



Trinity College Dublin

Coláiste na Tríonóide, Baile Átha Cliath

The University of Dublin

School of Computer Science and Statistics

A Clustering Framework for Functional Data: Functional Gaussian Process Mixture Model (FunGP)

Xiantao Zhao

Supervisor: Mimi Zhang

October 15, 2024

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Master of Science in Statistics by Research

Declaration

I hereby declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at <http://www.tcd.ie/calendar>.

I have completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at <http://tcd-ie.libguides.com/plagiarism/ready-steady-write>.

I consent / do not consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

I agree that this thesis will not be publicly available, but will be available to TCD staff and students in the University's open access institutional repository on the Trinity domain only, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Signed: Xiantao Zhao

Date: March 18, 2022

Abstract

Functional data analysis (FDA) is a rapidly evolving field that focuses on the analysis and interpretation of data where each observation is a function, typically represented by curves or surfaces over a continuum such as time or space. this dissertation presents a novel clustering framework, tailored for functional data, named FunGP. The proposed framework is divided into three layers, including smoother, projector, and clustering. Each layer can be used individually or combined into a complete clustering model process. Such a design aims for flexibility and ease of use, allowing methods and parameters to be adapted to user needs.

To verify the effectiveness of our framework, we conduct comprehensive evaluations using simulated and real data sets, demonstrating that FunGP not only surpasses traditional clustering methods in accuracy but also exhibits significant robustness in different scenarios. Our results highlight the potential of FunGP as a preferred solution for functional data clustering, combining accuracy, flexibility, and user-friendliness to address a wide range of analytical needs.

Acknowledgements

This work would not have been possible without the support and encouragement of many individuals, to whom I owe my deepest gratitude.

I extend my heartfelt thanks to my parents, whose unwavering love and support have been the cornerstone of my journey.

I am greatly grateful to my supervisor, Dr. Mimi Zhang, whose mentorship has been instrumental in the fruition of this project. Her expertise, insightful guidance, patience, and constant support have not only shaped this research but have also contributed significantly to my personal and academic growth.

I would also like to extend my sincere appreciation to the faculty, staff, and colleagues at the Trinity College Dublin, School of Computer Science and Statistics.

Special thanks to my friends, their support is of great help to me.

I am particularly grateful to I-Form, the Science Foundation Ireland Research Centre for Advanced Manufacturing, that provided financial support for my academic journey.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Data Declaration	3
1.3	Dissertation Structure	3
2	Background Research	5
2.1	Functional Data Analysis	5
2.1.1	Functional Data Smoothing	6
2.1.2	Functional Principal Component Analysis	7
2.2	Gaussian Process Mixture	8
2.3	Clustering	10
2.3.1	Functional Data Clustering	10
3	Model and Methodology	12
3.1	Initialization	13
3.2	Smoother	14
3.3	Projector	18
3.4	Univariate Gaussian Mixture Model: UniGMM	19
4	Evaluation	23
4.1	Simulation Scenarios	23
4.1.1	Simulation Scenarios Description	25
4.1.2	Simulation Scenarios Evaluation	27
4.1.3	Simulation Scenarios Conclusion	39
4.2	Real Data Scenarios	40
4.2.1	Real Data Description	40
4.2.2	Real Data Evaluation	44
4.2.3	Real Data Conclusion	59
4.3	Conclusion	62

5	Discussion	63
5.1	Research Challenges	63
5.2	Contribution	64
5.3	Future Work	65
6	Conclusion	67
A1	Appendix: FunGP Introduction	72
A1.1	Initialisation	72
A1.2	Smoother	72
A1.3	Projector	73
A1.4	Unigmm	73
A2	Appendix: Sensitivity Analysis	74
A2.1	Experimental Setup	74
A2.2	Results	75
A2.3	Conclusion	78

List of Figures

2.1	Functional Data and Smoothing	7
3.1	Model Structure	13
3.2	Smoother Example	17
4.1	Simulation Scenarios	24
4.2	Noised Simulation Scenarios	24
4.3	Clustering Results for Simulation Scenario 1	30
4.4	Clustering Results for Simulation Scenario 2	31
4.5	Clustering Results for Simulation Scenario 3	33
4.6	Clustering Results for Simulation Scenario 4	34
4.7	Clustering Results for Simulation Scenario 5	36
4.8	Clustering Results for Simulation Scenario 6	37
4.9	Clustering Results for Simulation Scenario 7	38
4.10	Real Data Scenarios	40
4.11	Clustering Results for Dataset Arrowhead	45
4.12	Clustering Results for Dataset Bird Chicken	47
4.13	Clustering Results for Dataset CBF	49
4.14	Clustering Results for Dataset Diatom Size Reduction	50
4.15	Clustering Results for Dataset ECG200	51
4.16	Clustering Results for Dataset Face Four	52
4.17	Clustering Results for Dataset Gun Point	54
4.18	Clustering Results for Dataset Meat	55
4.19	Clustering Results for Dataset Small Kitchen Appliances	56
4.20	Clustering Results for Dataset Symbols	57
4.21	Clustering Results for Dataset Synthetic Control	58
4.22	Clustering Results for Dataset Worms	60
A2.1	AMI Score against Varied Smoother Parameter	77
A2.2	ARI Score against Varied Smoother Parameter	77
A2.3	GCV Score against Varied Smoother Parameter	78

A2.4 GCV Score against AMI and ARI Scores	78
---	----

List of Tables

4.1	Simulation Scenario Description	27
4.2	Simulation Scenario ARI Scores	28
4.3	Simulation Scenario AMI Scores	28
4.4	Noised Simulation Scenario ARI Scores	29
4.5	Noised Simulation Scenario AMI Scores	29
4.6	Real Data Description	44
4.7	AMI and ARI Scores for Dataset Arrowhead	46
4.8	AMI and ARI Scores for Dataset Bird Chicken	47
4.9	AMI and ARI Scores for Dataset CBF	48
4.10	AMI and ARI Scores for Dataset Diatom Size Reduction	49
4.11	AMI and ARI Scores for Dataset ECG200	50
4.12	AMI and ARI Scores for Dataset Face Four	52
4.13	AMI and ARI Scores for Dataset Gun Point	53
4.14	AMI and ARI Scores for Dataset Meat	54
4.15	AMI and ARI Scores for Dataset Small Kitchen Appliances	55
4.16	AMI and ARI Scores for Dataset Symbols	56
4.17	AMI and ARI Scores for Dataset Synthetic Control	58
4.18	AMI and ARI Scores for Dataset Worms	59
4.19	ARI Scores for Real Data	60
4.20	AMI Scores for Real Data	61
A2.1	AMI Score for Different n_{basis} and Period	75
A2.2	ARI Score for Different n_{basis} and Period	75
A2.3	GCV Score for Different n_{basis} and Period	76

Notation

$\mathcal{T}$	function domain
X	random dunction
Ω	sample space, the set of possible outcomes
$\mathcal{F}$	event space, the set of events
$\Pr$	probability function
$\{x_i(t_i) : i = 1, \dots, n\}$	n sample function
$x = X(\cdot, \omega) \in \mathcal{H}(\mathcal{T}, \mathbb{R})$	the value of the X at the outcome $\omega \in \Omega$
$x(t)$	a random realization of the real-valued random variable $X(t, \cdot)$
$y_i(t)$	the observation of x_i at any point $t \in \mathcal{T}$
$\hat{y}_i(t)$	smoothed estimate of $y_i(t)$
$\{y_i(t_i) : i = 1, \dots, n\}$	raw data
ε_i	noise process
K	number of clusters
Z	hidden cluster label
π_k	mixing proportion $\Pr(Z = k) = \pi_k, \pi_k > 0$ and $\sum_{k=1}^K \pi_k = 1$
μ_k	mean function
Σ_k	covariance function
$GP(\mu_k, \Sigma_k)$	Gaussian Process (GP) with a mean function μ_k and a covariance function Σ_k
λ_{k1}	eigen-values
$\{b_{kv}\}_{v \in \mathbb{N}}$	orthonormal eigenfunctions
a_{ikv}	principle score associated with the eigenfunction b_{kv}
$\{\beta_v\}_{v \in \mathbb{N}}$	orthonormal basis
α_{iv}	projection coefficient onto the basis function β_v
M	membership matrix
B	binary membership indicator matrix
w	weight
A	affinity matrix
$z_i \in \{1, \dots, K\}$	hidden cluster label

1 Introduction

Functional data analysis (FDA) is a powerful statistical method for working with data represented as functions or curves rather than traditional numerical or categorical values. Essentially, it extends classical multivariate analysis techniques to environments where data points are viewed as curves or functions on a continuum such as time, space, or any other ordered domain. Because of its ability to capture and analyze the complex patterns and structures inherent in functional data, FDA has gained great interest and significance in a variety of fields including, but not limited to, biology, finance, engineering, and neuroscience. By focusing on the characteristics inherent in the functional form of data, FDA enables researchers and practitioners to discover meaningful insights, model relationships, make predictions, and make informed decisions, thus providing a versatile and indispensable toolkit for addressing real-world challenges in data analysis.

To address the challenge of clustering functional data, this project introduces a new novel functional data clustering framework termed the Functional Gaussian Process Mixture Model (FunGP). The proposed framework has the potential to capture the inherent variability of functional data and provide more accurate and reliable clustering results. This project aims to demonstrate the effectiveness of the proposed approach through various simulations and analyses, with the goal of contributing to the field of FDA by providing a new and effective approach for clustering functional data.

In our exploration of the FunGP, we introduce a novel three-layer structure designed to enhance the accuracy and efficiency of data clustering. The first layer, termed the smoother, employs a variety of smoothing methods to preprocess the data, allowing for flexibility in addressing the initial noise and irregularities present in raw datasets. This preparatory step is crucial for refining the data and making it more amenable to accurate clustering. Proceeding to the second layer, the projector transforms the smoothed data into a structured format suitable for cluster analysis. By leveraging different projection techniques, this layer effectively maps the preprocessed data into a space where the intrinsic patterns and relationships are more pronounced and readily discernible. The culmination of this process occurs in the third layer, where the data, now optimally prepared and structured, is subjected to a clustering method rooted in the Gaussian Mixture Model (GMM). This final

stage harnesses the power of GMM to discern and delineate the clusters within the data, yielding the ultimate clustering outcomes. Through this three-tiered approach, the FunGP model aims to provide a robust and versatile framework for clustering, capable of accommodating a wide array of data characteristics and clustering requirements.

The utilization of different smoothers enables us to adaptively adjust the level of smoothness applied to the functional data, allowing for the accommodation of varying degrees of complexity and noise within the datasets. Moreover, by employing a range of projectors, we offer users the ability to transform the functional data into different representation spaces, thereby facilitating the exploration of alternative data structures and feature spaces for clustering purposes.

By offering a diverse set of smoothers and projectors within our framework, we aim to enhance the versatility and applicability of functional data clustering across a wide range of domains and applications. Whether dealing with highly structured or noisy functional data, our framework provides users with the tools to effectively capture and analyze the underlying patterns and relationships, ultimately leading to more robust and insightful clustering results.

Through empirical evaluations and real-world case studies, we demonstrate the efficacy and flexibility of our framework in accommodating diverse data characteristics and addressing the unique challenges associated with clustering functional data. By empowering users with the ability to choose from a variety of smoothers and projectors, our framework represents a significant advancement in the field of functional data analysis, offering a customizable and adaptable solution for clustering complex functional datasets.

1.1 Motivation

The motivation behind our work is driven by the need to address the inherent complexity and diversity of functional data encountered across various domains, including biomedical research and financial analysis. Traditional clustering methods often falter in effectively capturing the intricate patterns and structures within functional datasets, characterized by curves or functions evolving over continuous domains such as time or space.

Functional data presents unique challenges, including high dimensionality, irregular sampling, and inherent variability, which conventional clustering approaches—typically predicated on Euclidean data structures—may inadequately capture. These traditional methods struggle to encapsulate the underlying relationships and dynamics within functional datasets, highlighting the urgency for innovative methodologies that can embrace the richness and complexity of functional data, offering interpretable and actionable insights [1–3].

Our motivation is fueled by the aspiration to equip researchers and practitioners with a

versatile and adaptable framework for clustering functional data. By providing a wide selection of smoothers and projectors, our aim is to enable users to customize the clustering process to meet the specific characteristics and needs of their datasets. This approach not only broadens the applicability of clustering techniques to an extensive range of functional data but also deepens the understanding of the underlying data structures and patterns.

Additionally, our framework is developed with a focus on usability. Recognizing the significance of accessibility, we strive to democratize the application of advanced clustering techniques by ensuring our framework is user-friendly and intuitive. Our goal is to empower individuals at different levels of expertise, including those without professional backgrounds, to leverage functional data analysis in their specific contexts.

Comprehensive documentation and user guides accompany our framework, offering detailed instructions and explanations to navigate the clustering process. We provide clear descriptions of the choices available for smoothers and projectors, complemented by practical examples and use cases, to enhance understanding and facilitate adoption.

1.2 Data Declaration

This study evaluates the efficacy of the Functional Gaussian Process Mixture Model (FunGP) for data clustering using multiple real-world datasets from diverse fields. These datasets were chosen to represent a diverse set of challenges typically encountered in functional data analysis, including but not limited to high dimensionality, intricate correlation structures within and across functional observations, and varying levels of noise.

Simulated datasets were generated to assess the model's performance under controlled conditions. These datasets allowed us to systematically vary the number of functions, their complexity, and the noise levels to evaluate the robustness and scalability of FunGP.

All the datasets are publicly available from online sources. In Chapter 4, we will discuss the datasets that are central to our examination of the FunGP for data clustering. These datasets have been carefully selected to challenge and validate the efficacy of the FunGP model across various scenarios, ensuring a thorough analysis of its performance and adaptability.

1.3 Dissertation Structure

This dissertation meticulously examines the concept of FunGP, organizing the investigation into six distinct chapters, each serving a specific purpose within the broader scope of the research. Chapter 1, Introduction, sets the scene by presenting the research topic,

delineating the study's primary objectives, and underscoring its importance to the wider academic and practical field. Chapter 2, Background Research, offers an exhaustive review of the existing literature, establishing a scholarly context for the study. Chapter 3, Model and Methodology, introduces the proposed model, detailing its theoretical underpinnings and operational mechanisms. Chapter 4, Evaluation, rigorously tests the model using both simulated and real-world data to assess its validity and applicability. Chapter 5, Discussion, delves into the implications of the findings, addressing potential concerns and outlining future prospects. Finally, Chapter 6, Conclusion, concisely summarizes the entire research effort, reflecting on its contributions.

2 Background Research

This chapter presents an extensive literature review on the topics of functional data clustering and Gaussian processes, as well as the most advanced methodologies currently available in these domains. The objective is to develop a thorough comprehension of the existing framework, encompassing both the theoretical foundations and the practical implementations, alongside the constraints inherent to current methods. By analyzing these key elements, we strive to identify areas for innovation in the field of Functional Gaussian Process Mixture Models (FunGP) for data clustering. This serves as the foundation for our forthcoming contributions to the discipline.

2.1 Functional Data Analysis

As discussed by Wang et al. [4] in detail, Functional Data Analysis(FDA) represents a pivotal advancement in statistical methodologies, offering a framework for analyzing data that inherently possesses a functional form, such as curves, surfaces, or anything that varies over a continuum. Levitin et al. [5] delineates the foundational principles of FDA, illustrating how data that are essentially functions (such as curves, surfaces, or anything smoothly varying) can be analyzed to uncover underlying patterns and relationships that traditional discrete data analysis methods might miss. FDA's relevance extends across numerous domains where observations are better understood as continuous processes rather than discrete points. The paper emphasizes the importance of this analytical approach in areas where high-dimensional data or complex relationships exist, providing insights that are not readily apparent through other means. By treating data as functions, researchers can employ a variety of sophisticated techniques such as functional principal component analysis, functional regression, and clustering of functional data, offering a nuanced understanding of the data's inherent structure and dynamics. This is what our model focuses on. And since the growth of this field, Ullah and Finch [6] summarize many FDA applications.

2.1.1 Functional Data Smoothing

Functional data refers to data that are naturally represented as functions, curves, or surfaces over a continuous domain, such as time or space. Unlike traditional data that consist of individual points or values, functional data capture the entire trajectory of a phenomenon, allowing for the analysis of patterns, trends, and variations across the entire function.

In the architecture of FunGP, the first layer is a smoother. By using Functional Data Smoothing techniques, users can freely choose various smoothers and adjust specific parameters. For Functional data smoothing methods, you can use traditional smoothing techniques, or you can treat the data as functional data and use FDA techniques. Spline smoothing techniques, such as B-spline wavelets, are widely used for their ability to construct smooth curves that approximate noisy data points effectively. Sakakibara [7] describes using the multiresolution analysis generated by compactly supported cardinal B-spline wavelets for data smoothing, highlighting the implementation algorithm for cubic B-splines and discussing the reasonable orders of truncation for interpolation and decomposition.

On the other hand, as discussed by Jeong, Lu, and Wang [8], wavelet-based techniques offer a flexible framework for dealing with non-stationary data and capturing localized features in functional data. The adaptability of wavelets to varying levels of data resolution makes them ideal for identifying and smoothing out noise while preserving significant data characteristics. These techniques are particularly useful in applications where functional data exhibit complicated transient patterns, such as in manufacturing processes. One notable approach to functional data analysis involves matrix completion techniques, which can distinguish between smooth and rough variations in discretely observed functional data. Descary and Panaretos [9] develop a method that allows for the recovery of both smooth and rough components of functional data by assuming identifiability conditions related to the complementary parsimony of these components. Their technique equates the recovery problem to rank-constrained matrix completion, enabling estimators to be constructed without prior knowledge of bandwidth or rank, providing a novel perspective on handling functional data analysis. Another important contribution to the field is by Yang et al. [10], who propose a nonparametric Bayesian approach for simultaneously smoothing all functional observations. Their method utilizes a Gaussian process prior for the mean function and an inverse-Wishart process prior for the covariance function. This hierarchical framework enables automatic mean-covariance estimation and the retention of systematic patterns across the functional observations, demonstrating the benefits of a Bayesian approach in functional data smoothing.

The advantage of using smoothing techniques in data analysis lies in their ability to enhance the interpretability and quality of the data by reducing noise, highlighting underlying trends, and improving the accuracy of subsequent analyses. Hitchcock, Booth, and Casella [11]

investigated the benefits of pre-smoothing functional data before performing cluster analysis. Their simulation study found that smoothing generally produces more accurate clustering results, suggesting that pre-smoothing can enhance the quality of cluster analysis by providing a cleaner dataset for analysis.

Generalized Cross-Validation (GCV) is a widely used method for selecting the smoothing parameter in smoothing splines and other regularization techniques. It provides a criterion for balancing the trade-off between goodness of fit and model complexity, aiming to prevent overfitting while maintaining model flexibility. Specifically, GCV operates by minimizing a function that estimates the prediction error of the model, allowing it to select an optimal smoothing parameter without requiring a separate validation set. GCV works by estimating the out-of-sample prediction error from the in-sample error, which avoids the need to split the data into training and testing subsets. The GCV score is computed based on the residual sum of squares (RSS) divided by a function of the effective number of parameters in the model. This approach corrects for potential bias caused by overfitting, making it a reliable method for selecting smoothing parameters in various functional data applications [12].

Some examples of functional data smoothing are shown in Figure 2.1

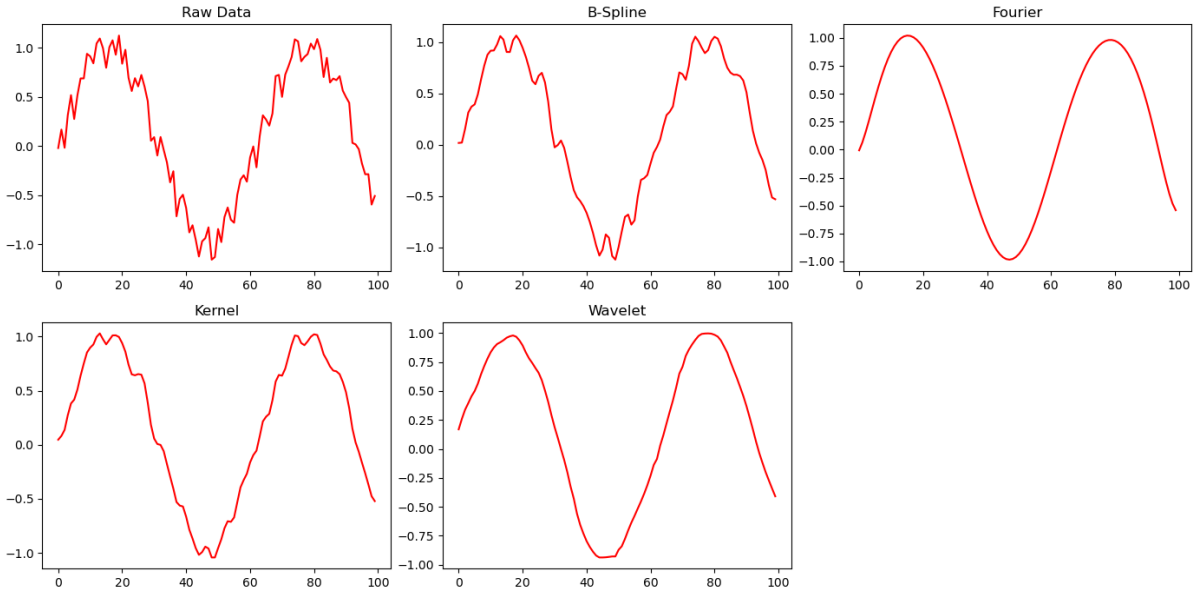


Figure 2.1: Functional Data and Smoothing

2.1.2 Functional Principal Component Analysis

In the architecture of FunGP, the second layer is the projector. This layer uses principal component analysis (PCA), kernel principal component analysis (KPCA), functional principal component analysis (FPCA), and orthogonal decomposition to project the data. FPCA is an extension of traditional PCA that is specifically designed for functional data. FPCA differs from traditional PCA in several key ways due to the nature of the data being analyzed.

