5 (2018), e0197499 (cit. on p. 38).

[LP14] Woong-Kee Loh and Young-Ho Park. “A survey on density-based clustering algorithms”. In: *Ubiquitous information technologies and applications*. Springer, 2014, pp. 775–780 (cit. on p. 33).

[Lom12] Aurore Lomet. “Sélection de modèle pour la classification croisée de données continues”. PhD thesis. Compiègne, 2012 (cit. on pp. 63, 89, 95).

[LM14] António M Lopes and JA Tenreiro Machado. “Analysis of temperature time-series: Embedding dynamics into the MDS method”. In: *Communications in Nonlinear Science and Numerical Simulation* 19.4 (2014), pp. 851–871 (cit. on p. 31).

[Łuc16] Maciej Łuczak. “Hierarchical clustering of time series data with parametric derivative dynamic time warping”. In: *Expert Systems with Applications* 62 (2016), pp. 116–130 (cit. on p. 36).

[Mac+67] James MacQueen et al. “Some methods for classification and analysis of multivariate observations”. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297 (cit. on p. 37).

[Mad18] Naveen Sai Madiraju. “Deep temporal clustering: Fully unsupervised learning of time-domain features”. PhD thesis. Arizona State University, 2018 (cit. on p. 34).

[Man+09] Vikash K Mansinghka, Eric Jonas, Cap Petschulat, et al. “Cross-categorization: A method for discovering multiple overlapping clusterings”. In: *Nonparametric Bayes Workshop at NIPS*. 2009 (cit. on p. 67).

[MV19] Matthieu Marbac and Vincent Vandewalle. “A tractable multi-partitions clustering”. In: *Computational Statistics & Data Analysis* 132 (2019), pp. 167–179 (cit. on pp. 66, 83, 125).

[MHM18] Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018) (cit. on p. 32).

[MLR19] Geoffrey J McLachlan, Sharon X Lee, and Suren I Rathnayake. “Finite mixture models”. In: *Annual review of statistics and its application* 6 (2019), pp. 355–378 (cit. on p. 40).

[MR07] Edward Meeds and Sam Roweis. *Nonparametric bayesian biclustering*. Tech. rep. Citeseer, 2007 (cit. on pp. 64, 108, 109, 128, 134).

[Meg+19] Khadidja Meguelati, Bénédicte Fontez, Nadine Hilgert, and Florent Masseglia. “Dirichlet process mixture models made scalable and effective by means of massive distribution”. In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 2019, pp. 502–509 (cit. on pp. 50, 51, 145).

[Mil19] Jeffrey W Miller. “An elementary derivation of the Chinese restaurant process from Sethuraman’s stick-breaking process”. In: *Statistics & Probability Letters* 146 (2019), pp. 112–117 (cit. on pp. 46, 148).

[MH13] Jeffrey W Miller and Matthew T Harrison. “A simple example of Dirichlet process mixture inconsistency for the number of components”. In: *arXiv preprint arXiv:1301.2708* (2013) (cit. on p. 148).

[MH14] Jeffrey W Miller and Matthew T Harrison. “Inconsistency of Pitman-Yor process mixtures for the number of components”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3333–3370 (cit. on p. 148).

[Min+06] David Minnen, Thad Starner, Irfan Essa, and Charles Isbell. “Discovering characteristic actions from on-body sensor data”. In: *2006 10th IEEE international symposium on wearable computers*. IEEE. 2006, pp. 11–18 (cit. on p. 31).

[Mur07] Kevin P Murphy. “Conjugate Bayesian analysis of the Gaussian distribution”. In: *def* $1.2\sigma 2$ (2007), p. 16 (cit. on pp. 51, 159, 162).

[NG10] Mohamed Nadif and Gérard Govaert. “Model-based co-clustering for continuous data”. In: *2010 Ninth international conference on machine learning and applications*. IEEE. 2010, pp. 175–180 (cit. on p. 60).

[Nea00] Radford M Neal. “Markov chain sampling methods for Dirichlet process mixture models”. In: *Journal of computational and graphical statistics* 9.2 (2000), pp. 249–265 (cit. on pp. 50, 52).

[Nel12] Joshua David Nelson. “On K-Means clustering using Mahalanobis distance”. PhD thesis. North Dakota State University, 2012 (cit. on p. 37).

[NH02] Raymond T. Ng and Jiawei Han. “CLARANS: A method for clustering objects for spatial data mining”. In: *IEEE transactions on knowledge and data engineering* 14.5 (2002), pp. 1003–1016 (cit. on p. 38).

[Ngu+17] Minh Nguyen, Sanjay Purushotham, Hien To, and Cyrus Shahabi. “m-tsne: A framework for visualizing high-dimensional multivariate time series”. In: *arXiv preprint arXiv:1708.07942* (2017) (cit. on p. 32).

[NBA13] Viet-An Nguyen, Jordan Boyd-Graber, and Stephen F Altschul. “Dirichlet mixtures, the Dirichlet process, and the structure of protein space”. In: *Journal of Computational Biology* 20.1 (2013), pp. 1–18 (cit. on p. 112).

[Ngu13] XuanLong Nguyen. “Convergence of latent mixing measures in finite and infinite mixture models”. In: *The Annals of Statistics* 41.1 (2013), pp. 370–400 (cit. on p. 148).

[NR07] Vit Niennattrakul and Chotirat Ann Ratanamahatana. “Inaccuracies of shape averaging method using dynamic time warping for time series data”. In: *International conference on computational science*. Springer. 2007, pp. 513–520 (cit. on p. 38).

[NR09] Vit Niennattrakul and Chotirat Ann Ratanamahatana. “Shape averaging under time warping”. In: *2009 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*. Vol. 2. IEEE. 2009, pp. 626–629 (cit. on p. 38).

[NS01] Krzysztof Nowicki and Tom A B Snijders. “Estimation and prediction for stochastic blockstructures”. In: *Journal of the American statistical association* 96.455 (2001), pp. 1077–1087 (cit. on p. 60).

[OW20] Oscar Olesen and Patrick Wadström. “Fault classification using shapelets and deep machine learning”. In: (2020) (cit. on p. 57).

[PG17] John Paparrizos and Luis Gravano. “Fast and accurate time-series clustering”. In: *ACM Transactions on Database Systems (TODS)* 42.2 (2017), pp. 1–49 (cit. on p. 36).

[PG15] John Paparrizos and Luis Gravano. “k-shape: Efficient and accurate clustering of time series”. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 2015, pp. 1855–1870 (cit. on pp. 37, 78).

[PBC21] Clément Pealat, Guillaume Bouleux, and Vincent Cheutet. “Improved Time-Series Clustering with UMAP dimension reduction method”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 5658–5665 (cit. on pp. 32, 33).

[PKG11] François Petitjean, Alain Ketterlin, and Pierre Gançarski. “A global averaging method for dynamic time warping, with applications to clustering”. In: *Pattern recognition* 44.3 (2011), pp. 678–693 (cit. on pp. 38, 78).

[PRB19] Edouard Pineau, Sebastien Razakarivony, and Thomas Bonald. “Seq2var: multivariate time series representation with relational neural networks and linear autoregressive model”. In: *International Workshop on Advanced Analysis and Learning on Temporal Data*. Springer. 2019, pp. 126–140 (cit. on p. 57).

[PGA15] Beatriz Pontes, Raúl Giráldez, and Jesús S Aguilar-Ruiz. “Biclustering on expression data: A review”. In: *Journal of biomedical informatics* 57 (2015), pp. 163–180 (cit. on p. 59).

[PL12] Zoltán Prekopcsák and Daniel Lemire. “Time series classification by class-specific Mahalanobis distance measures”. In: *Advances in Data Analysis and Classification* 6.3 (2012), pp. 185–200 (cit. on p. 28).

[Put+19] Givanna H Putri, Mark N Read, Irena Koprinska, et al. “ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data”. In: *Knowledge-Based Systems* 174 (2019), pp. 9–26 (cit. on p. 34).

[RS02] James O Ramsay and Bernard W Silverman. *Applied functional data analysis: methods and case studies*. Vol. 77. Springer, 2002 (cit. on p. 31).

[RVN21] Nathanaël Randriamihamison, Nathalie Vialaneix, and Pierre Neuvial. “Applicability and interpretability of Ward’s hierarchical agglomerative clustering with or without contiguity constraints”. In: *Journal of Classification* 38.2 (2021), pp. 363–389 (cit. on p. 36).