Unlike PCA, which operates on fixed-dimensional vectors, FPCA projects functional data onto a set of basis functions (such as B-splines or Fourier) to capture the primary modes of variation. While PCA analyzes a covariance matrix, FPCA works with a covariance function that captures relationships across the entire domain of the functional data. The output of FPCA includes smooth principal component functions that represent the dominant patterns of variation, and corresponding scores that reflect each observation's contribution to these patterns. This approach is necessary for functional data, as it preserves the continuity and underlying structure that PCA cannot handle with its focus on discrete data points. In FPCA, projection basis functions are used to represent continuous functional data as a combination of simpler functions, such as B-splines, Fourier series, or wavelets. These basis functions help approximate smooth curves by expressing each functional observation as a weighted sum of basis functions. The coefficients of these functions capture the key features of the data, allowing FPCA to reduce the infinite-dimensional data into a finite-dimensional form for analysis, while still preserving its continuous nature.

Yuan Xi-xi [13] provides a foundational overview of FPCA by contrasting multivariate statistical analysis methods with FDA approaches. Through examples, the utility and application of FPCA are illustrated, with data analysis facilitated by MATLAB software, emphasizing the method's versatility in extracting significant components from functional data. Hall and Hosseini-Nasab [14] delve into the intrinsic properties of FPCA, elucidating through stochastic expansions how FPCA effectively reduces infinite-dimensional functional data to a finite dimension. Their work quantifies statistical approximation errors and demonstrates how eigenvalue spacings influence FPCA's statistical performance, thereby enhancing understanding of this critical FDA tool. Shang [15] offers a comprehensive review of FPCA, covering its methodological foundations, applications in modeling, forecasting, and classification of functional data. This paper situates FPCA within the broader context of FDA, highlighting its critical role in addressing the challenges posed by datasets represented by curves, images, or shapes. Lin, Wang, and Cao [16] introduce an innovative approach to FPCA that improves interpretability by identifying intervals where functional principal components (FPCs) are significantly non-zero. This penalty-based method yields FPCs with enhanced interpretability compared to traditional FPCA, providing a novel perspective on understanding variations within sample curves.

2.2 Gaussian Process Mixture

In this section, we will focus on the Gaussian Process Mixture model (GPMM). Gaussian Process Model (GP) are a class of powerful non-parametric, probabilistic models used for a variety of purposes, particularly in the domain of machine learning for regression and classification tasks. According to Lemm [17], Mixtures of Gaussian process priors enhance

the flexibility in implementing complex and situation-specific prior information, especially beneficial for tasks with a small number of training data relative to their complexity. This approach generalizes classical regularization approaches by offering more flexibility without significantly increasing computational complexity. Luo and Sun [18] propose a new mixture model, Mixture of Gaussian Processes for Classification (MGPC), that employs logistic functions as likelihood for classification problems, showing significant improvements over existing methods in predictive performance across real-world datasets. This approach illustrates the benefits of mixtures of Gaussian Processes for classification problems.

Wilson [19] proposes a straightforward and general-purpose approach for rapid posterior sampling. This method is particularly compatible with sparse approximations, enabling scalability in both the training and testing phases. Through a series of experiments that examine the statistical properties and practical implications of different sampling schemes, it is demonstrated that decoupled sample paths accurately capture the essence of Gaussian process posteriors with significantly reduced computational overhead. Stachniss et al. [20] use the GP model in real-world applications. Due to the chaotic nature of gas dispersal, accurately predicting gas concentration at specific locations requires sophisticated modeling approaches. This paper addresses the issue by proposing a sparse Gaussian process mixture model, designed to represent both the smooth background signal and the areas with high concentrations effectively. The integration of training data sparsification into an EM (Expectation-Maximization) procedure allows for learning the mixture components and the gating function efficiently. For batch processes exhibiting different variance structures throughout the batch's duration, Ou and Martin [21] believe employing a single Gaussian process may not suffice for accurate modeling. Mixtures of Gaussian processes provide a solution by combining outputs from each component, weighted by a Gaussian kernel gating network. This approach addresses computational cost issues inherent in Gaussian processes and demonstrates advantages over single Gaussian process models.

In conclusion, Gaussian Process Mixtures offer a versatile and powerful approach for dealing with complex data modeling challenges across various domains, from regression to classification, showcasing significant advancements over traditional methods. These models provide a pathway towards more nuanced and adaptable statistical analyses, accommodating non-linearities, heterogeneity, and computational constraints inherent in modern data sets. Incorporating a GPMM as the third layer in the FunGP model for clustering purposes is a strategic choice that leverages the inherent strengths of GPMMs to enhance the model's performance and capability.

2.3 Clustering

Data clustering is a fundamental technique in the field of machine learning and data analysis that involves grouping a set of objects in such a way that objects in the same group (or cluster) are more similar to each other than to those in other groups. Its main goal is to discover the inherent structure within data, identify patterns, and simplify the analysis by treating clusters as individual entities. Clustering is a method of unsupervised learning, and it is typically used for exploratory data analysis to find hidden patterns or groupings in data. According to Everitt [22], cluster analysis organizes multivariate data into subgroups to reveal patterns or structures, applicable in medicine, psychology, and market research. This methodology focuses on practical aspects rather than mathematical intricacies. Jain [23] discusses the exploratory nature of clustering and its rich history across scientific fields. Data clustering continues to evolve, with significant contributions across multiple domains. These advancements are essential for exploring and understanding the vast amounts of data generated in various fields, from scientific research to commercial applications. The continual development of clustering methodologies and algorithms reflects the dynamic nature of this field and its critical role in data analysis.

2.3.1 Functional Data Clustering

This section focuses on functional data clustering. Zhang and Parnell [24] provide a comprehensive review of original clustering methods for functional data, proposing a systematic taxonomy to understand the connections and differences among existing methods. James and Sugar [1] develop a model-based procedure for clustering functional data, especially suitable for datasets with sparsely sampled curves. This method not only assigns cluster memberships but also predicts missing curve portions, offering tools for model evaluation. Akeweje and Zhang [25] also proposed a clustering method called GPMix, which shares a similar key idea with our approach: projecting functional data onto randomly generated functions. However, their algorithm uses the method of moments for parameter estimation. Their method also achieves effective clustering of functional data. Jacques and Preda [3] review clustering techniques for functional data, classifying them into four categories based on their approach to handling curve data. This survey includes a software review and real data application, demonstrating the algorithms' applicability across various fields.

Here, we will discuss in detail three state-of-art clustering methods used to compare our model FunGP. Bouveyron and Jacques [26] developed a novel approach for clustering functional data, specifically time series, through the adaptation of the high dimensional data clustering (HDDC) method to a functional context. Their method, termed FunHDDC, is grounded in a functional latent mixture model that accommodates functional data within

group-specific functional subspaces. This approach enables the handling of high-dimensional functional data by fitting it into parsimonious models that are both efficient and interpretive. Soueidatt [27] proposed a novel approach in R named funClustering, this package proposes a model-based clustering algorithm for multivariate functional data. The parametric mixture model, based on the assumption of normality of the principal components resulting from a multivariate functional PCA, is estimated by an EM-like algorithm. The main advantage of the proposed algorithm is its ability to take into account the dependence among curves. Bouveyron and Jachques [28] introduced another model-based functional data clustering method named funFEM. This package would be particularly useful for statisticians and data scientists working with data that vary over a continuum, such as time-series data or growth curves, providing tools for identifying homogenous groups within such data based on their functional characteristics.

3 Model and Methodology

This section introduces the Functional Gaussian Process Mixture Model (FunGP), an approach aimed at improving the clustering of functional data. This approach is based on the Gaussian Process (GP) which is a stochastic process where any finite collection of random variables follows a multivariate Gaussian distribution. The distribution of a Gaussian process is the joint distribution of all those (infinitely many) random variables, and as such, it is a distribution over functions with a continuous domain, e.g. time or space. It's widely used in machine learning and statistics, particularly for modeling distributions over functions. Our objective is to develop a user-friendly model characterized by its adjustable parameters. As discussed in background research, functional data, characterized by its complexity, poses unique challenges for analysis and interpretation. In addition, the application prospects in the field of functional data analysis are vast and varied, touching upon numerous disciplines. The complexity of functional data often makes it challenging for scholars from various fields to directly utilize existing clustering models. Our model is crafted with the intent to democratize the use of advanced clustering techniques for functional data, ensuring that scholars and practitioners from diverse fields can easily apply it to their unique datasets.

To achieve this, as illustrated in Figure 3.1, our model is structured into three distinct layers. Each layer allows users to freely select the parameters they need, and each layer is completely separate to allow users to complete specific tasks (such as smoothing and clustering). The intricacies of these layers and their functionalities will be elaborated upon in subsequent sections. We also provide specific usage instructions. Smoother is used to smooth the data, projector is used to map the data to eigenvalues, and Unigmm is for the final clustering based on the Gaussian mixture model. Specific information will be shown in the following sections.

In the previous section, we discussed the use of smoothers and projectors for functional data processing. Smoothness is a crucial aspect of functional data analysis (FDA) as functional data typically represent underlying continuous processes. Ensuring smoothness not only helps in filtering out noise but also facilitates capturing the true underlying process.



Figure 3.1: Model Structure

Smoothness is important for maintaining the continuity and interpretability of the data, ensuring that statistical methods are correctly applied, and providing a foundation for derivative analysis, such as calculating rates of change. In addition, projection is also important because it reduces dimensionality, making data more manageable and processable. By extracting meaningful features that capture the essential characteristics of the data, projection can also filter out noise and provide a clearer, more interpretable representation of the underlying process. In addition, it simplifies computations by converting infinite-dimensional problems into finite ones and facilitates the application of statistical modeling techniques such as regression and clustering.

We provide four types of smoothers, namely B-spline, Fourier, kernel, and wavelet. In addition, we provide four types of projectors, namely orthogonal basis expansion (expansion), Functional Principal Component Analysis (FPCA), Principal Component Analysis (PCA), and Kernel Principal Component Analysis (KPCA). In the projector, We provide B-spline, Fourier, kernel, and wavelet methods for FPCA, and B-spline, Fourier, wavelet for expansion, along with standard PCA and KPCA.

3.1 Initialization

This section is about initialization before use. We need to load some dependent libraries.

```
1 import numpy as np
2 import skfda
3 import pywt
4 import scipy
5 import sklearn
6 from smoother.gpmix_smoother import smoother
7 from projector.gpmix_projector import projector
8 from unigmm.gpmix_unigmm import unigmm
```

After completing this step, we can go down to each layer of the structure.

3.2 Smoother

The first layer in our model FunGP is the smoother. The primary objective of the smoother layer within FunGP is to preprocess raw data, transforming it into a smooth representation suitable for functional data analysis. This process involves converting discrete data points into a continuous function, represented as FDataGrid, which is a structured format for functional data. This format is proposed by Ramos-Carreno et al. [29] in their python package scikit-fda. The operation performed in this step is only to represent the raw data using a continuous basis function.

Let us denote by $\mathcal{T}$ a closed interval within $\mathbb{R}$. Consider X as a random function situated within a probability space defined by $(\Omega, \mathcal{F}, \text{Pr})$. We assume that the X is generated from the Gaussian Process(GP). A Gaussian Process(GP) is a collection of random variables, any finite number of which follows a joint Gaussian distribution. A Gaussian Process is defined as $GP(\mu_k, \Sigma_k)$, where μ_k is the mean function, representing the expected value of the process at any point and Σ_k is the covariance function (or kernel), which defines the covariance between the values at any two points [30]. This assumption justifies the use of PCA for dimensionality reduction in the next section, as PCA captures the major modes of variation in the data, which in this case are expected to follow a Gaussian-like process. Additionally, the assumption that the data originates from a GP supports the use of Gaussian Mixture Model(GMM) for clustering, as GMM is effective in identifying clusters within data that exhibit Gaussian-like behavior, allowing for a probabilistic approach to clustering that is well-suited to functional data. It is important to note that this is an assumption, not a proven fact, but it provides a sound theoretical basis for the application of the rest part of FunGP. The assumption helps guide the selection of methods and simplifies the modeling of complex, high-dimensional functional data.

In this context, a sample function $x = X(\cdot, \omega)$, where ω is an outcome in Ω , belongs to the space $\mathcal{H}(\mathcal{T}, \mathbb{R})$. The sample function x represents a specific instantiation of the random function X , observed at the outcome ω . Furthermore, the function $x(t)$ can be viewed as a random realization of the random variable $X(t, \cdot)$, frequently referred to as X_t in literature.

In real application, the value of x_i at any given point t within $\mathcal{T}$, expressed as $y_i(t)$, can include an additive error component, modeled as $y_i(t) = x_i(t) + \varepsilon_i(t)$. Here, ε_i signifies the noise process characterized by a mean of zero, $E[\varepsilon_i(t)] = 0$, and a variance denoted by $E[\varepsilon_i^2(t)] = \sigma^2(t)$. In this case, we generally assume that if the sample function x_i is from $GP(\mu_k, \Sigma_k)$, the conditional distribution of the sample path is multivariate Gaussian.

Given the raw data $\{y_i(t) : i = 1, \dots, n\}$. The first step is smoothing. Our model provides four different smoothers, namely B-spline, Fourier, wavelet, and kernel. Select different

smoothers according to the user's needs and then convert the original raw data into smoothed data $\{\hat{y}(t_i) : i = 1, \dots, n\}$.

Given a set of raw data $\{y_i(t) : i = 1, \dots, n\}$, the initial phase in processing this data involves smoothing. This step is crucial for reducing noise and capturing the underlying patterns in the data more effectively. Our model accommodates this need by offering four distinct smoothing techniques: B-spline, Fourier, wavelet, and kernel. Each of these smoothers is tailored to different types of data characteristics and user requirements. B-spline smoothing is particularly beneficial for data that requires flexible, piecewise polynomial representation, making it ideal for capturing smooth curves with variable rates of change. Fourier smoothing is best suited for periodic or cyclic data, utilizing trigonometric functions to model oscillatory behavior effectively. Wavelet smoothing offers a multi-resolution approach, allowing for the analysis of data at various scales, which is especially useful for capturing abrupt changes or non-stationary features in the data. Kernel smoothing employs a non-parametric approach, using a weighted average of nearby points to smooth the data, which is versatile for a wide range of applications. The usage examples of each smoother are listed below. We can freely adjust the specific parameters.

```
1 smoother1 = smoother(smoother = 'bspline', smoother_args = {'
    degree': 3, 'n_basis': 50})
2 fdate_smooth1 = smoother1.fit(data)
```

When smoother is B-spline, the optional parameters are *degree* and *n_basis*, which are the number of degrees and the number of basis functions.

$$B_{i,1}(x) = 1 \quad \text{if } t_i \leq x < t_{i+1}, \quad 0 \text{ otherwise}$$

$$B_{i,k}(x) = \frac{x - t_i}{t_{i+k} - t_i} B_{i,k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} B_{i+1,k-1}(x)$$

where k represents degree and i represents *n_basis*.

```
1 smoother2 = smoother(smoother = 'fourier', smoother_args = {'
    n_basis': 10, 'period': 3})
2 fdate_smooth2 = smoother2.fit(data)
```

when smoother is Fourier, the optional parameters are *n_basis* and *period*, which are the

number of basis functions and how many periods of Fourier function.

$$\begin{aligned}\phi_0(t) &= \frac{1}{\sqrt{2}} \\ \phi_{2n-1}(t) &= \frac{\sin\left(\frac{2\pi n}{T}t\right)}{\sqrt{\frac{T}{2}}} \\ \phi_{2n}(t) &= \frac{\cos\left(\frac{2\pi n}{T}t\right)}{\sqrt{\frac{T}{2}}}\end{aligned}$$

where t represents period and n represents n_basis .

```
1 smoother3 = smoother(smoother = 'kernel', smoother_args = {'
    bandwidth': 0.01})
2 fdate_smooth3 = smoother3.fit(data)
```

when smoother is kernel, the optional parameters are *bandwidth*, which are window width of the kernel (also called h or bandwidth). In this case, we create the matrix $\hat{H}$ as explained below:

$$\hat{H}_{i,j} = \frac{K\left(\frac{d(x_j - x'_i)}{h}\right)}{\sum_{k=1}^n K\left(\frac{d(x_k - x'_i)}{h}\right)}$$

For smoothing, $\{x_1, \dots, x_n\}$ are the points with known value and $\{x'_1, \dots, x'_m\}$ are the points for which it is desired to estimate the smoothed value. The distance d is the absolute value function. $K(\cdot)$ is a kernel function and h is the bandwidth. We use a normal kernel as the kernel function which is a standard normal density function [31].

```
1 smoother4 = smoother(smoother = 'wavelet', smoother_args = {'
    wavelet': 'db4', 'level': 4})
2 fdate_smooth4 = smoother4.fit(data)
```

when smoother is wavelet, the optional parameters are *wavelet* and *level*, which are which kind of wavelet and decomposition level. In this case, we compute multilevel discrete wavelet transform of data.

Algorithm 1 describes the specific process of the smoother.

Algorithm 1: Smoother

Input: Raw data $\mathcal{D} = \{y_i(t) : i = 1, \dots, n\}$, smoother type, smoother arguments.

Output: Smoothed data $\{\hat{y}_i\}_{i=1}^n$.

- 1 Obtain estimated smooth functions $\{\hat{y}_i\}_{i=1}^n$ based on the selected smoother and arguments.
-

Figure 3.2 illustrates the application of four different smoothing techniques—B-spline, Fourier, wavelet, and kernel—on the same dataset, which is characterized by its intricate, highly fluctuating nature. This visual comparison reveals that for data with such complex patterns and strong fluctuations, the wavelet and kernel smoothers tend to yield more effective results. The wavelet smoother, with its capability to handle sudden changes and non-stationary behavior in the data, adeptly manages to capture the intricate details and variations within the dataset. Its multi-resolution analysis provides a nuanced approach to dealing with the data's variability, making it particularly suited for this kind of dataset. On the other hand, the kernel smoother, through its non-parametric nature, offers a flexible approach to smoothing, adjusting locally to the density of data points. This adaptability allows it to effectively trace the data's path, smoothing out noise while retaining the critical features of the underlying function. In contrast, B-spline and Fourier smoothers, while useful in many contexts, may not fully capture the essence of strongly fluctuating data due to their inherent characteristics. B-splines might oversimplify complex variations, and Fourier smoothers could struggle with non-periodic fluctuations. Being able to freely choose smoother gives users a lot of freedom, and this step can also be used alone to smooth data.

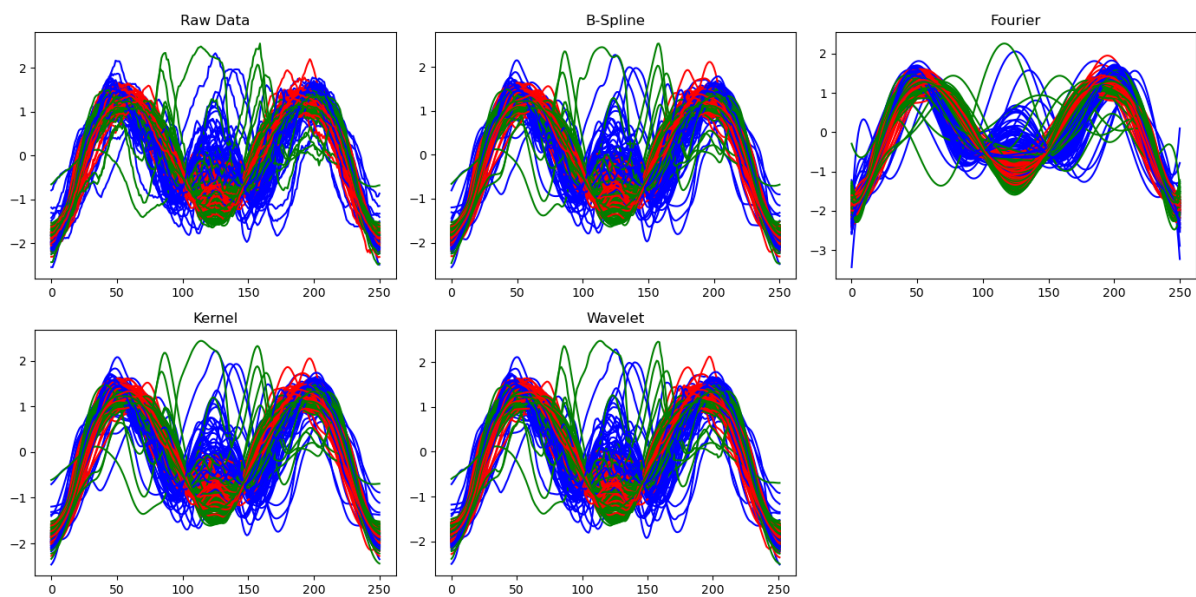


Figure 3.2: Smoother Example

Note that Figure 3.2 uses the real data set Arrowhead, and the specific content of the data will be described in Chapter 4: Evaluation. The different color stands for different classes the data belongs to. The specific parameters of smoother used in the figure are as follows:

```
1 bspline_smoother = smoother(smoother = 'bspline', smoother_args =
    {'degree': 3, 'n_basis': 50})
2 fourier_smoother = smoother(smoother = 'fourier', smoother_args =
    {'n_basis': 10, 'period': 3})
```

```

3 kernel_smoother = smoother(smoother = 'kernel', smoother_args = {'
    bandwidth': 0.01})
4 wavelet_smoother = smoother(smoother = 'wavelet', smoother_args =
    {'wavelet': 'db4', 'level': 4})

```

3.3 Projector

The second layer of FunGP is the projector. The primary objective of the projector layer within FunGP is to calculate the projection coefficients.