[RK09] Teemu Räsänen and Mikko Kolehmainen. “Feature-based clustering for electricity use time series data”. In: *International conference on adaptive and natural computing algorithms*. Springer. 2009, pp. 401–412 (cit. on p. 29).

[RK05] Chotirat Ann Ratanamahatana and Eamonn Keogh. “Three myths about dynamic time warping data mining”. In: *Proceedings of the 2005 SIAM international conference on data mining*. SIAM. 2005, pp. 506–510 (cit. on p. 28).

[Res21] Precedence Research. *Advanced Driver Assistance System (ADAS) Market Size, Share, Growth, Trends, Regional Outlook and Forecasts 2021 - 2030*. 2021 (cit. on pp. 2, 12).

[Ric+20] Guillaume Richard, Benoit Grossin, Guillaume Germaine, Georges Hébrail, and Anne de Moliner. “Autoencoder-based time series clustering with energy applications”. In: *arXiv preprint arXiv:2002.03624* (2020) (cit. on p. 32).

[RL14] Alex Rodriguez and Alessandro Laio. “Clustering by fast search and find of density peaks”. In: *science* 344.6191 (2014), pp. 1492–1496 (cit. on p. 33).

[RW13] Jo Røislien and Brita Winje. “Feature extraction across individual time series observations with spikes using wavelet principal component analysis”. In: *Statistics in medicine* 32.21 (2013), pp. 3660–3669 (cit. on p. 147).

[Roo95] Mats Rooth. “Two-dimensional clusters in grammatical relations”. In: *AAAI Symposium on representation and acquisition of lexical knowledge*. 1995 (cit. on p. 60).

[RM18] Gordon J Ross and Dean Markwick. *dirichletprocess: An R Package for Fitting Complex Bayesian Nonparametric Models*. 2018 (cit. on p. 112).

[Sak71] Hiroaki Sakoe. “Dynamic-programming approach to continuous speech recognition”. In: *1971 Proc. the International Congress of Acoustics, Budapest*. 1971 (cit. on p. 28).

[SC78] Hiroaki Sakoe and Seibi Chiba. “Dynamic programming algorithm optimization for spoken word recognition”. In: *IEEE transactions on acoustics, speech, and signal processing* 26.1 (1978), pp. 43–49 (cit. on p. 34).

[SC07] Stan Salvador and Philip Chan. “Toward accurate dynamic time warping in linear time and space”. In: *Intelligent Data Analysis* 11.5 (2007), pp. 561–580 (cit. on p. 28).

[Sam+11] A. Samé, F. Chamroukhi, Gérard Govaert, and P. Aknin. “Model-based clustering and segmentation of time series with changes in regime”. In: *Advances in Data Analysis and Classification* 5 (4 2011), pp. 301–321 (cit. on pp. 32, 39, 70, 73).

[SRV21] Luis Sanhudo, Joao Rodrigues, and Enio Vasconcelos Filho. “Multivariate time series clustering and forecasting for building energy analysis: Application to weather data quality control”. In: *Journal of Building Engineering* 35 (2021), p. 101996 (cit. on p. 58).

[Sch15] Patrick Schäfer. “The BOSS is concerned with time series classification in the presence of noise”. In: *Data Mining and Knowledge Discovery* 29.6 (2015), pp. 1505–1530 (cit. on pp. 29, 77).

[SL17] Patrick Schäfer and Ulf Leser. “Fast and accurate time series classification with weasel”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 637–646 (cit. on p. 29).

[Sch+19] Amandine Schmutz, Julien Jacques, Charles Bouveyron, Laurence Chèze, and Pauline Martin. “Co-clustering de courbes fonctionnelles multivariées”. In: *Journées des Statistiques*. 2019 (cit. on pp. 63, 107).

[SS15] Miranda A Schreurs and Sibyl D Steuwer. “Autonomous driving-political, legal, social, and sustainability dimensions”. In: *Autonomes Fahren*. Springer, 2015, pp. 151–173 (cit. on pp. 2, 12).