If we consider a sample function x_i belongs to the k th cluster, then it is a random realization of the k th random function X_k . Assume that Z is the hidden cluster label variable where the probability of a function belonging to cluster k is denoted by $\Pr(Z = k) = \pi_k$. These probabilities satisfy two conditions: $\pi_k > 0$ and $\sum_{k=1}^K \pi_k = 1$. In the context where $Z = k$, the conditional distribution of the random function X is that of X_k . In this case, we assume each random function X_k is a Gaussian process. Denote this to $\mathcal{GP}(\mu_k, \Sigma_k)$, where μ_k is the mean function, and Σ_k is the covariance function. According to the spectral theorem for compact self-adjoint operators [32], the covariance function $\Sigma_k(s, t)$ of X_k has series representation

$$\Sigma_k(s, t) = \sum_{v=1}^{\infty} \lambda_{kv} b_{kv}(s) b_{kv}(t),$$

where $\lambda_{k1} \geq \lambda_{k2} \geq \dots \geq 0$ are the eigen-values, and $\{b_{kv}\}_{v \in \mathbb{N}}$ are the orthonormal eigen-functions. Then a sample function x_i , with $z_i = k$, has the following decomposition:

$$x_i(t) = \mu_k(t) + \sum_{v=1}^{\infty} a_{ikv} b_{kv}(t)$$

where $a_{ikv} = \langle x_i - \mu_k, b_{kv} \rangle$ is principle score associated with the eigenfunction b_{kv} . Note that X_k is a Gaussian process so that a_{ikv} has a Gaussian distribution. However, in practical scenarios, the true cluster labels of sample functions are unknown, posing a challenge for direct decomposition. To navigate this, we can let μ denote the population mean function: $\mu(t) = \sum_{k=1}^K \pi_k \mu_k(t)$. Let $\{\beta_v\}_{v \in \mathbb{N}}$ denote orthonormal basis for the Hilbert space $L^2(\mathcal{T}, R)$. By projecting the difference between x_i and the population mean μ , represented as $x_i - \mu$, onto this basis, we can express any sample function in terms of its projection coefficients relative to the orthonormal basis:

$$x_i(t) = \mu(t) + \sum_{v=1}^{\infty} \alpha_{iv} \beta_v(t),$$

where $\alpha_{iv} = \langle x_i - \mu, \beta_v \rangle$ is the projection coefficient onto the basis function β_v . So that we

obtain projection coefficients.

Our package provides PCA, KPCA, FPCA, and orthogonal expansion as the projection method. And we also provide B-spline, Fourier, wavelet, and monomial as the basis. This can be combined into more than a dozen different projection combinations. Here is only one example that uses FPCA as the projection method and B-spline as the projection basis:

```
1 projector = projector(projection_method = 'fpca', projection_args
    = {'n_components': 15, 'basis': 'bspline', 'n_basis': 15, '
      order': 3})
2 coefficients = projector.fit(fdate_smooth)
```

The methods mentioned above are all commonly used dimensionality reduction techniques, among which PCA and KPCA focus on finite-dimensional data (vectors), while FPCA is used for infinite-dimensional data (functions).

Algorithm 2 describes the detail of the projector:

Algorithm 2: Projector

Input: Smoothed data $\{\hat{y}_i\}_{i=1}^n$, projector type, projector arguments.

Output: Projection coefficients $\{\alpha_{iv}\}_{i=1}^n$.

- 1 Estimate the population mean function μ and the n sample functions $\{x_i\}_{i=1}^n$
- 2 Calculate the n projection coefficients:

$$\alpha_{iv} = \langle x_i - \mu, \beta_v \rangle, \quad 1 \leq i \leq n.$$

- 3 **return** $\{\alpha_{iv}\}_{i=1}^n$
-

3.4 Univariate Gaussian Mixture Model: UniGMM

The third and last layer of the FunGP termed the uniGMM layer, is dedicated to performing the clustering of data by leveraging a Gaussian Process Mixture Model. This sophisticated approach builds on the foundational work conducted in the preceding layers of the model.

As we mentioned in the previous projector section, after decomposing the principal components $\{a_{1kv}, a_{2kv}, \dots, a_{nkv}\}$ of a Gaussian process, we know these values follow a univariate Gaussian distribution. Therefore, conditioning on $z_i = k$, the distribution of α_{iv} follows univariate Gaussian distribution with mean $\langle \mu_k - \mu, \beta_v \rangle$ and variance $\sum_{l=1}^{\infty} \lambda_{kl} \langle b_{kl}, \beta_v \rangle^2$. So we can say that the projection coefficients $\{\alpha_{1v}, \alpha_{2v}, \dots, \alpha_{nv}\}$ follow a univariate Gaussian mixture distribution:

$$\alpha_{iv} \sim \sum_{k=1}^K \pi_k \mathcal{N} \left(\langle \mu_k - \mu, \beta_v \rangle, \sum_{l=1}^{\infty} \lambda_{kl} \langle b_{kl}, \beta_v \rangle^2 \right).$$

We then fit a Gaussian mixture model to the projection coefficients, using a mixture model with K components on the principal components $\{\alpha_{1v}, \alpha_{2v}, \dots, \alpha_{nv}\}$. We fit the model by using the EM algorithm, here we use the implementation of GMM in the Python library Scikit-learn [33]. The fitted model is denoted as $\sum_{k=1}^K \pi_{vk} \phi(\alpha; u_{vk}, \gamma_{vk}^2)$, where each Gaussian density ϕ has a mean u_{vk} and a variance γ_{vk}^2 . The cluster membership matrix $\mathbf{M}_v$ is then obtained as follows:

$$m_{ik}^v = \frac{\pi_{vk} \phi(\alpha_{iv}; u_{vk}, \gamma_{vk}^2)}{\sum_{j=1}^K \pi_{vj} \phi(\alpha_{iv}; u_{vj}, \gamma_{vj}^2)}, \quad 1 \leq i \leq n, 1 \leq k \leq K.$$

A cluster membership matrix is a matrix used in clustering algorithms to represent the assignment of data points to clusters. Each row of the matrix corresponds to a data point, and each column represents a cluster. The entries in the matrix indicate the degree to which each data point belongs to a particular cluster [34].

Each membership matrices $\{M_1, M_2, \dots, M_v\}$ represents each curve x_i and their corresponding membership values to each cluster. Then we construct a binary membership indicator matrix $\mathbf{B}_v$ via:

$$\mathbf{b}_{ik}^v = \begin{cases} 1, & \text{if } k = \arg \max_{1 \leq j \leq K} \{m_{ij}^v\}; \\ 0, & \text{otherwise.} \end{cases}$$

A binary membership indicator matrix is a specific type of matrix used in clustering algorithms to represent the assignment of data points to clusters in hard clustering scenarios which means each data point is assigned to exactly one cluster. It is a binary matrix where each element takes a value of 0 or 1, indicating whether a data point belongs to a particular cluster.

Then we need to calculate the weight $w_v(> 0) : \sum_{v=1}^V w_v = 1$.

The concept of c -separation, as introduced by Dasgupta [35], two p -variate Gaussian distributions $\mathcal{N}(\mathbf{u}_k, \Sigma_k)$ and $\mathcal{N}(\mathbf{u}_j, \Sigma_j)$ are c -separated if the distance between their mean vectors is at least c times the square root of p times the maximum largest eigen-value of their covariance matrices:

$$\|\mathbf{u}_k - \mathbf{u}_j\| \geq c \sqrt{p \times \max\{\lambda_{\max}(\Sigma_k), \lambda_{\max}(\Sigma_j)\}}$$

where $\lambda_{\max}(\Sigma)$ is denotes the largest eigenvalue of Σ . A mixture of Gaussian distributions is

c-separated if its component Gaussian distributions are pairwise c-separated. Maitra and Melnykov [36] extend this concept by defining the probability of misclassification between two mixture components. For the two Gaussians $\mathcal{N}(\mathbf{u}_k, \Sigma_k)$ and $\mathcal{N}(\mathbf{u}_j, \Sigma_j)$ with mixing proportions π_k and π_j , define $\omega_{j|k}$ as the probability that an instance from the k th mixture component $\mathcal{N}(\mathbf{u}_k, \Sigma_k)$ is misclassified into the j th mixture component:

$$\omega_{j|k} = \Pr(\pi_k \phi(\mathbb{X}; \mathbf{u}_k, \Sigma_k) < \pi_j \phi(\mathbb{X}; \mathbf{u}_j, \Sigma_j) \mid \mathbb{X} \sim \mathcal{N}(\mathbf{u}_k, \Sigma_k)).$$

Although the analytical calculation of $\omega_{j|k}$ is not straightforward due to the complexity of integrating over the multivariate Gaussian density, numerical methods provide a practical alternative for estimation, especially in the case of univariate Gaussian Mixture Models (uniGMMs) [36].

Then we can calculate the total probability of misclassification ω_v :

$$\omega_v = \sum_{k=1}^K \pi_{vk} \left(\sum_{j \neq k} \omega_{j|k} \right)$$

Then the weight w_v for the v th base clustering is:

$$w_v = \frac{\omega_v^{-1}}{\sum_{j=1}^V \omega_j^{-1}}$$

So that we apply a multivariate clustering method on the affinity matrix

$\mathbf{A} = \sum_{v=1}^V w_v \mathbf{B}_v \mathbf{B}_v^T$ and return the identified cluster labels $\{z_i\}_{i=1}^n$. Note that this multivariate clustering method can be freely selected. In this dissertation, spectral clustering is mainly used. Here V comes from the number of basis functions $\{\beta_v\}_{v=1}^V$. For each $1 \leq v \leq V$, we fit a univariate GMM to the projection coefficients $\{\alpha_{1v}, \dots, \alpha_{nv}\}$, and then extract the cluster membership matrix $\mathbf{M}_v = [m_{ik}^v]_{n \times K}$ from the estimated model and so that we finally get affinity matrix $\mathbf{A}$. The affinity matrix partially reflects the clustering results, as it is derived from a combination of the membership matrix and the overlap produced by the univariate Gaussian Mixture Model (uniGMM). This combination helps capture the similarities between data points based on their probabilistic cluster assignments, providing a representation that can reveal the underlying cluster structure.

Below is an example of a uniGMM:

```
1 gmm = unigmm(n_components = n_labels)
2 gmms = gmm.fit(coefficients)
3 affinity_matrix = gmm.calculate_weights()
4 sc = SpectralClustering(n_clusters = n_labels, affinity='
    precomputed', assign_labels='discretize')
```

```

5 | sc.fit(affinity_matrix)
6 | labels = sc.labels_

```

Algorithm 3 describes the details of the uniGMM:

Algorithm 3: UniGMM

Input: Projection coefficients $\{\alpha_{iv}\}_{i=1}^n$, the number of clusters K , uniGMM arguments.

Output: The learned cluster labels $\{z_i\}_{i=1}^n$.

- 1 **for** $v = 1, \dots, V$ **do**:
- 2 Train a univariate GMM from the projection coefficients $\{\alpha_{iv}\}_{i=1}^n$, denoted by $\sum_{k=1}^K \pi_{vk} \phi(\alpha; u_{vk}, \gamma_{vk}^2)$.
- 3 Obtain the cluster membership matrix $\mathbf{M}_v$:

$$m_{ik}^v = \frac{\pi_{vk} \phi(\alpha_{iv}; u_{vk}, \gamma_{vk}^2)}{\sum_{j=1}^K \pi_{vj} \phi(\alpha_{iv}; u_{vj}, \gamma_{vj}^2)}, \quad 1 \leq i \leq n, 1 \leq k \leq K.$$

- 4 Construct a binary membership indicator matrix $\mathbf{B}_v$:

$$\mathbf{b}_{ik}^v = \begin{cases} 1, & \text{if } k = \arg \max_{1 \leq j \leq K} \{m_{ij}^v\}; \\ 0, & \text{otherwise.} \end{cases}$$

- 5 Calculate the weight $w_v(> 0) : \sum_{v=1}^V w_v = 1$.
 - 6 **end for**
 - 7 Apply a multivariate clustering method on the affinity matrix $\mathbf{A} = \sum_{v=1}^V w_v \mathbf{B}_v \mathbf{B}_v^T$ and get the identified cluster labels $\{z_i\}_{i=1}^n$.
 - 8 **return** $\{z_i\}_{i=1}^n$
-

It can be found that this algorithm is based on a loop, so the previous projector reduced the dimension of the data, which reduced the calculation amount of this step a lot. The specific evaluation of the algorithm will be carried out in the next section.

4 Evaluation

In this section, we delve into a detailed evaluation of our proposed clustering algorithm, setting it against the backdrop of contemporary state-of-the-art methods in the field, namely funFEM [28], funHDDC [26], and funCluster [27]. This comparative analysis spans both simulated and real-world datasets, employing two critically acclaimed quantitative metrics for assessing clustering performance: the Adjusted Rand Index (ARI) and the Adjusted Mutual Information (AMI).

The evaluation of our model encompasses an extensive analysis across both simulated and real-world datasets, emphasizing its applicability and versatility across a broad spectrum of domains. We meticulously selected a total of 12 real-world datasets from various fields such as medicine, biology, climate science, economics, and more, ensuring a comprehensive assessment of the model’s performance in diverse settings. All datasets used in our evaluation are sourced from open-access platforms, ensuring transparency, reproducibility, and the facilitation of future research endeavors.

4.1 Simulation Scenarios

In this section, we will evaluate our algorithm against state-of-the-art clustering methods, specifically within the framework of simulated scenarios. The functional simulated data we used based on publications [37–40] with an added layer of complexity introduced through random noise. These scenarios employ the exponentiated quadratic kernel function for Gaussian Process (GP) simulation, setting σ to 1 to ensure consistency across simulations.

Figure 4.1 presents a visual representation of seven distinct sets of simulated data scenarios. The utilization of different colors within these visualizations serves to delineate individual curves, each corresponding to a separate cluster.

To further examine the resilience and performance of our model in the presence of noise, we systematically augmented our simulated datasets with noise. This was accomplished by integrating scaled random samples drawn from a normal distribution, carefully calibrated to

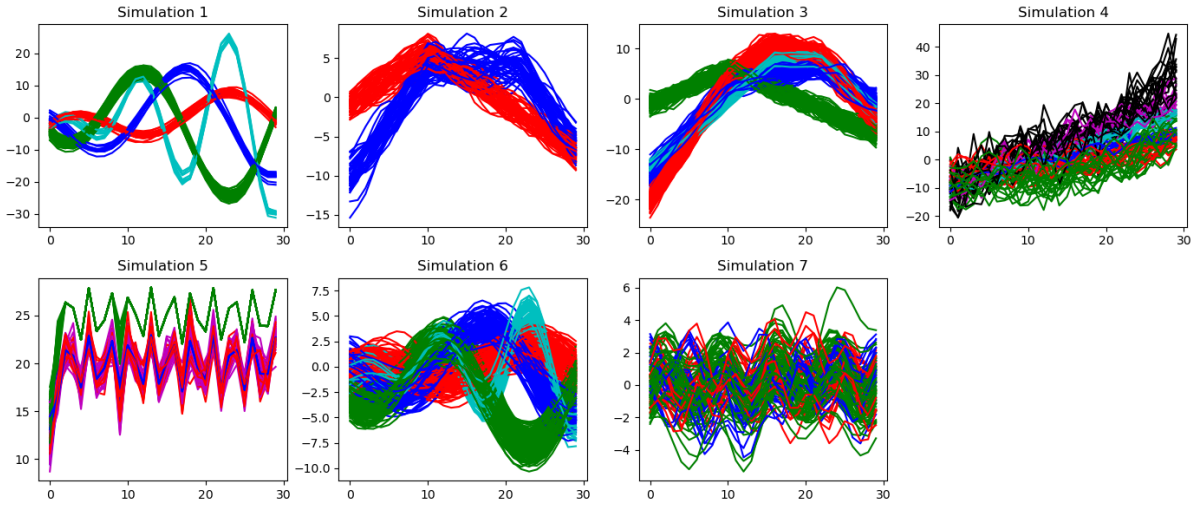


Figure 4.1: Simulation Scenarios

ensure that the induced fluctuations did not alter the original data by more than 10%. Figure 4.2 offers a comparative visualization of these seven noised simulated data scenarios. Similar to the previous figure, different colors are employed to signify the membership of each curve to distinct clusters, facilitating an evaluation of the model's effectiveness in noise management.

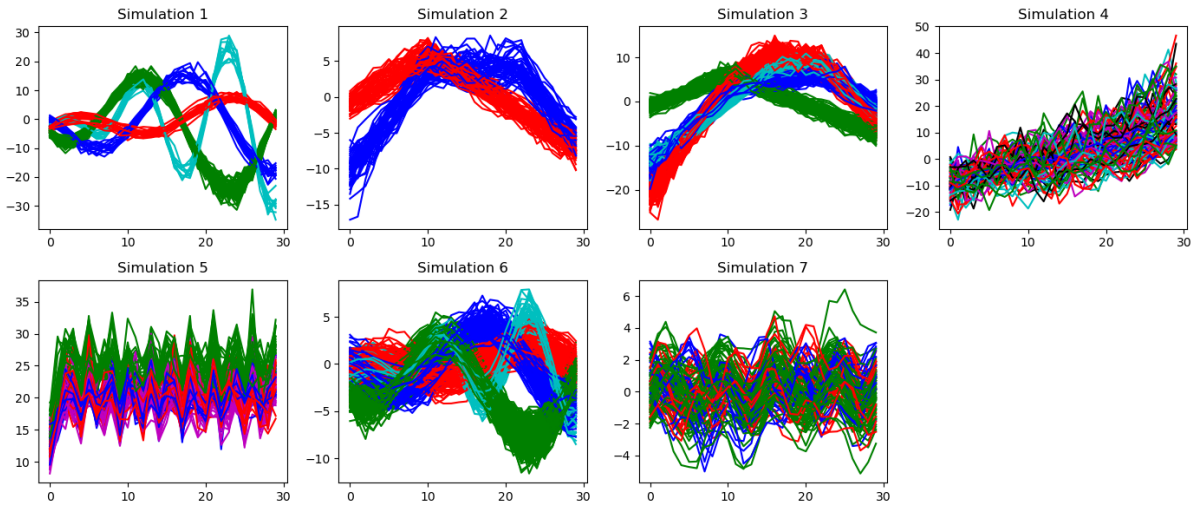


Figure 4.2: Noised Simulation Scenarios

Deliberately introducing noise into a simulated data set serves a dual purpose. First, it simulates the complexity of real-world data, where noise is an unavoidable component, thereby testing the applicability of the algorithm in real-world scenarios. Second, it allows for a nuanced evaluation of a model's ability to distinguish between real data variation and random noise, a key factor in the accuracy and reliability of clustering results.

4.1.1 Simluation Scenarios Description

This section describes seven sets of simulation scenarios in detail.

Scenario 1: As discussed in the dataset provided by Delaigle and Hall [37]. The number of cluster is $K = 4$. An independent sample of $N = 100$ univariate curves is simulated for 30 observations. We define the mixture coefficients $\pi_k = [0.2, 0.2, 0.5, 0.1]$. A description of each mean curve and GP simulation is described below:

$$\begin{aligned}\mu_1(t) &= -8 \sin(t) \log(t + 2), & C_1 : Y(t) &= \mathcal{GP}(\mu_1(t), k(t, t)), \\ \mu_2(t) &= 4 \cos(t) \log(t + 0.5), & C_2 : Y(t) &= \mathcal{GP}(\mu_2(t), k(t, t)), \\ \mu_3(t) &= -2 - 4 \cos(t) \sqrt{5t + 0.5}, & C_3 : Y(t) &= \mathcal{GP}(\mu_3(t), k(t, t)), \\ \mu_4(t) &= 5 \cos(2t) \log(t + 0.5) \sqrt{t + 0.5}. & C_4 : Y(t) &= \mathcal{GP}(\mu_4(t), k(t, t)).\end{aligned}$$

Scenario 2: As discussed in the dataset provided by Garcia-Escudero and Gordaliza [38]. The number of cluster is $K = 2$. An independent sample of $N = 100$ univariate curves is simulated for 30 observations. We define the mixture coefficients $\pi_k = [0.5, 0.5]$. We add some Gaussian noise $\epsilon(t)$, where the probability density for the noise is $\epsilon(t) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}$. In this case, $\mu = 0$ and $\sigma = 0.08$. A description of each mean curve and GP simulation is described below:

$$\begin{aligned}\mu_1(t) &= 6 - |t - 7|, & C_1 : Y(t) &= \mathcal{GP}(\mu_1(t), k(t, t)) + \mathcal{GP}(\mu_2(t), k(t, t)) + c(t), \\ \mu_2(t) &= 6 - |t - 15|. & C_2 : Y(t) &= \mathcal{GP}(\mu_2(t), k(t, t)) + c(t).\end{aligned}$$

Scenario 3: As discussed in the dataset provided by Garcia-Escudero and Gordaliza [38]. The number of cluster is $K = 4$. An independent sample of $N = 500$ univariate curves is simulated for 30 observations. We define the mixture coefficients $\pi_k = [0.3, 0.3, 0.2, 0.2]$. We add some Gaussian noise $\epsilon(t)$, where the probability density for the noise is $\epsilon(t) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}$. In this case, $\mu = 0$ and $\sigma = 0.08$. A description of each mean curve and GP simulation is described below:

$$\begin{aligned}\mu_1(t) &= 6 - |t - 11|, & C_1 : Y(t) &= \mathcal{GP}(\mu_2(t), k(t, t)) + \mathcal{GP}(\mu_3(t), k(t, t)) + \sqrt{0.1} * c(t) - 5, \\ \mu_2(t) &= 6 - |t - 7|, & C_2 : Y(t) &= \mathcal{GP}(\mu_1(t), k(t, t)) + \mathcal{GP}(\mu_2(t), k(t, t)) \\ & & & + \mathcal{GP}(\mu_3(t), k(t, t)) + \sqrt{0.5} * c(t) - 5, \\ \mu_2(t) &= 6 - |t - 15|. & C_3 : Y(t) &= \mathcal{GP}(\mu_2(t), k(t, t)) + \sqrt{10} * c(t), \\ & & C_4 : Y(t) &= \mathcal{GP}(\mu_1(t), k(t, t)) + \mathcal{GP}(\mu_3(t), k(t, t)) + \sqrt{0.5} * c(t).\end{aligned}$$

Scenario 4: As discussed in the dataset provided by Golovkine, Klutchnikoff and Patilea [39]. The number of cluster is $K = 4$. An independent sample of $N = 100$ univariate curves is simulated for 30 observations. We define the mixture coefficients $\pi_k = [1/6, 1/6, 1/6, 1/6, 1/6, 1/6, 1/6]$. We add some Gaussian noise $\epsilon(t)$, where the

probability density for the noise is $\epsilon(t) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}$. In this case, $\mu = 0$ and $\sigma = 0.08$. A description of each mean curve and GP simulation is described below:

$$\begin{aligned}
C_1 : Y(t) &= \mathcal{GP}(\cos(t/10), k(t, t)) + \mathcal{GP}(\sin(1 + t/10), k(t, t)) + \epsilon(t) + t - 21/2, \\
C_2 : Y(t) &= \mathcal{GP}(\cos(2 * t/10), k(t, t)) + 2 * \mathcal{GP}(\sin(2 + t/10), k(t, t)) + \epsilon(t) + t - 21/2, \\
C_3 : Y(t) &= \mathcal{GP}(\cos(3 * t/10), k(t, t)) + 3 * \mathcal{GP}(\sin(3 + t/10), k(t, t)) + \epsilon(t) + t - 21/2, \\
C_4 : Y(t) &= \mathcal{GP}(\sin(1 * t/10), k(t, t)) + \mathcal{GP}(\cos(1 + t/10), k(5 * t, 5 * t)) \\
&+ \mathcal{GP}\left(\sqrt{t/10} + t/10 + 1, k(8 * t, 8 * t)\right) + \epsilon(t) + t - 21/2, \\
C_5 : Y(t) &= \mathcal{GP}(\sin(2 * t/10), k(t, t)) + 2 * \mathcal{GP}(\cos(2 + t/10), k(5 * t, 5 * t)) \\
&+ 2 * \mathcal{GP}\left(\sqrt{t/10} + t/10 + 1, k(8 * t, 8 * t)\right) + \epsilon(t) + t - 21/2, \\
C_6 : Y(t) &= \mathcal{GP}(\sin(3 * t/10), k(t, t)) + 3 * \mathcal{GP}(\cos(3 + t/10), k(5 * t, 5 * t)) \\
&+ 3 * \mathcal{GP}\left(\sqrt{t/10} + t/10 + 1, k(8 * t, 8 * t)\right) + \epsilon(t) + t - 21/2,
\end{aligned}$$

Scenario 5: As discussed in the dataset provided by Golovkine, Klutchnikoff and Patilea [39]. The number of cluster is $K = 4$. An independent sample of $N = 100$ univariate curves is simulated for 30 observations. We define the mixture coefficients $\pi_k = [0.15, 0.15, 0.3, 0.2, 0.2]$. A description of each mean curve and GP simulation is described below:

$$\begin{aligned}
\mu_1(t) &= \sqrt{2} * \sin\left(1 - \frac{\pi}{2}t\right), \quad C_1 : Y(t) = \mu_4(t) + \mathcal{GP}(\mu_1(t), k(t, 16 * t)) + \\
&\quad \mathcal{GP}(\mu_2(t), k(t, 64/9 * t)) + \mathcal{GP}(\mu_3(t), k(t, 16/9 * t)), \\
\mu_1(t) &= \sqrt{2} * \sin\left(2 - \frac{\pi}{2}t\right), \quad C_2 : Y(t) = \mu_4(t) + \mathcal{GP}(\mu_1(t), k(t, t)) + \\
&\quad \mathcal{GP}(\mu_2(t), k(t, 4/9 * t)) + \mathcal{GP}(\mu_3(t), k(t, 1/9 * t)), \\
\mu_1(t) &= \sqrt{2} * \sin\left(3 - \frac{\pi}{2}t\right), \quad C_3 : Y(t) = \mu_5(t) + \mathcal{GP}(\mu_1(t), k(t, 16 * t)) + \\
&\quad \mathcal{GP}(\mu_2(t), k(t, 64/9 * t)) + \mathcal{GP}(\mu_3(t), k(t, 16/9 * t)), \\
\mu_1(t) &= 20/(1 + \exp(-t)), \quad C_4 : Y(t) = \mu_5(t) + \mathcal{GP}(\mu_1(t), k(t, t)) + \\
&\quad \mathcal{GP}(\mu_2(t), k(t, 4/9 * t)) + \mathcal{GP}(\mu_3(t), k(t, 1/9 * t)), \\
\mu_1(t) &= 25/(1 + \exp(-t)). \quad C_5 : Y(t) = \mu_4(t) + \mathcal{GP}(\mu_1(t), k(t, t)) + \\
&\quad \mathcal{GP}(\mu_2(t), k(t, 4/9 * t)) + \mathcal{GP}(\mu_3(t), k(t, 1/9 * t)).
\end{aligned}$$

Scenario 6: As discussed in the dataset provided by Delaigle and Hall [37]. The number of cluster is $K = 4$. An independent sample of $N = 100$ univariate curves is simulated for 30 observations. We define the mixture coefficients $\pi_k = [0.2, 0.2, 0.5, 0.1]$. A description of

each mean curve and GP simulation is described below:

$$\begin{aligned}
\mu_1(t) &= -2 \sin(t) \log(t + 2), & C_1 : Y(t) &= \mathcal{GP}(\mu_1(t), k(t, t)), \\
\mu_2(t) &= \cos(t) \log(t + 0.5), & C_2 : Y(t) &= \mathcal{GP}(\mu_2(t), k(t, t)), \\
\mu_3(t) &= -2 - \cos(t) \sqrt{5t + 0.5}, & C_3 : Y(t) &= \mathcal{GP}(\mu_3(t), k(t, t)), \\
\mu_4(t) &= \cos(2t) \log(t + 0.5) \sqrt{t + 0.5}. & C_4 : Y(t) &= \mathcal{GP}(\mu_4(t), k(t, t)).
\end{aligned}$$

Scenario 7: As discussed in the dataset provided by Meng, Liang, Cao and He [40]. The number of cluster is $K = 3$. An independent sample of $N = 100$ univariate curves is simulated for 30 observations. We define the mixture coefficients $\pi_k = [0.2, 0.2, 0.6]$. We add some Gaussian noise $\epsilon(t)$, where the probability density for the noise is $\epsilon(t) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}$. In this case, $\mu = 0$ and $\sigma = 1$. A description of each mean curve and GP simulation is described below:

$$\begin{aligned}
\mu_1(t) &= 3, & C_1 : Y(t) &= \mathcal{GP}(\mu_1(t), k(t, t)) + \epsilon(t), \\
\mu_2(t) &= \cos(1.5 * \pi * t), & C_2 : Y(t) &= \mathcal{GP}(\mu_2(t), k(t, t)) + \epsilon(t), \\
\mu_3(t) &= -\sin(1.5 * \pi * t), & C_3 : Y(t) &= \mathcal{GP}(\mu_3(t), k(t, t)) + \epsilon(t).
\end{aligned}$$

The brief information of each simulation Scenario is given in table 4.1 below.

Data	Observations	Frequency Measurements	Classes	Added Noise Ratio
Scenario 1	100	30	4	0.1
Scenario 2	100	30	2	0.1
Scenario 3	500	30	4	0.1
Scenario 4	100	30	6	0.1
Scenario 5	100	30	5	0.1
Scenario 6	100	30	4	0.1
Scenario 7	100	30	3	0.1

Table 4.1: Simulation Scenario Description

4.1.2 Simulation Scenarios Evaluation

In this section, we present a detailed analysis of the clustering algorithm performance based on the simulation data. We compare the clustering results obtained by our proposed algorithm (FunGP) with three state-of-the-art clustering algorithms: FunHDDC, FunFEM, and FunClust. The evaluation metrics used for comparison are the Adjusted Rand Index (ARI) and the Adjusted Mutual Information (AMI). Table 4.2 and 4.3 show the clustering results for each of the simulated scenarios.

According to the setup of the state-of-the-art algorithms, it is important to note that their parameter options are more limited compared to FunGP. The three methods used for

comparison do not offer extensive tuning parameters (such as different projector types and parameters) or smoothing methods. Therefore, the same smoothed data is used by all algorithms for each specific dataset. The parameter selections for FunHDDC, FunFEM, and FunClust are listed as follows:

```

1 FunHDDC: threshold = 0.1, model = 'ABQkDk', itermax = 2000, eps =
    1e-6, init = 'kmeans', criterion="bic", algo='EM', d_select="
    Cattell"
2 FunFEM: model = "AkjBk", crit = "bic", init = "kmeans", Tinit = c
    (), maxit = 50, eps = 1e-5, disp = FALSE, lambda = 0, graph =
    FALSE
3 FunClust: thd = 0.005, epsilon = 1e-5, increaseDimension = FALSE,
    hard = FALSE, fixedDimension = integer(0), nbInit = 5,
    nbIterInit = 20, nbIteration = 200

```

Table 4.2: Simulation Scenario ARI Scores

Scenario	FunGP	FunHDDC	FunFEM	FunClust
1	1.000	1.000	1.000	1.000
2	1.000	1.000	1.000	0.026
3	0.979	1.000	0.700	0.137
4	0.294	0.480	0.571	0.150
5	0.719	0.385	0.629	0.405
6	0.929	0.530	1.000	0.033
7	0.653	0.051	0.036	0.004

Table 4.3: Simulation Scenario AMI Scores

Scenario	FunGP	FunHDDC	FunFEM	FunClust
1	1.000	1.000	1.000	1.000
2	1.000	1.000	1.000	0.076
3	0.968	1.000	0.833	0.158
4	0.427	0.629	0.719	0.309
5	0.823	0.523	0.746	0.604
6	0.903	0.748	1.000	0.015
7	0.585	0.037	0.000	0.007

In Scenario 1, as presented in figure 4.3, all algorithms achieve high ARI and AMI scores, indicating perfect or near-perfect agreement with the ground truth clustering. This is related to the more pronounced polynomial characteristic distribution of the simulated data, which hardly affects the results even with the addition of scaled noise. Different colors in the figure 4.3 below represent different clusters. It can be seen from the figure that except FunClust, which was affected after adding noise, all others completed perfect clustering. For FunClust, noise causes data in one cluster to be completely identified as another cluster.

Table 4.4: Noised Simulation Scenario ARI Scores

Scenario	FunGP	FunHDDC	FunFEM	FunClust
1	1.000	1.000	1.000	0.510
2	1.000	1.000	1.000	0.014
3	0.959	0.989	0.259	0.001
4	0.224	-0.006	-0.005	-0.008
5	0.403	0.323	0.350	0.405
6	0.862	0.847	1.000	0.005
7	0.453	-0.002	-0.005	0.004

Table 4.5: Noised Simulation Scenario AMI Scores

Scenario	FunGP	FunHDDC	FunFEM	FunClust
1	1.000	1.000	1.000	0.679
2	1.000	1.000	1.000	0.020
3	0.943	0.985	0.407	0.002
4	0.034	-0.028	-0.002	-0.009
5	0.321	0.437	0.422	0.604
6	0.864	0.860	1.000	0.001
7	0.480	-0.003	-0.011	0.040

The choices for smoother and projector are as follows:

```

1 smoother(smoother = 'bspline', smoother_args = {'degree': 3, '
  n_basis': 50})
2 projector(projection_method = 'pca', projection_args = {'
  n_components': 12})

```

Listing 4.1: Smoother and Projector Arguments for Simulation Scenario 1

In Scenario 2, as presented in figure 4.4, the overall situation is similar to Scenario 1, but the impact of noise is greater in FunClust. FunClust achieves poor results on this kind of data, approaching the results of random clustering.

The choices for smoother and projector are as follows:

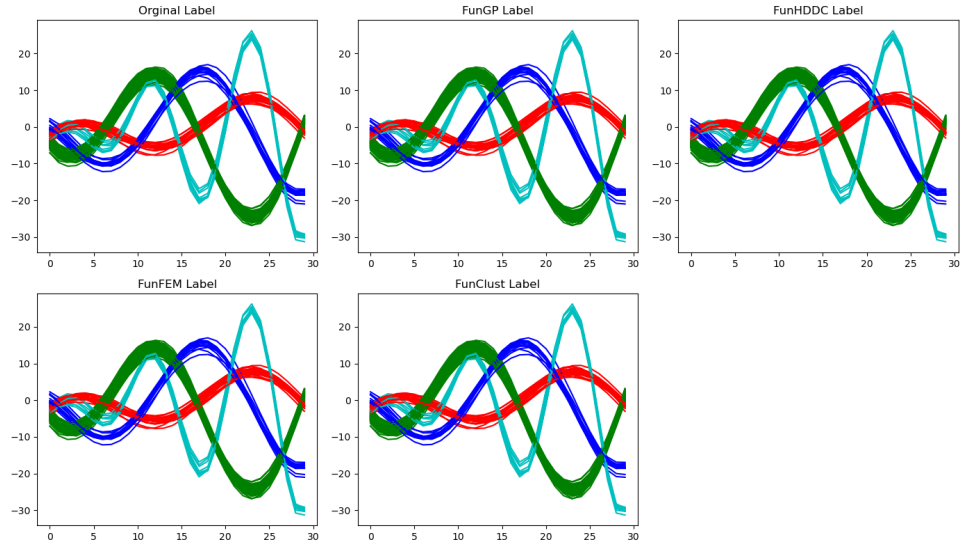
```

1 smoother(smoother = 'bspline', smoother_args = {'degree': 3, '
  n_basis': 50})
2 projector(projection_method = 'pca', projection_args = {'
  n_components': 12})

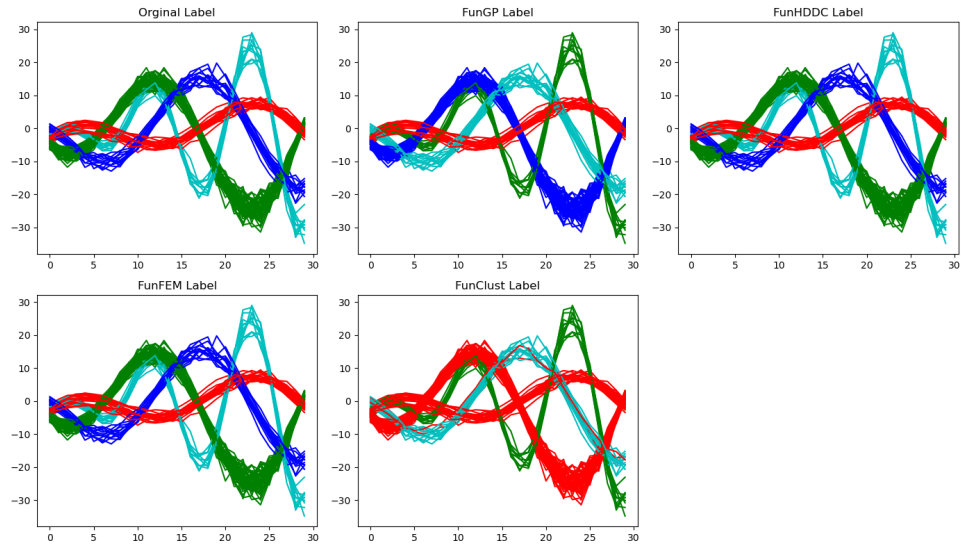
```

Listing 4.2: Smoother and Projector Arguments for Simulation Scenario 2

In Scenario 3, as presented in figure 4.5, as the characteristics between clusters in the simulated data become more similar, the results produced by the four algorithms begin to become less perfect. It can be seen that FunGP and FunHDDC have significantly better