[SGH16] NK Schuurman, RPPP Grasman, and EL Hamaker. “A comparison of inverse-wishart prior specifications for covariance matrices in multilevel autoregressive models”. In: *Multivariate Behavioral Research* 51.2-3 (2016), pp. 185–206 (cit. on p. 112).

[Sch78] Gideon Schwarz. “Estimating the dimension of a model”. In: *The annals of statistics* (1978), pp. 461–464 (cit. on p. 44).

[SM13] Pavel Senin and Sergey Malinchik. “Sax-vsm: Interpretable time series classification using sax and vector space model”. In: *2013 IEEE 13th international conference on data mining*. IEEE. 2013, pp. 1175–1180 (cit. on p. 29).

[Set94] Jayaram Sethuraman. “A constructive definition of Dirichlet priors”. In: *Statistica sinica* (1994), pp. 639–650 (cit. on pp. 48, 127).

[SY03] Cyrus Shahabi and Donghui Yan. “Real-time Pattern Isolation and Recognition Over Immersive Sensor Data Streams.” In: *MMM*. Citeseer. 2003, pp. 93–113 (cit. on p. 56).

[SB08] Hanhuai Shan and Arindam Banerjee. “Bayesian co-clustering”. In: *2008 Eighth IEEE International Conference on Data Mining*. IEEE. 2008, pp. 530–539 (cit. on p. 64).

[Shi+21] Ahmed Shifaz, Charlotte Pelletier, Francois Petitjean, and Geoffrey I Webb. “Elastic Similarity Measures for Multivariate Time Series Classification”. In: *arXiv preprint arXiv:2102.10231* (2021) (cit. on p. 56).

[SSB17] Emilie Shireman, Douglas Steinley, and Michael J Brusco. “Examining the effect of initialization strategies on the performance of Gaussian mixture modeling”. In: *Behavior research methods* 49.1 (2017), pp. 282–293 (cit. on p. 42).

[Sho+17] Mohammad Shokoohi-Yekta, Bing Hu, Hongxia Jin, Jun Wang, and Eamonn Keogh. “Generalizing DTW to the multi-dimensional case requires an adaptive approach”. In: *Data mining and knowledge discovery* 31.1 (2017), pp. 1–31 (cit. on p. 56).

[SWK15] Mohammad Shokoohi-Yekta, Jun Wang, and Eamonn Keogh. “On the non-trivial generalization of dynamic time warping to the multi-dimensional case”. In: *Proceedings of the 2015 SIAM international conference on data mining*. SIAM. 2015, pp. 289–297 (cit. on p. 56).

[SS05] Ashish Singhal and Dale E Seborg. “Clustering multivariate time-series data”. In: *Journal of Chemometrics: A Journal of the Chemometrics Society* 19.8 (2005), pp. 427–438 (cit. on p. 56).

[SAJ18] Yosra Ben Slimen, Sylvain Allio, and Julien Jacques. “Model-based co-clustering for functional data”. In: *Neurocomputing* 291 (2018), pp. 97–108 (cit. on pp. 6, 17, 63, 84, 86, 107).

[Smy97] Padhraic Smyth. “Clustering sequences with hidden Markov models”. In: *Advances in neural information processing systems*. 1997, pp. 648–654 (cit. on p. 39).

[SDG16] Saeid Soheily-Khah, Ahlame Douzal-Chouakria, and Eric Gaussier. “Generalized k-means-based clustering for temporal data under weighted and kernel time warp”. In: *Pattern Recognition Letters* 75 (2016), pp. 63–69 (cit. on p. 38).

[Spe+18] Ricardo Carlini Sperandio, Simon Malinowski, Laurent Amsaleg, and Romain Tavenard. “Time series retrieval using dtw-preserving shapelets”. In: *International Conference on Similarity Search and Applications*. Springer. 2018, pp. 257–270 (cit. on p. 57).

[SKK00] Michael Steinbach, George Karypis, and Vipin Kumar. “A comparison of document clustering techniques”. In: (2000) (cit. on p. 37).

[TY06] Jun-ichi Takeuchi and Kenji Yamanishi. “A unifying framework for detecting outliers and change points from time series”. In: *IEEE transactions on Knowledge and Data Engineering* 18.4 (2006), pp. 482–492 (cit. on p. 32).