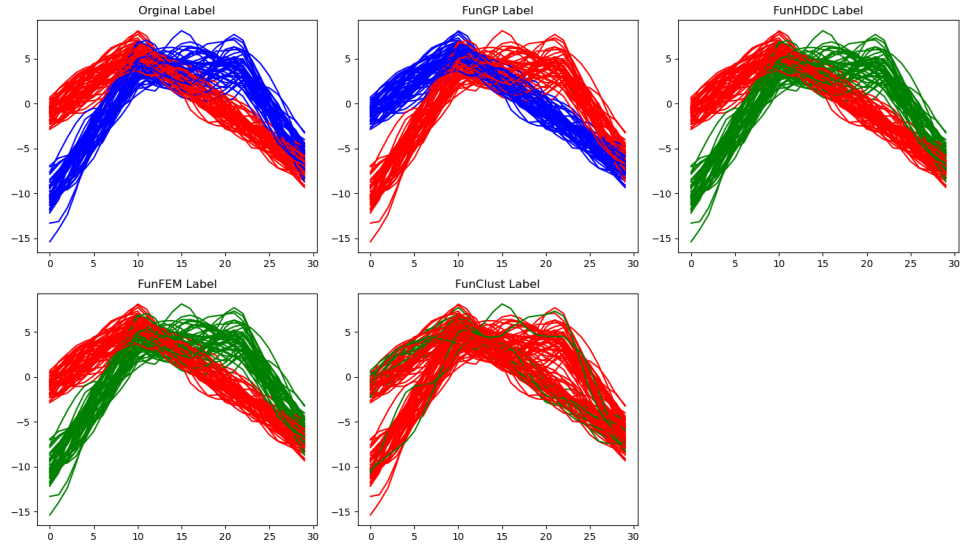


(a) Simulation Label 1

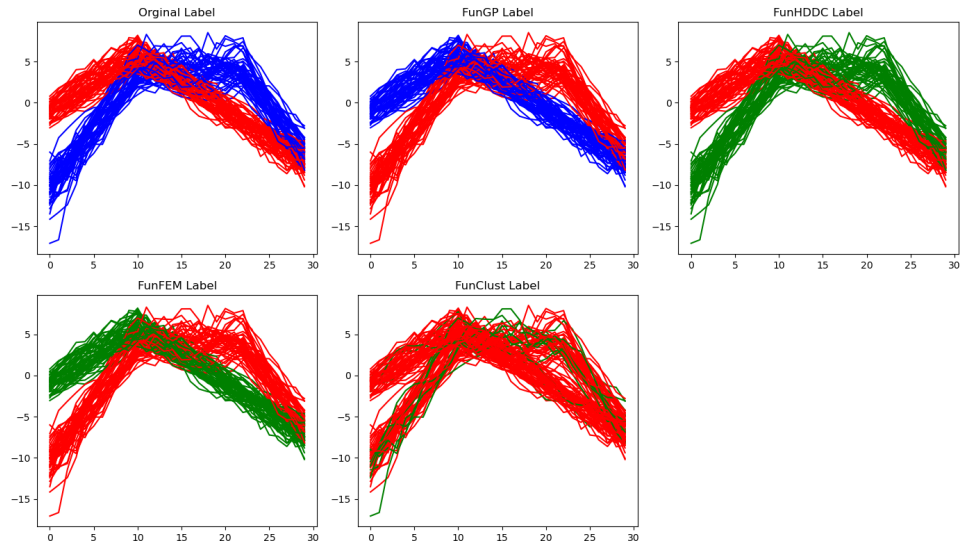


(b) Noised Simulation Label 1

Figure 4.3: Clustering Results for Simulation Scenario 1



(a) Simulation Label 2



(b) Noised Simulation Label 2

Figure 4.4: Clustering Results for Simulation Scenario 2

results and robustness to FunFEM and FunClust. The first two algorithms were not significantly negatively affected after adding noise since they still achieved near-perfect results, but FunFEM and FunClust, which originally had poorer clustering results for the data, were greatly affected. ARI and AMI decreased from (0.7, 0.833) and (0.136, 0.158) to (0.259, 0.497) and (0.001, 0.002). It can be seen from the figure 4.5 that adding noise to this type of data causes the algorithm to have a great negative impact on the ability to identify different types of data.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'bspline', smoother_args = {'degree': 3, '
  n_basis': 50})
2 projector(projection_method = 'pca', projection_args = {'
  n_components': 12})
```

Listing 4.3: Smoother and Projector Arguments for Simulation Scenario 3

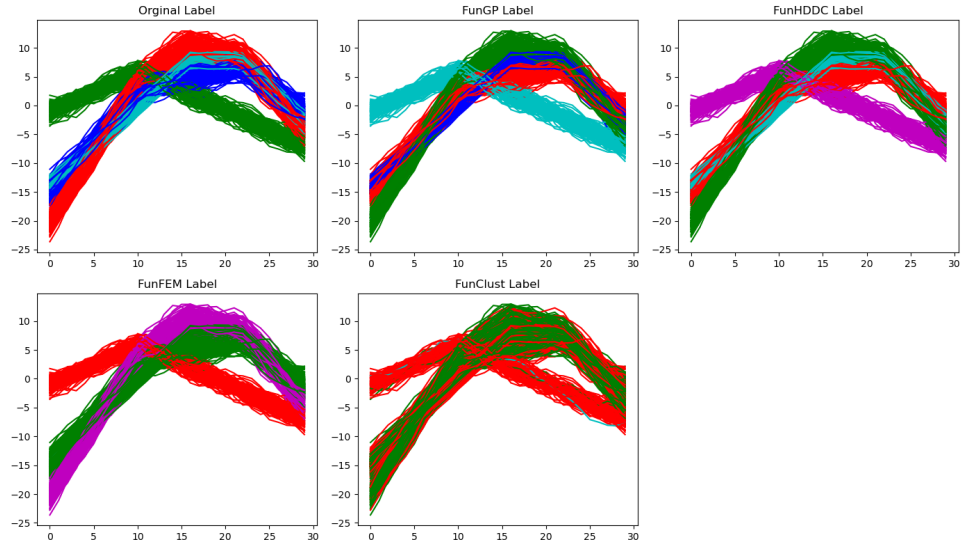
In Scenario 4, as presented in figure 4.6, this set of simulated data is generated by adding different degrees of Gaussian noise to the same mean function. Note that the Gaussian noise here is different from the scaled noise added later. Scaled noise is random noise added up and down to the entire data based on itself, and its value does not exceed 10% of the data itself. The purpose of this set of simulated data is to test the algorithm's ability to identify several Gaussian noises added to the same mean function to varying degrees. According to AMI and ARI, it can be seen that FunHDDC and FunFEM have better results in this simulation scenario, while our algorithm is relatively weak. And all algorithms lose their correctness after adding scaled noise. Note that when AMI and ARI are less than or equal to 0, it indicates that the clustering result is close to pure random clustering. This is because the data itself is generated by adding different degrees and styles of Gaussian noise. Adding noise again will destroy the characteristics of the data itself. Since the jitter of the data itself is relatively severe, it is difficult to see the difference in results from the figure 4.6.

The choices for smoother and projector are as follows:

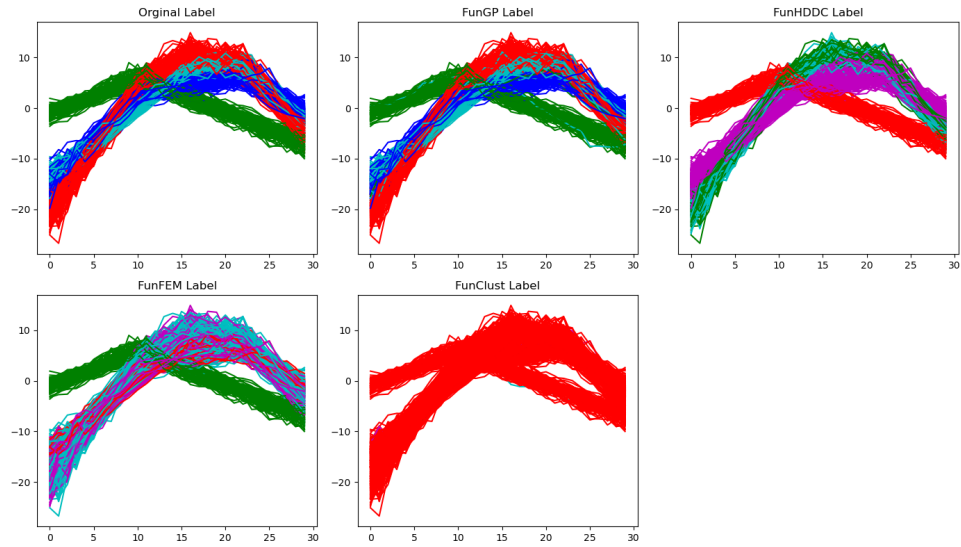
```
1 smoother(smoother = 'bspline', smoother_args = {'degree': 3, '
  n_basis': 50})
2 projector(projection_method = 'fpca', projection_args = {'
  n_components': 15, 'basis': 'fourier', 'n_basis': 15})
```

Listing 4.4: Smoother and Projector Arguments for Simulation Scenario 4

In Scenario 5, as presented in figure 4.7, the mean function of this set of data is based on exponential operations and trigonometric functions, resulting in certain periodic disturbances in the data. For such data, because our FunGP can freely choose different smoothers and projectors, we can combine them more flexibly to get better results. Different smoothers and

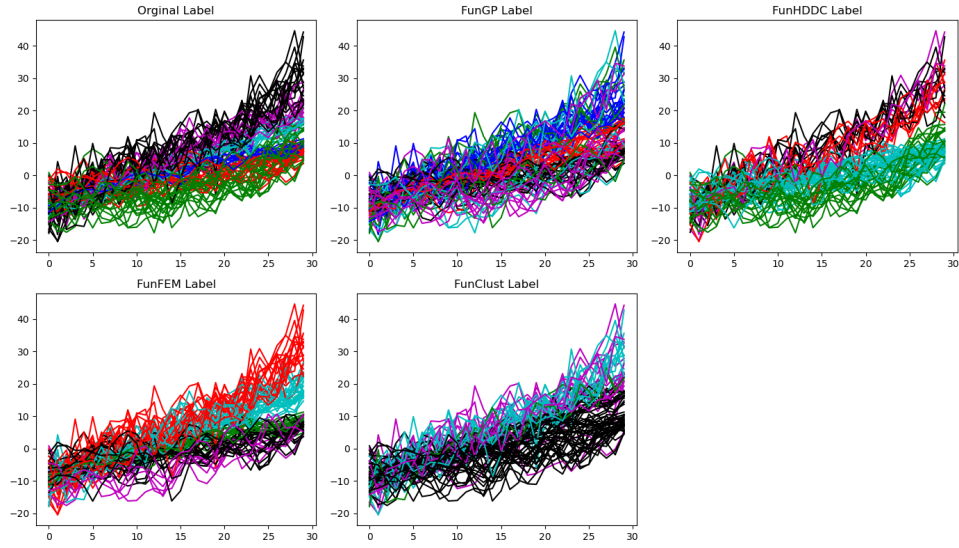


(a) Simulation Label 3

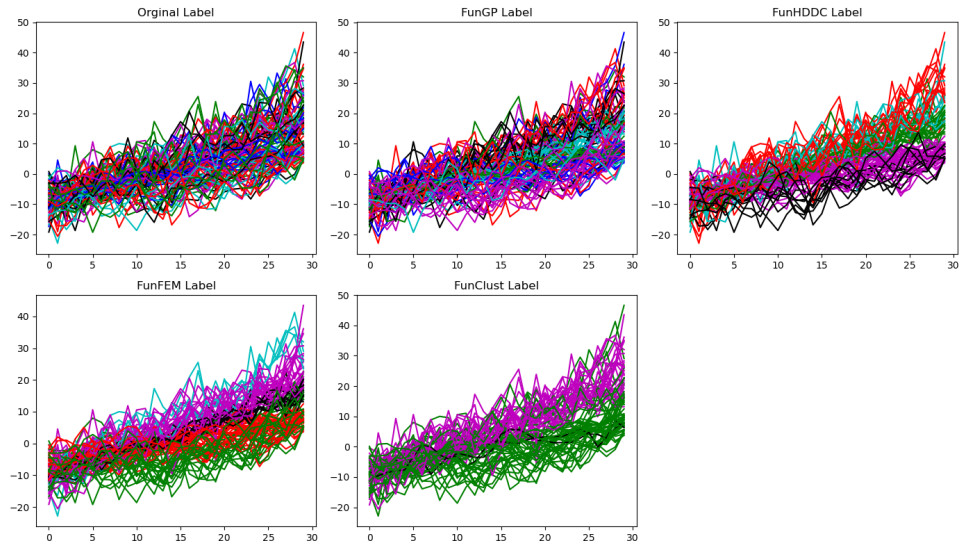


(b) Noised Simulation Label 3

Figure 4.5: Clustering Results for Simulation Scenario 3



(a) Simulation Label 4



(b) Noised Simulation Label 4

Figure 4.6: Clustering Results for Simulation Scenario 4

projectors during the test will have a huge impact on the results, especially in this simulation scenario. It can be seen from the results that FunGP has significant advantages. But when noise is added, the smoother and projector modes may cause the noise to have a greater impact on the algorithm. The addition of noise to this data greatly changes the structure of the data, resulting in much fewer gaps between classes. From the figure 4.7, we can clearly see the changes before and after adding noise, which also has a great impact on the results of the algorithm.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'fourier', smoother_args = {'n_basis': 10, '
    period': 3})
2 projector(projection_method = 'kpca', projection_args = {'
    n_components': 12})
```

Listing 4.5: Smoother and Projector Arguments for Simulation Scenario 5

In Scenario 6, as presented in figure 4.8, this set of data has a relatively similar structure to Scenario 1. Both mean functions are composed of trigonometric functions, but the difference between different clusters in Scenario 6 will be smaller. In the face of data that cannot achieve perfect clustering results but has a relatively simple structure, FunGP maintains its advantage over other algorithms, except that FunFEM still achieves perfect classification results.

The choices for smoother and projector are as follows:

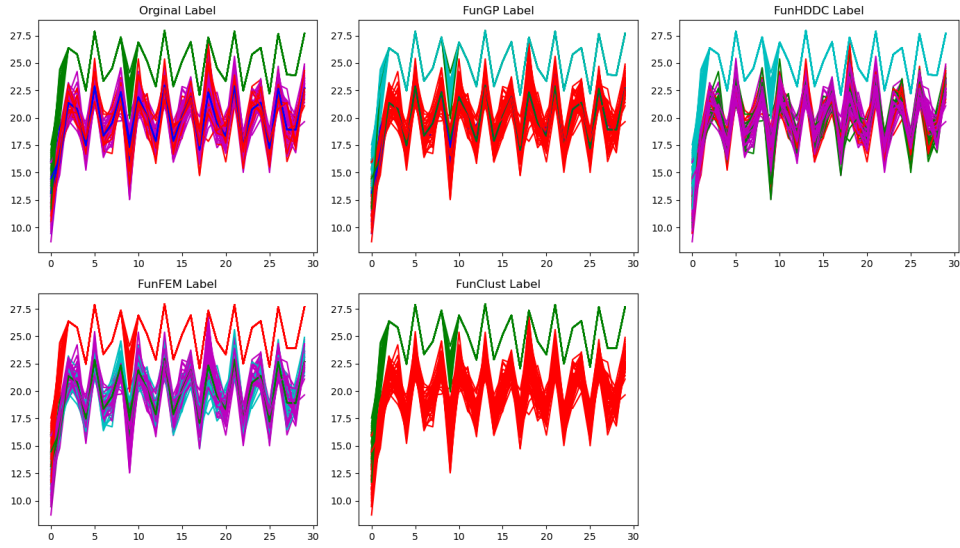
```
1 smoother(smoother = 'fourier', smoother_args = {'n_basis': 10, '
    period': 3})
2 projector(projection_method = 'pca', projection_args = {'
    n_components': 12})
```

Listing 4.6: Smoother and Projector Arguments for Simulation Scenario 6

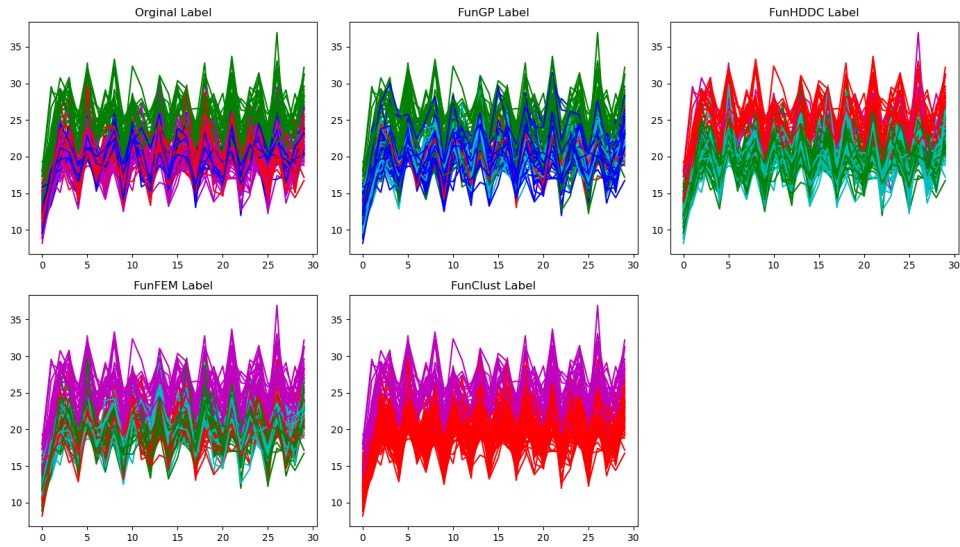
In Scenario 7, as presented in figure 4.9, the differences in structure between this set of data clusters were not recognized by the most of clustering algorithms we used. But our algorithm achieves good results to a certain extent because we can freely use different smoothers and projectors. On this data, we used the orthogonal wavelet as the projector and achieved results that the other three algorithms could not achieve. Given that other algorithms almost give results equivalent to random clustering.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'wavelet', smoother_args = {'wavelet': 'db4',
    'level': 4})
```

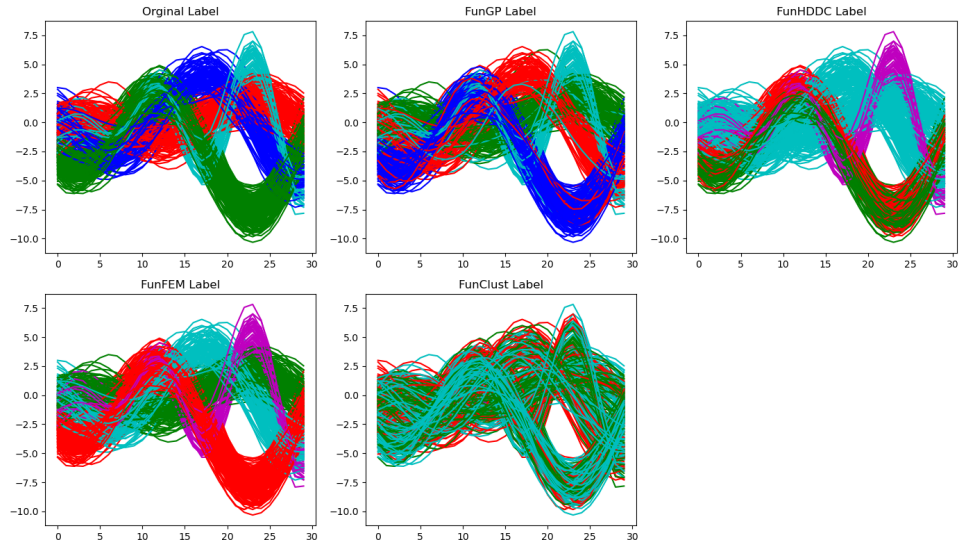


(a) Simulation Label 5

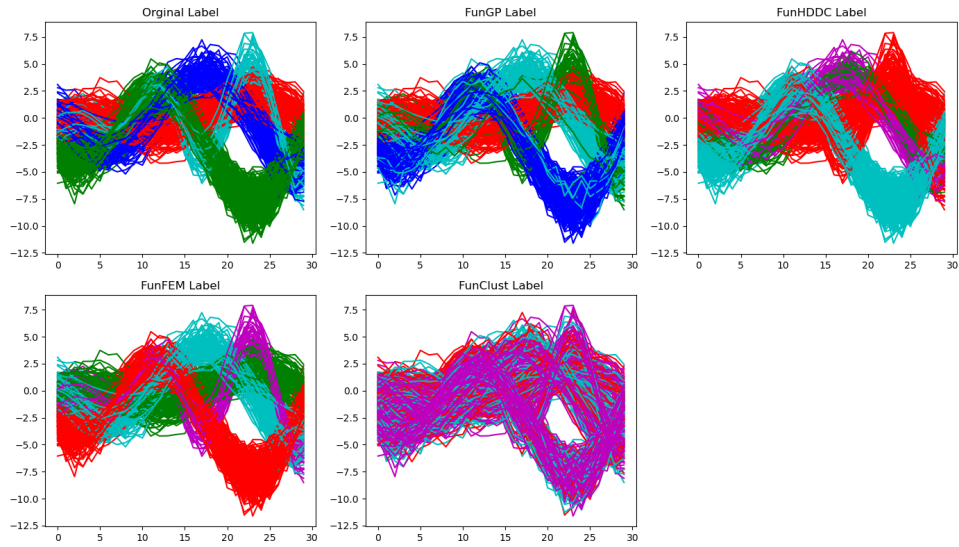


(b) Noised Simulation Label 5

Figure 4.7: Clustering Results for Simulation Scenario 5



(a) Simulation Label 6

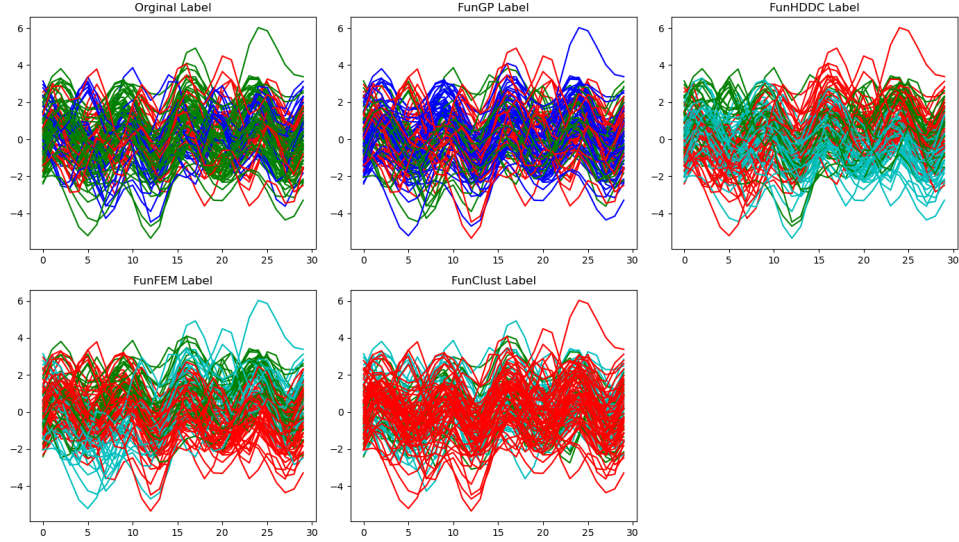


(b) Noised Simulation Label 6

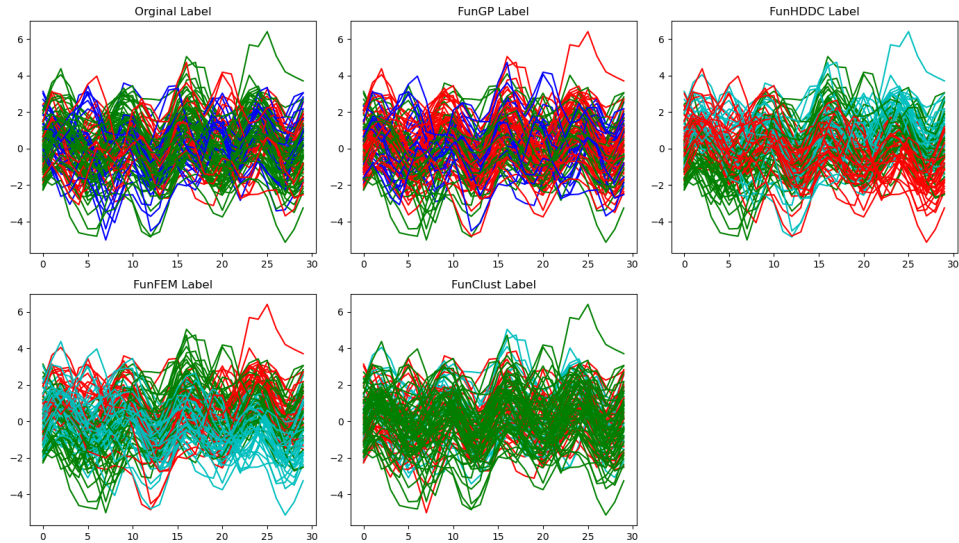
Figure 4.8: Clustering Results for Simulation Scenario 6

```
2 projector(projection_method = 'expansion', projection_args = {'
    basis': 'wavelet', 'n_basis': 15, 'wavelet': 'db4'})
```

Listing 4.7: Smoother and Projector Arguments for Simulation Scenario 7



(a) Simulation Label 7



(b) Noised Simulation Label 7

Figure 4.9: Clustering Results for Simulation Scenario 7

According to the simulation scenarios in 7, it can be seen that our algorithm has the best overall performance. In scenarios 1, 2, and 5, our algorithm showed the best results. In scenarios 3 and 6, our algorithm was very close to the perfect clustering result: ARI and AMI reached (0.979, 0.968) and (0.929, 0.862) respectively. After adding noise, it can be seen that noise has a certain degree of negative impact on the clustering results of the four algorithms. However, our algorithm has no significant advantage in its ability to resist noise, especially in Scenario 5, the impact of noise is significant. But overall FunGP still maintains

relatively strong competitiveness compared to FunHDDC, FunFEM and FunClust.

4.1.3 Simulation Scenarios Conclusion

After conducting experiments on seven sets of simulated data, along with their noise-processed versions, our algorithm demonstrated certain advantages over comparable algorithms. However, in some scenarios, it did not exhibit significantly stronger clustering capabilities. While our method showed promising performance in specific cases, its overall behavior suggests that it presents a more complex and challenging approach.

In some relatively easy tasks, such as scenarios 1 and 2, our algorithm is not as fast as other algorithms. This is because our algorithm has more steps. Although this brings greater flexibility and stability, it also leads to low operating efficiency. In this case, most algorithms are capable of achieving satisfactory results, and FunGP does not demonstrate sufficient competitiveness.

However, as the complexity of the data increases, FunGP begins to demonstrate stronger performance. For instance, in Scenarios 6, and 7, FunGP delivers superior results in handling complex data, particularly in Scenario 7, where other algorithms only manage to produce random clustering outcomes. In contrast, our algorithm achieves relatively robust results. This advantage stems from the flexibility of FunGP to select different smoothers and projectors. In these scenarios, we employed an orthogonal wavelet as the projector, which significantly outperformed other methods. This performance improvement comes at the cost of increased parameter tuning. FunGP requires more time to determine optimal parameters compared to other algorithms, which may be a trade-off for its enhanced performance in more complex datasets. The need for careful parameter adjustment is a key aspect of FunGP's ability to handle intricate data structures effectively.

In Scenario 4, our algorithm performs worse than FunHDDC and FunFEM. The data is derived from adding different degrees of Gaussian noise to the same mean function. FunGP excels in extracting and interpreting the characteristics of complex data. However, it exhibits a notable limitation when dealing with datasets that share the same mean function. In such cases, the algorithm struggles to differentiate between clusters effectively, resulting in diminished performance. This weakness highlights a potential area for improvement in the algorithm's ability to handle data with homogeneous mean structures.

Overall, FunGP demonstrates solid performance across most simulation scenarios. Its flexibility in selecting smoothers and projectors allows it to adapt effectively to varying data complexities, delivering competitive results in a wide range of situations. Despite certain limitations, such as handling data with similar mean functions, FunGP generally proves to be a robust and versatile tool in many simulated environments.

4.2 Real Data Scenarios

In this section, we will use real data sets to test and evaluate our algorithms and state-of-the-art clustering methods. Similar to the previous simulation scenarios section, we will use ARI and AMI to analyse the results.

4.2.1 Real Data Description

We used 12 real data from various fields, figure 4.10 shows the original data set, and different colors in the figure represent different clusters. All data are functional data that vary over time.

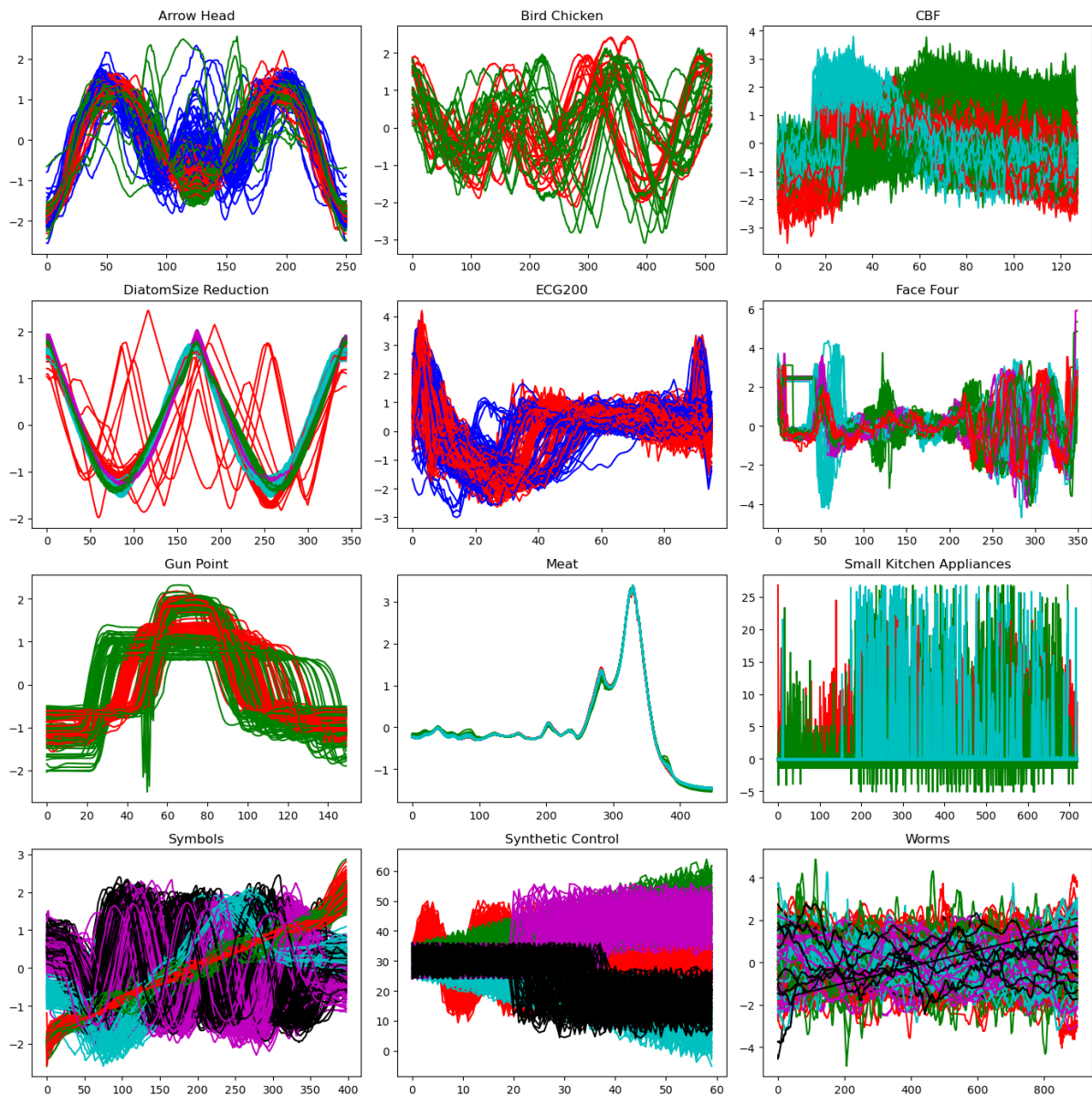


Figure 4.10: Real Data Scenarios

Arrowhead As discussed in the dataset provided by Ye and Keogh [41]. The arrowhead data consists of outlines of the images of arrowheads. The shapes of the projectile points are converted into a time series using the angle-based method. In this dataset, $N = 211$ univariate observation pairs and 251 frequency measurements are made over time. x_i gives the class membership. The 3 classes are called "Avonlea", "Clovis" and "Mix":

$$x_i \in \{1, 2, 3\} \text{ with } \begin{cases} 1 & \longleftrightarrow & \text{"Avonlea"} \\ 2 & \longleftrightarrow & \text{"Clovis"} \\ 3 & \longleftrightarrow & \text{"Mix"} \end{cases}$$

Bird Chicken As discussed in the dataset provided by J. Hills, A. Bagnall. This dataset can be found at <https://www.timeseriesclassification.com/description.php?Dataset=BirdChicken>.

Bird/Chicken is the problem of distinguishing between an outline of a bird and a chicken. In this dataset, $N = 40$ univariate observation pairs and 512 frequency measurements. x_i gives the class membership. The 2 classes are called "Bird", "Chicken":

$$x_i \in \{1, 2\} \text{ with } \begin{cases} 1 & \longleftrightarrow & \text{"Bird"} \\ 2 & \longleftrightarrow & \text{"Chicken"} \end{cases}$$

CBF As discussed in the dataset provided by Saito [42]. Cylinder-Bell-Funnel is a simulated data set. Data from each class is standard normal noise plus an offset term which differs for each class. In this dataset, $N = 930$ univariate observation pairs and 128 frequency measurements. x_i gives the class membership. The 3 classes are called "Cylinder", "Bell", "Funnel":

$$x_i \in \{1, 2, 3\} \text{ with } \begin{cases} 1 & \longleftrightarrow & \text{"Cylinder"} \\ 2 & \longleftrightarrow & \text{"Bell"} \\ 3 & \longleftrightarrow & \text{"Funnel"} \end{cases}$$

Diatom Size Reduction As discussed in the dataset can be found in <http://www.timeseriesclassification.com/description.php?Dataset=DiatomSizeReduction>. It shows outlines of four types of diatoms In this dataset, $N = 322$ univariate observation pairs and 345 frequency measurements. x_i gives the class membership. The 4 classes are called "Gomphonema augur", "Fragilariforma bicapitata", "Stauroneis smithii" and "Eunotia tenella":

$$x_i \in \{1, 2, 3, 4\} \text{ with } \begin{cases} 1 \longleftrightarrow \text{"Gomphonema augur"} \\ 2 \longleftrightarrow \text{"Fragilariforma bicapitata"} \\ 3 \longleftrightarrow \text{"Stauroneis smithii"} \\ 4 \longleftrightarrow \text{"Eunotia tenella"} \end{cases}$$

ECG200 As discussed in the dataset provided by Olszewski [43]. Each series traces the electrical activity recorded during one heartbeat. In this dataset, $N = 200$ univariate observation pairs and 96 frequency measurements. x_i gives the class membership. The 2 classes are a normal heartbeat and a Myocardial Infarction:

$$x_i \in \{1, 2\} \text{ with } \begin{cases} 1 \longleftrightarrow \text{"Normal Heartbeat"} \\ 2 \longleftrightarrow \text{"Myocardial Infarction"} \end{cases}$$

Face Four As discussed in the dataset provided by A. Ratanamahatana, E. Keogh <http://www.timeseriesclassification.com/description.php?Dataset=FaceFour.A> Four-class face outline problem. In this dataset, $N = 112$ univariate observation pairs and 350 frequency measurements. x_i gives the class membership. The 4 classes are :

$$x_i \in \{1, 2, 3, 4\} \text{ with } \begin{cases} 1 \longleftrightarrow \text{"Face Outline 1"} \\ 2 \longleftrightarrow \text{"Face Outline 2"} \\ 3 \longleftrightarrow \text{"Face Outline 3"} \\ 4 \longleftrightarrow \text{"Face Outline 4"} \end{cases}$$

Gun Point As discussed in the dataset provided by Ratanamahatana [44]. This data set involves one female actor and one male actor making a motion with their hand. In this dataset, $N = 200$ univariate observation pairs and 150 frequency measurements. x_i gives the class membership. The 2 classes are "Gun" and "No Gun":

$$x_i \in \{1, 2\} \text{ with } \begin{cases} 1 \longleftrightarrow \text{"Gun"} \\ 2 \longleftrightarrow \text{"No Gun"} \end{cases}$$

Meat As discussed in the dataset provided by K. Kemlsey, A. Bagnall [45]. Food spectrographs are used in chemometrics to classify food types, a task that has obvious applications in food safety and quality assurance. In this dataset, $N = 120$ univariate observation pairs and 448 frequency measurements. x_i gives the class membership. The 3 classes are chicken, pork, and turkey:

$$x_i \in \{1, 2, 3\} \text{ with } \begin{cases} 1 \longleftrightarrow \text{"chicken"} \\ 2 \longleftrightarrow \text{"pork"} \\ 3 \longleftrightarrow \text{"turkey"} \end{cases}$$

Small Kitchen Appliances As discussed in the dataset provided by Lines and Bagnall <https://www.uea.ac.uk/about/school-of-computing-sciences/research>. These problems were taken from data recorded as part of a government-sponsored study called Powering the Nation. The intention was to collect behavioral data about how consumers use electricity within the home to help reduce the UK's carbon footprint. In this dataset, $N = 750$ univariate observation pairs and 720 frequency measurements. x_i gives the class membership. The 3 classes are Kettle, Microwave and Toaster:

$$x_i \in \{1, 2, 3\} \text{ with } \begin{cases} 1 \longleftrightarrow \text{"Kettle"} \\ 2 \longleftrightarrow \text{"Microwave"} \\ 3 \longleftrightarrow \text{"Toaster"} \end{cases}$$

Synthetic Control This dataset contains 600 examples of control charts synthetically generated by the process in Alcock and Manolopoulos (1999) <https://archive.ics.uci.edu/dataset/139/synthetic+control+chart+time+series>. In this dataset, $N = 600$ univariate observation pairs and 60 frequency measurements. x_i gives the class membership. The 6 classes are:

$$x_i \in \{1, 2, 3, 4, 5, 6\} \text{ with } \begin{cases} 1 \longleftrightarrow \text{"Normal"} \\ 2 \longleftrightarrow \text{"Cyclic"} \\ 3 \longleftrightarrow \text{"Increasing trend"} \\ 4 \longleftrightarrow \text{"Decreasing trend"} \\ 5 \longleftrightarrow \text{"Upward shift"} \\ 6 \longleftrightarrow \text{"Downward shift"} \end{cases}$$

Symbols As discussed in the dataset provided by J. Brady, E. Keogh <http://www.timeseriesclassification.com/description.php?Dataset=Symbols>. This data indicates an experiment in which thirteen people participated. They were asked to copy the randomly appearing symbol as best they could. There were 3 possible symbols, each person contributed about 30 attempts. In this dataset, $N = 1020$ univariate observation pairs and 398 frequency measurements. x_i gives the class membership. The 3 classes are:

$$x_i \in \{1, 2, 3\} \text{ with } \begin{cases} 1 \longleftrightarrow \text{"Symbol 1"} \\ 2 \longleftrightarrow \text{"Symbol 2"} \\ 3 \longleftrightarrow \text{"Symbol 3"} \end{cases}$$

Worms As discussed in the dataset provided by Andre Brown, A. Bagnall [46, 47]. The data relates to 258 traces of worms converted into four "eigenworm" series. In this dataset, $N = 258$ univariate observation pairs and 900 frequency measurements. x_i gives the class membership. The 4 classes are:

$$x_i \in \{1, 2, 3, 4\} \text{ with } \begin{cases} 1 \longleftrightarrow \text{"eigenworm 1"} \\ 2 \longleftrightarrow \text{"eigenworm 2"} \\ 3 \longleftrightarrow \text{"eigenworm 3"} \\ 4 \longleftrightarrow \text{"eigenworm 4"} \end{cases}$$

The information of each dataset is given in table 4.6 below.

Data	Observations	Frequency Measurements	Classes
Arrow Head	211	251	3
Bird Chicken	40	512	2
CBF	930	128	3
Diatom Size Reduction	322	345	4
ECG200	200	96	2
FaceFour	112	350	4
GunPoint	200	150	2
Meat	120	448	3
Small Kitchen Appliances	750	720	3
Synthetic control	600	60	6
Symbols	1020	398	3
Worms	258	800	4

Table 4.6: Real Data Description

4.2.2 Real Data Evaluation

In this section, We will evaluate the results on real data. This includes the selection of smoother and projector used in each set of real data and a comparison of FunGP, FunHDDC, FunFem, and FunCluster.

Arrowhead The Arrowhead dataset exhibits large fluctuations but lacks periodicity. Notably, there are certain regions where the data between the three classes are close to one another, which could potentially lead to disturbances in classification. When actually

selecting smoothers and projectors, we use a pre-designed loop enumeration method, try all smoothers, select the three best solutions and match them with all projectors, and then perform fine-tuning of hyperparameters to obtain better results. After conducting thorough testing, we selected the following smoother and projector for optimal performance. It is important to note that our method may require additional time for parameter selection due to its flexibility in choosing and tuning hyperparameters. This added flexibility allows for better adjustment to the specific characteristics of the data, then leading to improved results.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'bspline', smoother_args = {'degree': 3, '
  n_basis': 50})
2 projector(projection_method = 'pca', projection_args = {'
  n_components': 12})
```

Listing 4.8: Smoother and Projector Arguments for Dataset Arrowhead

The result of each clustering method is in figure 4.11.

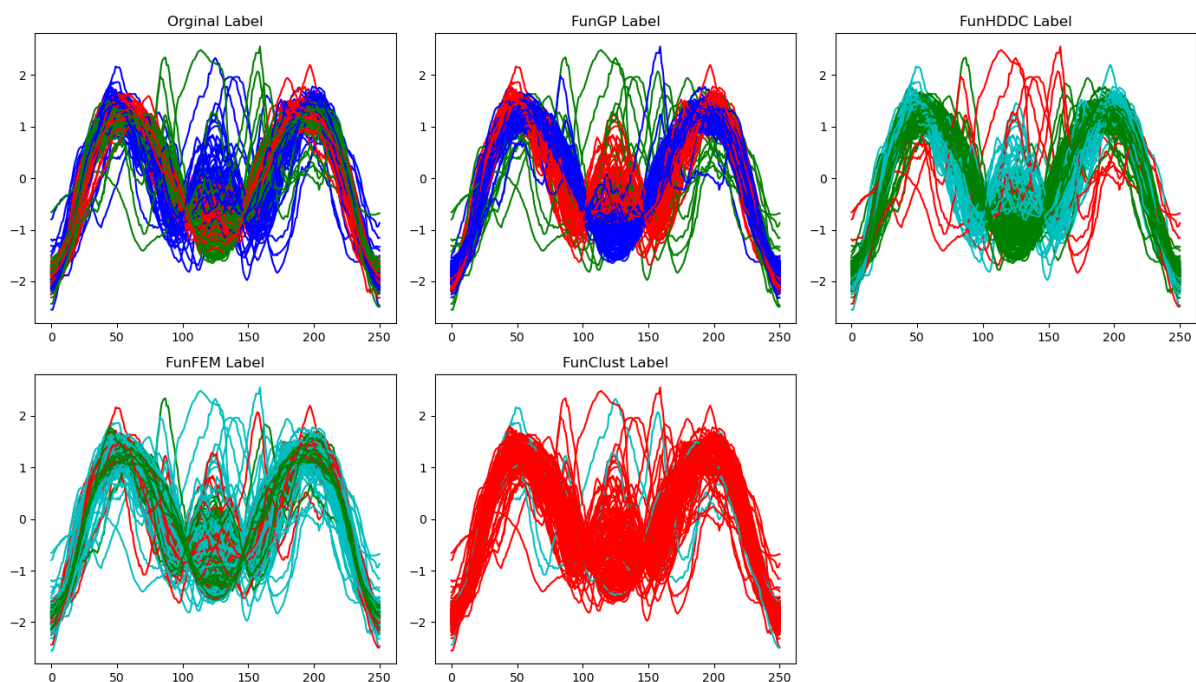


Figure 4.11: Clustering Results for Dataset Arrowhead

The first step of our algorithm is to fit the data by choosing a suitable smoother. The purpose of this step is to process the data, normalize it, and facilitate subsequent steps. We use Generalized cross-validation (GCV) as a reference to evaluate the smoother fitting effect. This does not mean using GCV as the only evaluation criterion, but when the GCV is too large, we will consider not using the corresponding smoother type. After testing, we chose to use B-spline smoother for fitting this data set.

The second step of the algorithm is to use the projector to obtain the eigenvalue of the data. We provide a variety of projection methods including PCA, KPCA, FPCA, and orthogonal expansion. And can freely select parameters such as B-spline, Fourier, monomial, and wavelet. Users can make choices based on the specific conditions of the data to get better results. For this data, we finally chose PCA as the projector. And result shown as table 4.7:

Table 4.7: AMI and ARI Scores for Dataset Arrowhead

	FunGP	FunHDDC	FunFEM	FunClust
AMI	0.305	0.198	0.031	0.007
ARI	0.239	0.290	0.035	0.040

FunGP shows the highest AMI score (0.305) among the four methods, indicating it has the best performance in terms of mutual information adjusted for chance. Its ARI score (0.239) is competitive but not the highest, suggesting it has a good but not the best performance in terms of finding a similar group structure as the truth, with adjustments for chance. In conclusion, FunGP appears to be the most effective at capturing mutual information between the true and the predicted clusterings, while FunHDDC is slightly better at matching the exact cluster groups. FunFEM and FunClust significantly lag behind in performance according to these metrics. The choice between FunGP and FunHDDC might depend on whether the emphasis is on capturing the overall information shared between clusterings (FunGP) or the exactness of the cluster matching (FunHDDC).