[Tav+20] Neda Tavakoli, Sima Siami-Namini, Mahdi Adl Khanghah, Fahimeh Mirza Soltani, and Akbar Siami Namin. “An autoencoder-based deep learning approach for clustering time series data”. In: *SN Applied Sciences* 2.5 (2020), pp. 1–25 (cit. on p. 32).

[Teh+06] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. “Hierarchical dirichlet processes”. In: *Journal of the american statistical association* 101.476 (2006), pp. 1566–1581 (cit. on p. 48).

[TP94] Hiro Y Toda and Peter CB Phillips. “Vector autoregression and causality: a theoretical overview and simulation study”. In: *Econometric reviews* 13.2 (1994), pp. 259–285 (cit. on p. 57).

[Tok+17] Tomoki Tokuda, Junichiro Yoshimoto, Yu Shimizu, et al. “Multiple co-clustering based on nonparametric mixture models with heterogeneous marginal distributions”. In: *PloS one* 12.10 (2017), e0186566 (cit. on p. 127).

[TW02] Dat Tran and Michael Wagner. “Fuzzy c-means clustering-based speaker verification”. In: *AFSS International Conference on Fuzzy Systems*. Springer. 2002, pp. 318–324 (cit. on p. 38).

[VH08] Laurens Van der Maaten and Geoffrey Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008) (cit. on p. 32).

[Van20] Vincent Vandewalle. “Multi-Partitions Subspace Clustering”. In: *Mathematics* 8.4 (2020), p. 597 (cit. on p. 125).

[VDR14] William R Vega-Brown, Marek Doniec, and Nicholas G Roy. “Nonparametric Bayesian inference on multivariate exponential families”. In: *Advances in Neural Information Processing Systems* 27 (2014), pp. 2546–2554 (cit. on p. 159).

[VF05] Michel Verleysen and Damien François. “The curse of dimensionality in data mining and time series prediction”. In: *International work-conference on artificial neural networks*. Springer. 2005, pp. 758–770 (cit. on p. 28).

[Wan14] Lin Wang. “Multi-band multi-centroid clustering based permutation alignment for frequency-domain blind speech separation”. In: *Digital Signal Processing* 31 (2014), pp. 79–92 (cit. on p. 31).

[WWW07] Xiaozhe Wang, Anthony Wirth, and Liang Wang. “Structure-based statistical features and multivariate time series clustering”. In: *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE. 2007, pp. 351–360 (cit. on p. 57).

[War63] Joe H Ward Jr. “Hierarchical grouping to optimize an objective function”. In: *Journal of the American statistical association* 58.301 (1963), pp. 236–244 (cit. on p. 36).

[Wau18] Didier Wautier. *Autonomous Vehicle’s Digital Simulation Challenges*. 2018 (cit. on pp. 4, 14).

[Wes92] Mike West. *Hyperparameter estimation in Dirichlet process mixture models*. Duke University ISDS Discussion Paper# 92-A03, 1992 (cit. on pp. 118, 129).

[WDX13] Sinead Williamson, Avinava Dubey, and Eric Xing. “Parallel Markov chain Monte Carlo for nonparametric mixture models”. In: *International Conference on Machine Learning*. 2013, pp. 98–106 (cit. on pp. 50, 51).

[WF12] Jason Wyse and Nial Friel. “Block clustering with collapsed latent block models”. In: *Statistics and Computing* 22.2 (2012), pp. 415–428 (cit. on pp. 64, 108).

[Xan14] Petros Xanthopoulos. “A review on consensus clustering methods”. In: *Optimization in science and engineering*. Springer, 2014, pp. 553–566 (cit. on p. 53).

[Xie+19] Juan Xie, Anjun Ma, Anne Fennell, Qin Ma, and Jing Zhao. “It is time to apply biclustering: a comprehensive review of biclustering applications in biological and biomedical data”. In: *Briefings in bioinformatics* 20.4 (2019), pp. 1450–1465 (cit. on p. 59).

[XGF16] Junyuan Xie, Ross Girshick, and Ali Farhadi. “Unsupervised deep embedding for clustering analysis”. In: *International conference on machine learning*. PMLR. 2016, pp. 478–487 (cit. on p. 34).