Bird Chicken The Bird Chicken dataset exhibits greater fluctuations, as indicated by the larger derivative of the data, and its variation more closely aligns with the characteristics of wavelets compared to the Arrowhead dataset. This suggests that wavelet-based methods may be more effective for capturing the dynamics of Bird Chicken data due to its sharper and more frequent changes.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'kernel', smoother_args = {'bandwidth': 0.01})
2 projector(projection_method = 'kpca', projection_args = {'n_components': 12})
```

Listing 4.9: Smoother and Projector Arguments for Dataset Bird Chicken

The data fluctuated violently, so we finally chose to use kernel smoother and KPCA as the smoother and projector. The result is shown in the table 4.8:

The result of each clustering method is in figure 4.12.

Both FunGP and FunFEM show the best performance in terms of mutual information with the ground truth, with FunGP slightly outperforming others in terms of the Rand index,

Table 4.8: AMI and ARI Scores for Dataset Bird Chicken

	FunGP	FunHDDC	FunFEM	FunClust
AMI	0.066	0.002	0.066	-0.015
ARI	0.070	0.002	0.048	-0.012

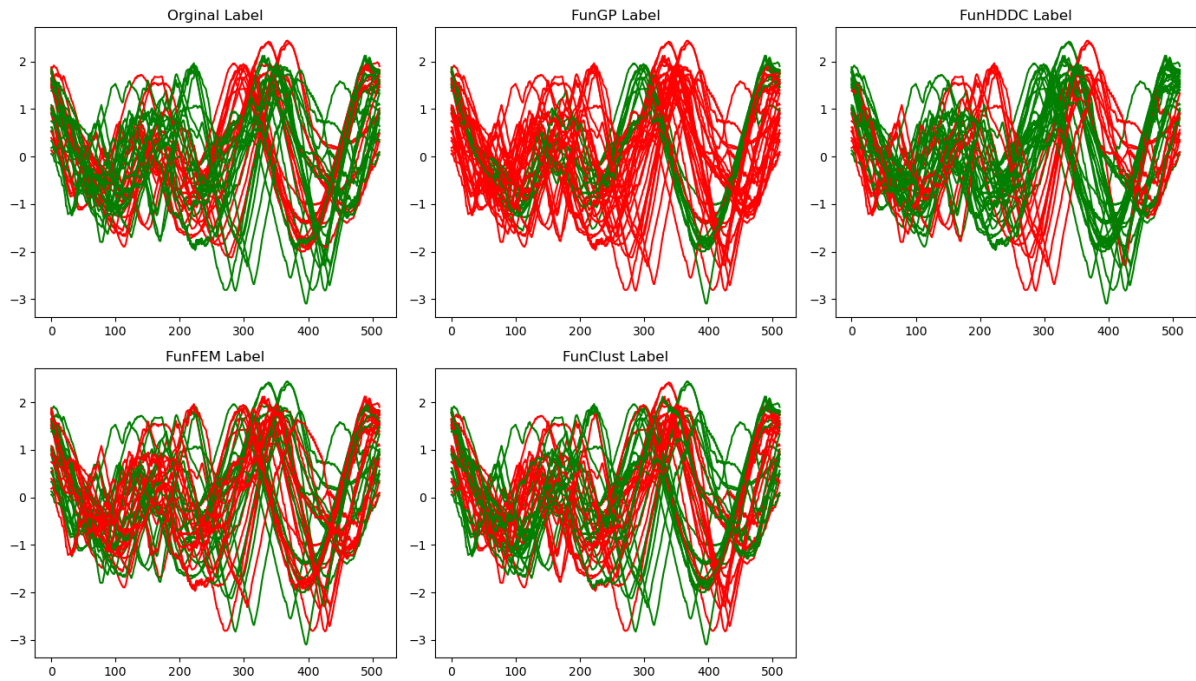


Figure 4.12: Clustering Results for Dataset Bird Chicken

indicating better accuracy in the exactness of cluster matching. FunHDDC shows almost no practical agreement with the ground truth, and FunClust actually performs worse than random clustering would, indicating a potential mismatch between the method and the data.

CBF The fluctuations in the CBF dataset are much more pronounced, exhibiting almost violent up-and-down swings even compared to the Bird Chicken dataset. This extreme variability makes it challenging to achieve a good fit using methods such as spline and orthogonal expansion. In contrast, wavelet and kernel-based approaches perform relatively better, as they are more adept at capturing the rapid changes and irregular patterns present in the CBF data.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'wavelet', smoother_args = {'wavelet': 'db4',
  'level': 4})
2 projector(projection_method = 'expansion', projection_args = {'
  basis': 'wavelet', 'n_basis': 15, 'wavelet': 'db4'})
```

Listing 4.10: Smoother and Projector Arguments for Dataset CBF

This set of data has strong fluctuations, which is more consistent with wavelet. The result is shown in the table 4.9:

Table 4.9: AMI and ARI Scores for Dataset CBF

	FunGP	FunHDDC	FunFEM	FunClust
AMI	0.455	0.326	0.008	0
ARI	0.462	0.331	0.008	0

The result of each clustering method is in figure 4.13.

FunGP demonstrates the highest performance among the compared methods, with AMI and ARI scores of 0.455 and 0.462, respectively. These scores suggest a moderate to good level of agreement between the FunGP clustering results and the ground truth. This indicates that FunGP is quite effective in capturing the true cluster structure of the data. FunHDDC also shows a reasonable performance but not as high as FunGP. In contrast, FunFEM and FunClust struggle to find any meaningful structure in the data, with FunClust showing no effectiveness at all.

Diatom Size Reduction The Diatom Size Reduction dataset is not inherently complex, but its challenge lies in the lack of strong feature distinction between the different classes. Despite this, by selecting an appropriate smoother and projector, FunGP is able to more effectively capture the subtle characteristics of the data, leading to improved classification

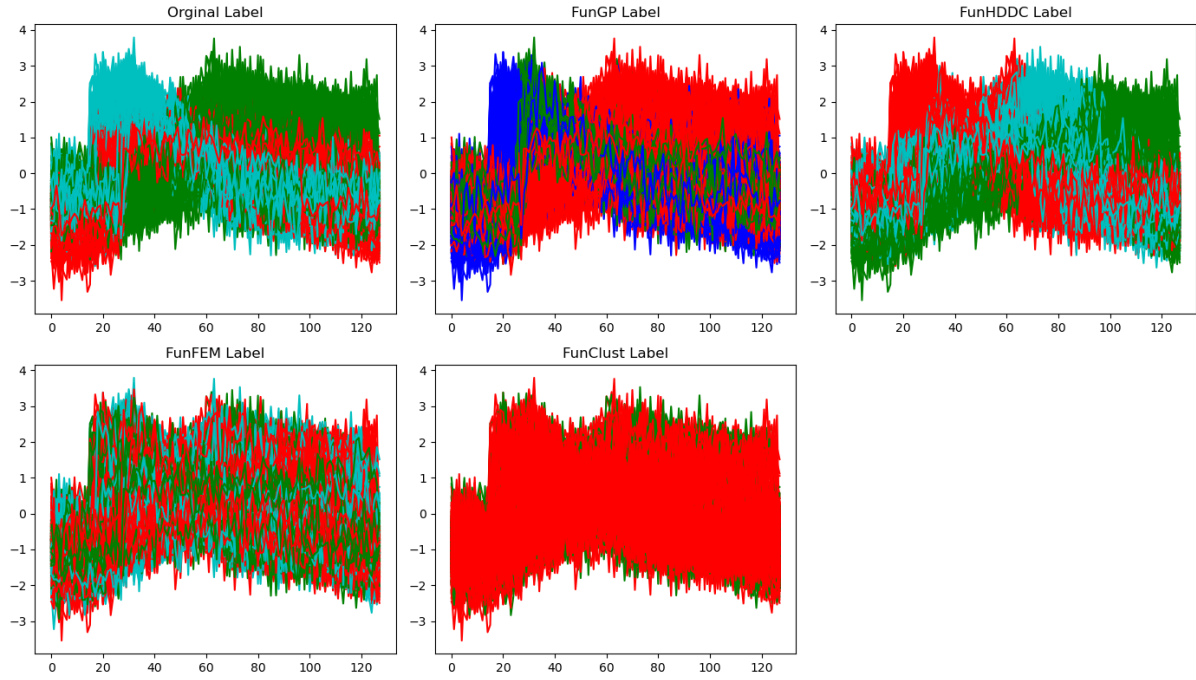


Figure 4.13: Clustering Results for Dataset CBF

performance. This highlights the importance of careful parameter tuning in enhancing the model's ability to differentiate between classes with minimal distinguishing features.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'bspline', smoother_args = {'degree': 3, '
    n_basis': 50})
2 projector(projection_method = 'fpca', projection_args = {'
    n_components': 10, 'basis': 'monomial', 'n_basis': 10})
```

Listing 4.11: Smoother and Projector Arguments for Dataset Diatom Size Reduction

The result is shown in the table 4.10:

Table 4.10: AMI and ARI Scores for Dataset Diatom Size Reduction

	FunGP	FunHDDC	FunFEM	FunClust
AMI	0.875	0.431	0.759	-0.001
ARI	0.905	0.589	0.786	-0.001

The result of each clustering method is in figure 4.14.

FunGP exhibits the highest performance across both metrics, with an AMI score of 0.875 and an ARI score of 0.905. These scores indicate a very high level of agreement between the FunGP clustering results and the ground truth, suggesting that FunGP is highly effective in accurately identifying the true cluster structure within the dataset. FunGP and FunFEM emerge as the top-performing methods for this dataset, with FunGP showing the best overall

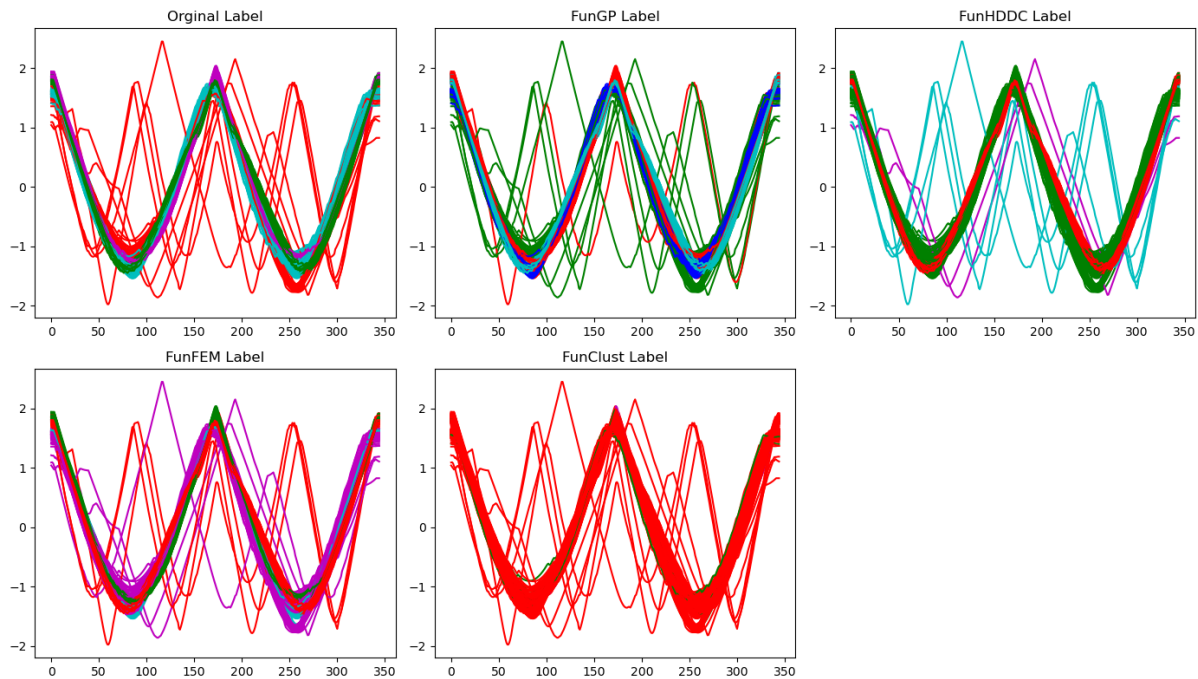


Figure 4.14: Clustering Results for Dataset Diatom Size Reduction

performance. FunHDDC offers moderate accuracy, while FunClust appears to be unsuitable for the task based on these metrics.

ECG200 The two classes of the ECG200 dataset are almost entangled, making classification particularly challenging. While FunGP appears to separate the two classes more distinctly, this separation does not align with the actual structure of the original data, resulting in relatively poor classification performance. This suggests that, in this case, the model may be overfitting or capturing patterns that do not correspond to the true class distinctions, highlighting a limitation in handling such closely intertwined data.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'wavelet', smoother_args = {'wavelet': 'db4',
2           'level': 4})
2 projector(projection_method = 'kpca', projection_args = {'n_components': 10})
```

Listing 4.12: Smoother and Projector Arguments for Dataset ECG200

The result is shown in the table 4.11:

Table 4.11: AMI and ARI Scores for Dataset ECG200

	FunGP	FunHDDC	FunFEM	FunClust
AMI	0.199	0.251	0.066	-0.029
ARI	0.303	0.161	0.153	0.004

The result of each clustering method is in figure 4.15.

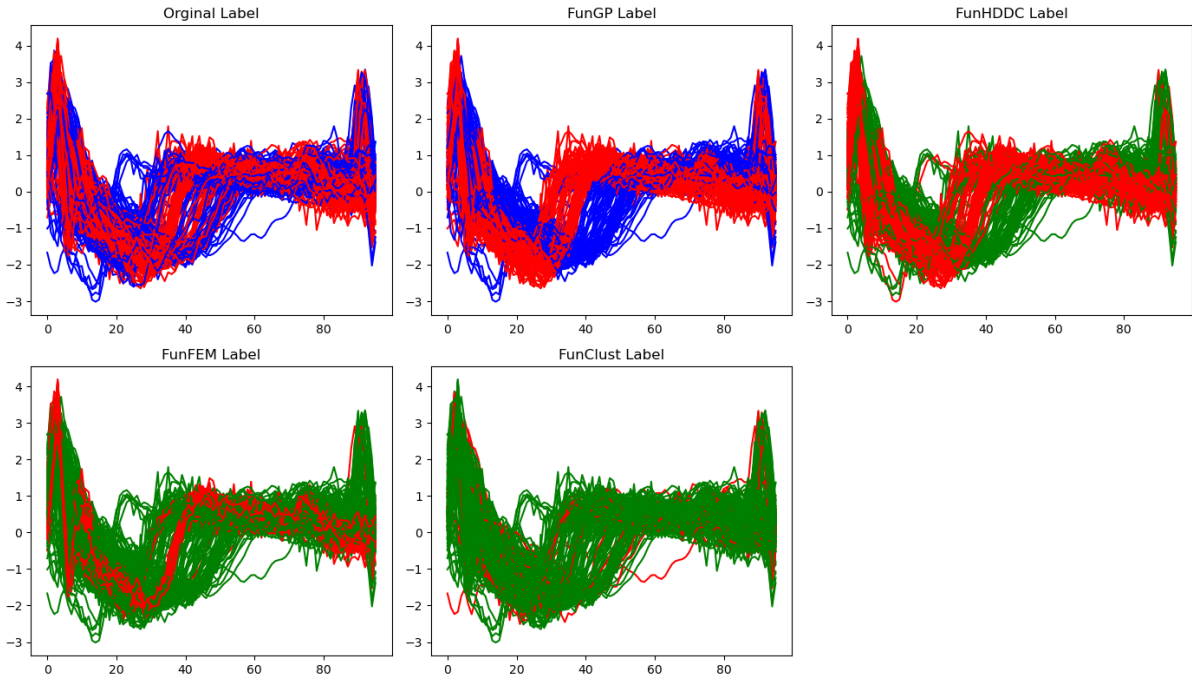


Figure 4.15: Clustering Results for Dataset ECG200

While FunHDDC exhibits the highest AMI score, suggesting a better overall understanding of the cluster structure, FunGP leads in terms of ARI, indicating a greater accuracy in the exact matching of clusters to the ground truth. FunFEM and particularly FunClust lag behind, with FunClust showing the least effectiveness in clustering according to these metrics.

Face Four Face Four is a relatively challenging dataset. As observed from visual analysis, the data seems to exhibit more distinct characteristics in the first half, where patterns are clearer. However, in the latter half, the data becomes increasingly entangled, making it difficult to differentiate between classes. This mixture of clear and overlapping features presents a significant challenge for effective classification.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'wavelet', smoother_args = {'wavelet': 'db4',
2           'level': 4})
3 projector(projection_method = 'fpca', projection_args = {'n_components': 15, 'basis': 'fourier', 'n_basis': 15})
```

Listing 4.13: Smoother and Projector Arguments for Dataset Face Four

The result is shown in the table 4.12:

The result of each clustering method is in figure 4.16.

Table 4.12: AMI and ARI Scores for Dataset Face Four

	FunGP	FunHDDC	FunFEM	FunClust
AMI	0.434	0.363	0.209	0.006
ARI	0.373	0.465	0.227	0.014

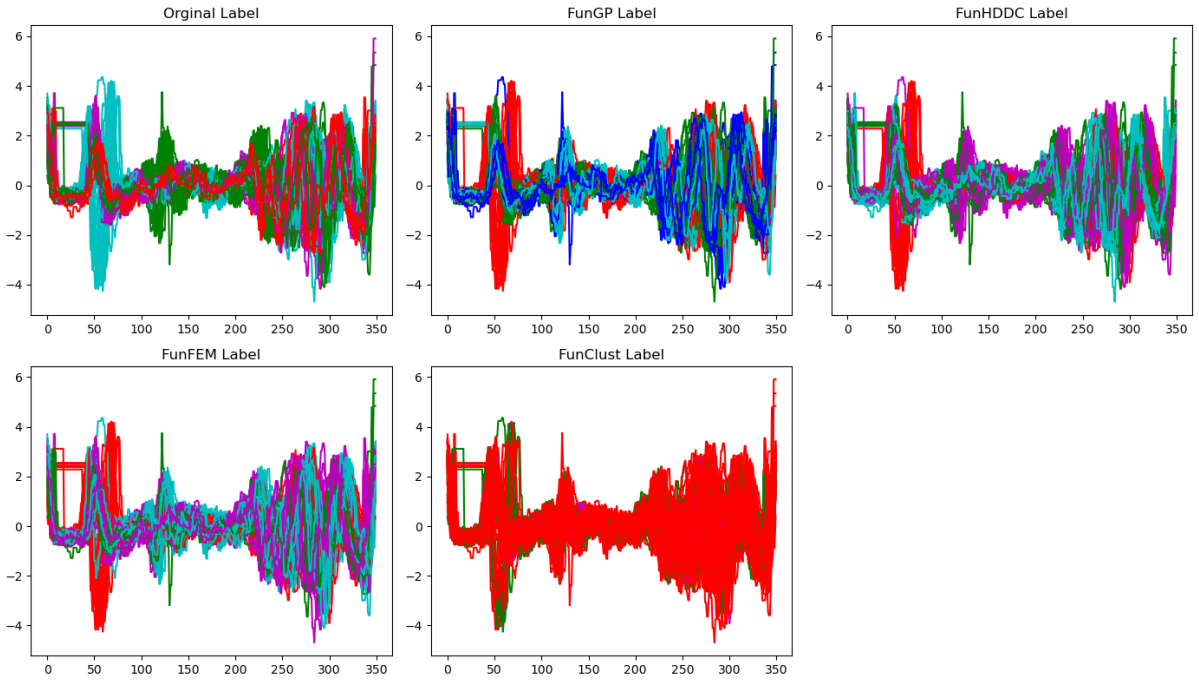


Figure 4.16: Clustering Results for Dataset Face Four

FunGP demonstrates a good balance between capturing the true cluster structure (as indicated by AMI) and accurately matching the clusters to the ground truth (as indicated by ARI). FunHDDC, while slightly lower in mutual information, shows the highest precision in cluster matching, making it potentially the best choice for tasks where precision in cluster assignment is critical. FunFEM shows limited effectiveness, and FunClust appears to be largely ineffective for the task at hand, based on these metrics.

Gun Point Gun Point is a dataset that shares some similarities with Bird Chicken, as the distinguishing characteristics between its two classes are challenging to capture. In this case, FunGP appears to have identified a pattern, but the pattern does not accurately reflect the true underlying structure of the data. This results in misclassification or suboptimal performance, indicating that the model may be overfitting or focusing on irrelevant features, rather than correctly discerning the subtle differences between the classes.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'kernel', smoother_args = {'bandwidth': 0.01})
2 projector(projection_method = 'expansion', projection_args = {'basis': 'wavelet', 'n_basis': 15, 'wavelet': 'db4'})
```

Listing 4.14: Smoother and Projector Arguments for Dataset Gun Point

The result is shown in the table 4.13:

Table 4.13: AMI and ARI Scores for Dataset Gun Point

	FunGP	FunHDDC	FunFEM	FunClust
AMI	0.256	-0.005	-0.005	0.003
ARI	0.200	-0.004	-0.004	0.008

The result of each clustering method is in figure 4.17.

FunGP emerges as the clear leader in both capturing the true structure of the data and in accurately matching clusters to the ground truth. The other methods (FunHDDC, FunFEM, and FunClust) show performance near or below the level of random chance, with FunHDDC and FunFEM having negative scores, indicating a lack of effective clustering. FunClust, despite a very low positive score, still indicates an almost negligible effectiveness.

Meat Identifying characteristic patterns in the Meat dataset by the naked eye is challenging; however, both FunGP and FunHDDC are able to accurately capture the data's fluctuations, enabling them to perform effective clustering. Their ability to detect subtle variations in the data allows these methods to overcome the limitations of visual analysis, resulting in robust clustering outcomes. This highlights the strength of these algorithms in handling datasets with less obvious features.