[XY02] Yimin Xiong and Dit-Yan Yeung. “Mixtures of ARMA models for model-based time series clustering”. In: *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* IEEE. 2002, pp. 717–720 (cit. on p. 39).

[Xu+19] Dongkuan Xu, Wei Cheng, Bo Zong, et al. “Deep co-clustering”. In: *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM. 2019, pp. 414–422 (cit. on p. 60).

[YS04] Kiyoung Yang and Cyrus Shahabi. “A PCA-based similarity measure for multivariate time series”. In: *Proceedings of the 2nd ACM international workshop on Multimedia databases*. 2004, pp. 65–74 (cit. on p. 56).

[ZMK12] Jesin Zakaria, Abdullah Mueen, and Eamonn Keogh. “Clustering time series using unsupervised-shapelets”. In: *2012 IEEE 12th International Conference on Data Mining*. IEEE. 2012, pp. 785–794 (cit. on pp. 27, 31, 38, 57).

[ZA18] Beibei Zhang and Baiguo An. “Clustering time series based on dependence structure”. In: *PloS one* 13.11 (2018), e0206753 (cit. on p. 36).

[ZHT06] Zhang Zhang, Kaiqi Huang, and Tieniu Tan. “Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes”. In: *18th International Conference on Pattern Recognition (ICPR’06)*. Vol. 3. IEEE. 2006, pp. 1135–1138 (cit. on p. 28).

Webpages

[@Spa] Apache Spark. *Apache Spark Expectation maximization Implementation*. URL: http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm (cit. on p. 42).

Clustering Multi-Blocs et Visualisation Analytique de Données Séquentielles Massives Issues de Simulation du Véhicule Autonome

Le développement de systèmes d'aide à la conduite demeure un défi technique pour les constructeurs automobiles. La validation de ces systèmes nécessite de les éprouver dans un nombre considérable de contextes de conduites. Pour ce faire, le Groupe Renault a recourt à la simulation massive, qui permet de reproduire précisément la complexité des conditions physiques de conduite et produit une grande quantité de séries temporelles multivariées. Le rôle de l'expert métier est alors d'explorer ces données pour déterminer précisément les capacités du système d'aide à la conduite étudié et, si nécessaire, de raffiner sa conception.

Un jeu de données simulées peut contenir jusqu'à plusieurs centaines de milliers de simulations, décrites par plusieurs centaines de variables. Face à de telles dimensions, le travail du développeur requiert une grande expertise, ainsi qu'un travail d'investigation minutieux et chronophage.

L'objectif de cette thèse est de produire des algorithmes d'aide à l'exploration et à l'analyse de ces jeux de données issues de simulation massive. Basées sur des hypothèses de construction argumentées, nous proposons deux approches probabilistes de classification non supervisée de séries temporelles, adaptées à des jeux de données temporelles univariées et multivariées.

La première contribution de cette thèse se base sur une hypothèse de construction par scénario pour réaliser une classification de séries temporelles univariées. Cette méthode réalise une segmentation individuelle de chaque série temporelle, puis partitionne l'ensemble des segments extraits pour construire un dictionnaire de 'prototypes' de régimes de conduite. Ce dictionnaire permet de recoder les séries temporelles en séquences de catégories, qui sont alors partitionnées par une méthode de classification hiérarchique ascendante.

La suite des contributions se concentre sur l'analyse de jeux de données multivariées, grâce à des modélisations multi-blocs qui regroupent les différentes variables en fonction de leur distribution et de leur partition individuelle. Ces modèles Bayésiens Non-Paramétriques intègrent nativement une sélection de modèle qui permet d'adapter automatiquement la dimension du modèle au contenu des jeux de données étudiés.

La pertinence de ces contributions est illustrée par des applications issues de cas d'usage industriels Renault : la validation d'un système de freinage d'urgence, d'un système d'aide au maintien dans la voie, et d'un système d'évitement d'urgence.

Mots-clefs: Classification Non Supervisée, Classification par blocs, Classification Multiple, Séries temporelles, Modèles de mélange, Modèles Bayésiens Non Paramétriques, Validation de Systèmes d'Aide a la Conduite.

Keywords: Clustering, Coclustering, Multi-Clustering, Time Series, Mixture Models, Bayesian Non-Parametric Models, Advanced Driver-Assistance Systems validation.