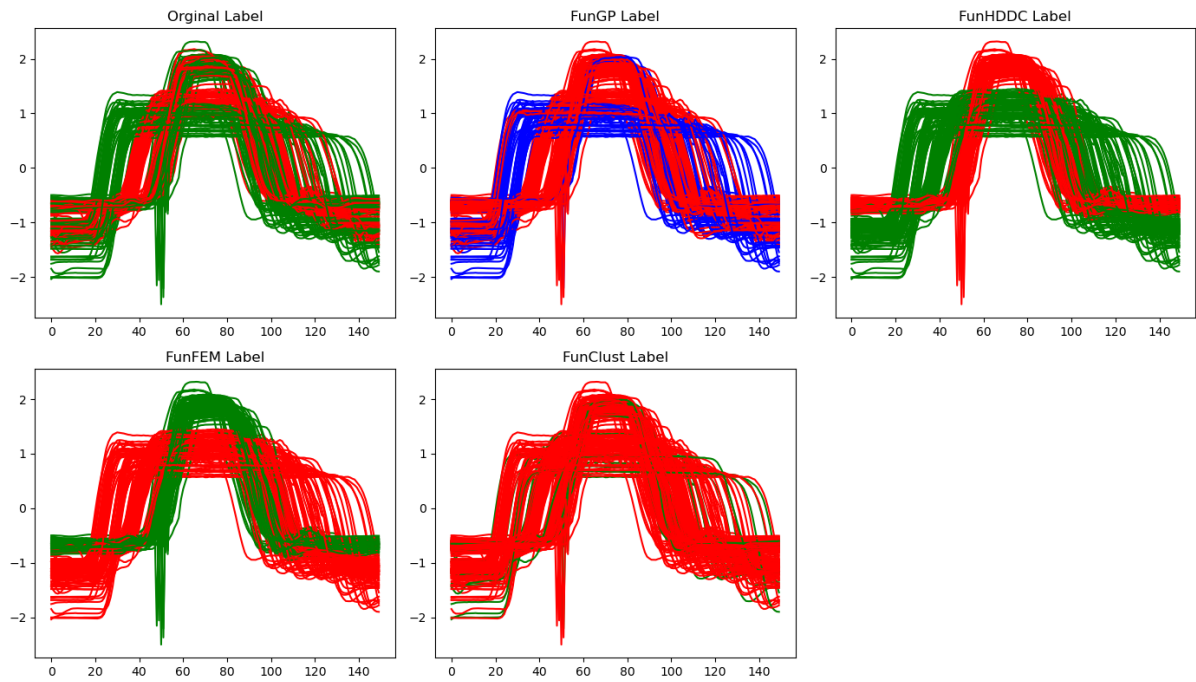


Figure 4.17: Clustering Results for Dataset Gun Point

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'bspline', smoother_args = {'degree': 3, '
    n_basis': 50})
2 projector(projection_method = 'pca', projection_args = {'
    n_components': 15})
```

Listing 4.15: Smoother and Projector Arguments for Dataset Meat

The result is shown in the table 4.14:

Table 4.14: AMI and ARI Scores for Dataset Meat

	FunGP	FunHDDC	FunFEM	FunClust
AMI	0.740	0.681	0.490	0.192
ARI	0.785	0.701	0.615	0.256

The result of each clustering method is in figure 4.18.

FunGP emerges as the top performer, with high scores in both AMI and ARI, indicating its superior ability to accurately identify and match the true cluster structure. FunHDDC also shows strong performance, albeit slightly behind FunGP. FunFEM offers moderate effectiveness, with its performance being noticeably lower than the top two methods but still significant. FunClust, on the other hand, demonstrates limited effectiveness in both capturing the true cluster structure and accurately matching clusters to the ground truth.

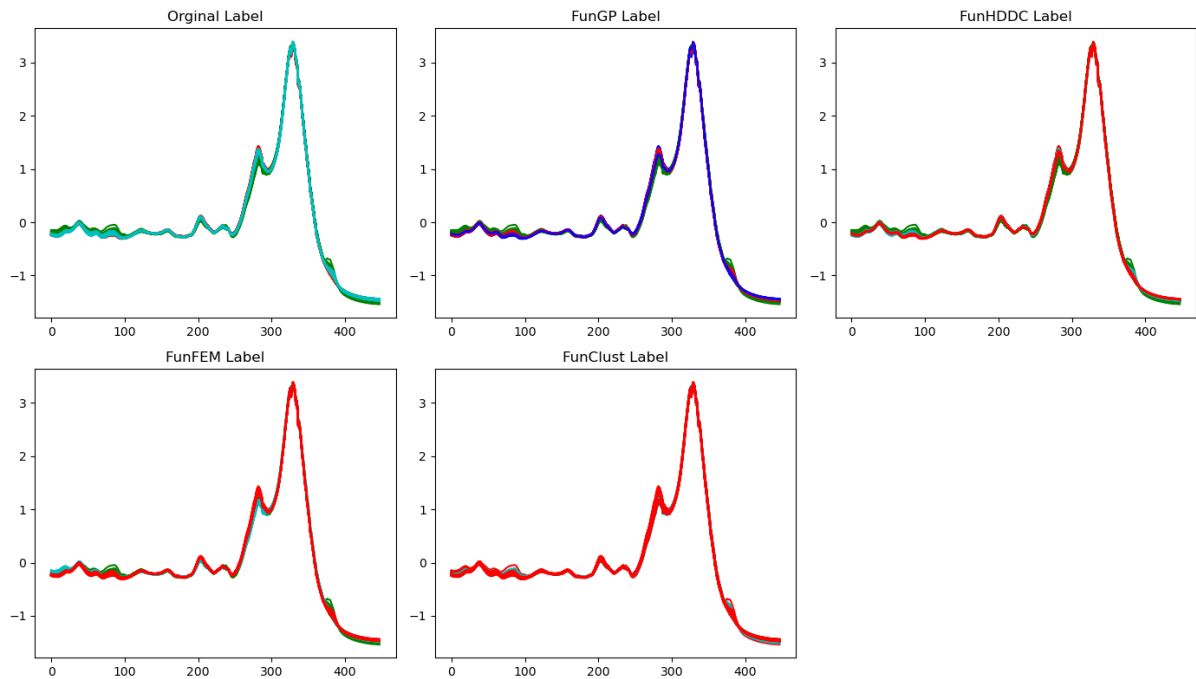


Figure 4.18: Clustering Results for Dataset Meat

Small Kitchen Appliances The Small Kitchen Appliances dataset is particularly challenging due to its unique characteristics, where each class fluctuates around a value of zero. This lack of distinct separation between classes presents significant difficulties for clustering algorithms, as the subtle variations are difficult to discern, leading to potential misclassification or poor clustering performance.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'wavelet', smoother_args = {'wavelet': 'db3',
2           'level': 4})
3 projector(projection_method = 'kpca', projection_args = {'
4           n_components': 10})
```

Listing 4.16: Smoother and Projector Arguments for Dataset Small Kitchen Appliances

The result is shown in the table 4.15:

Table 4.15: AMI and ARI Scores for Dataset Small Kitchen Appliances

	FunGP	FunHDDC	FunFEM	FunClust
AMI	0.227	0.014	0.047	0
ARI	0.219	0.061	0.084	-0.001

The result of each clustering method is in figure 4.19.

FunGP appears to be the most effective method among the four, with low to moderate scores in both AMI and ARI, suggesting some capability in accurately identifying and

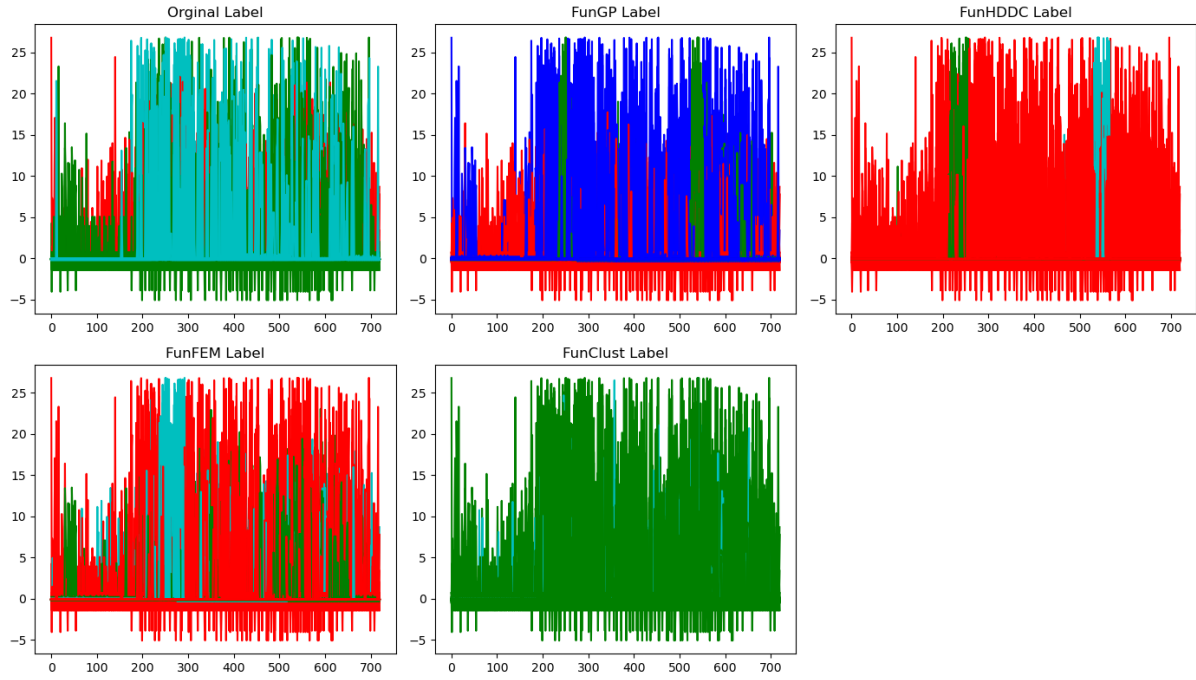


Figure 4.19: Clustering Results for Dataset Small Kitchen Appliances

matching the true cluster structure. FunHDDC, FunFEM, and FunClust show limited to no effectiveness.

Symbols The Symbol dataset is relatively easy to distinguish, as several distinct classes are clearly observable. The data structure appears to resemble a combination of polynomials, which simplifies the task for clustering algorithms. The clear separation between classes and the smooth nature of the data make this dataset more straightforward for analysis and classification compared to more complex or entangled datasets.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'bspline', smoother_args = {'degree': 3, '
    n_basis': 50})
2 projector(projection_method = 'kpca', projection_args = {'
    n_components': 10})
```

Listing 4.17: Smoother and Projector Arguments for Dataset Symbols

The result is shown in the table 4.16:

Table 4.16: AMI and ARI Scores for Dataset Symbols

	FunGP	FunHDDC	FunFEM	FunClust
AMI	0.908	0.704	0.535	0
ARI	0.784	0.795	0.659	0

The result of each clustering method is in figure 4.20.

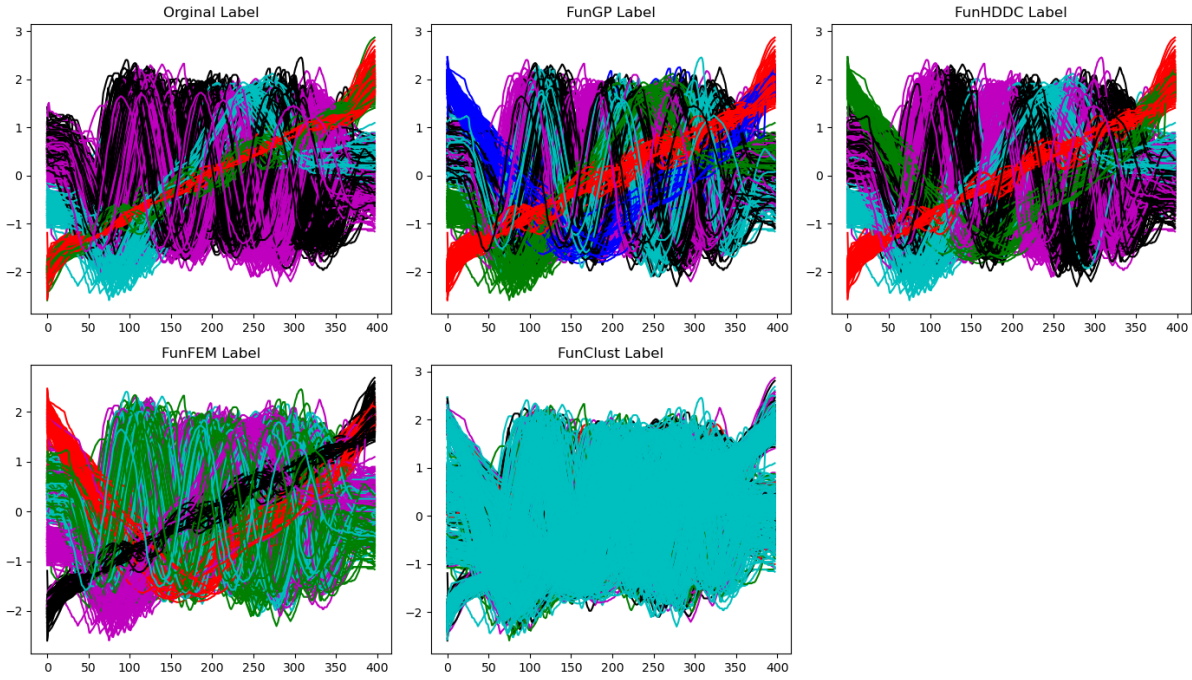


Figure 4.20: Clustering Results for Dataset Symbols

FunGP emerges as the top performer, showing the best ability to both identify and accurately match the true cluster structure. FunHDDC follows closely, with a slight edge in ARI, indicating its strong capability in cluster matching. FunFEM shows moderate effectiveness, and FunClust demonstrates no effectiveness in capturing or matching the true cluster structure according to these metrics.

Synthetic Control The Synthetic Control dataset presents an interesting challenge. Compared to other datasets, its primary difficulty lies in the data's resistance to smoothing due to inherent complexities. The fluctuation functions exhibit pronounced ups and downs, moving distinctly in different directions. This characteristic makes it challenging for clustering algorithms that are inclined to identify local features, as they may struggle to capture the global patterns within the data.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'fourier', smoother_args = {'n_basis': 10, '
  period': 3})
2 projector(projection_method = 'expansion', projection_args = {'
  basis': 'bspline', 'n_basis': 15, 'degree': 3})
```

Listing 4.18: Smoother and Projector Arguments for Dataset Synthetic Control

The result is shown in the table 4.17:

The result of each clustering method is in figure 4.21.

Table 4.17: AMI and ARI Scores for Dataset Synthetic Control

	FunGP	FunHDDC	FunFEM	FunClust
AMI	0.530	0.524	0.034	0.166
ARI	0.409	0.701	0.050	0.359

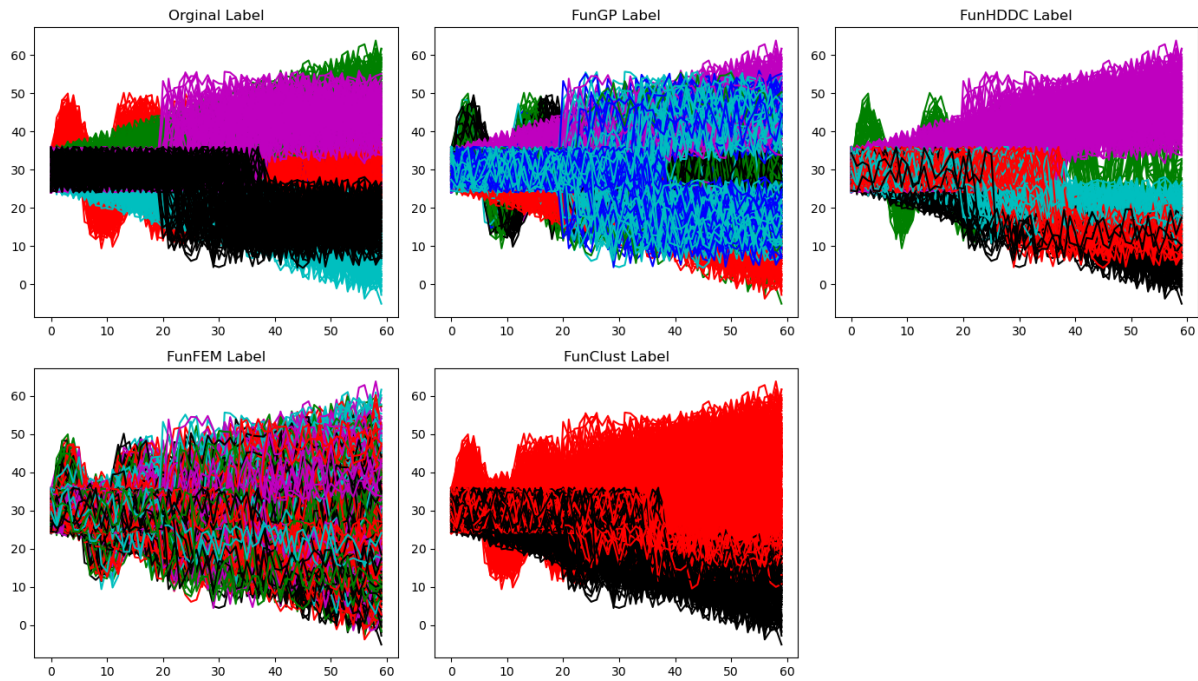


Figure 4.21: Clustering Results for Dataset Synthetic Control

FunHDDC stands out with the highest ARI score, indicating it has the best precision in clustering accuracy, despite its AMI score being closely competitive with FunGP. FunGP shows moderate effectiveness across both metrics, offering a balanced performance. FunFEM appears to be the least effective, with minimal mutual information and precision, while FunClust, despite its lower AMI, manages a relatively high ARI score, indicating a peculiar discrepancy between its ability to capture the true cluster structure and its precision in cluster assignment.

Worms The Worms dataset is highly disordered when observed, making it difficult to capture its underlying characteristics during experimentation. As a result, the final clustering outcomes are similarly messy, highlighting a limitation of FunGP in handling such complex and irregular data. This suggests that FunGP may struggle with datasets that lack clear structure or exhibit significant variability, underscoring the need for further refinement in dealing with noisy and chaotic data patterns.

The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'wavelet', smoother_args = {'wavelet': 'db3',
2           'level': 4})
3 projector(projection_method = 'expansion', projection_args = {'basis': 'wavelet', 'n_basis': 15, 'wavelet': 'db4'})
```

Listing 4.19: Smoother and Projector Arguments for Dataset Worms

The result is shown in the table 4.18:

Table 4.18: AMI and ARI Scores for Dataset Worms

	FunGP	FunHDDC	FunFEM	FunClust
AMI	0.123	0.012	0.006	-0.035
ARI	0.036	0.046	0.050	0.026

The result of each clustering method is in figure 4.22.

In summary, all methods show limited effectiveness in this clustering task, with relatively low scores across both AMI and ARI. In comparison, FunGP and FunHDDC are slightly better than other algorithms, but still have limited effects.

4.2.3 Real Data Conclusion

Based on the clustering performance across various datasets, FunGP consistently demonstrates superior performance, particularly in datasets with complex or fluctuating data structures such as CBF, Diatom Size Reduction, and Symbols. Its flexibility in handling

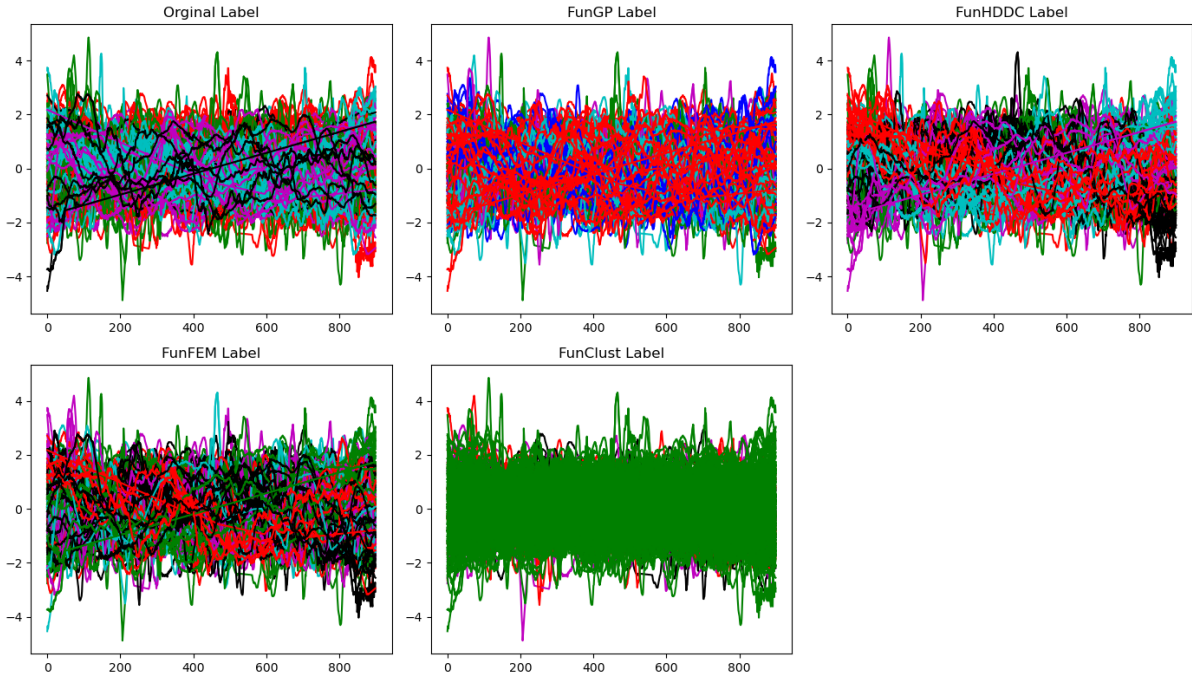


Figure 4.22: Clustering Results for Dataset Worms

diverse data characteristics makes it the most robust method in this analysis. In the table 4.19 4.20 below I show all the real data results.

Data	FunGP	FunHDDC	FunFEM	FunClust
ArrowHead	0.239	0.290	0.035	0.040
BirdChicken	0.070	0.002	0.048	-0.012
CBF	0.462	0.331	0.008	0.000
DiatomSizeReduction	0.905	0.589	0.786	-0.001
ECG200	0.303	0.161	0.153	0.004
FaceFour	0.373	0.465	0.227	0.014
GunPoint	0.200	-0.004	-0.004	0.008
Meat	0.785	0.701	0.615	0.256
SmallKitchenAppliances	0.219	0.061	0.084	-0.001
Symbols	0.784	0.795	0.659	-0.000
SyntheticControl	0.409	0.701	0.050	0.359
Worms	0.036	0.046	0.051	0.026

Table 4.19: ARI Scores for Real Data

AMI is more sensitive to the distribution of cluster sizes and captures the mutual information shared between two sets of clusters, while ARI focuses on pairwise relationships and measures how consistently pairs of points are placed in the same or different clusters. AMI can be less biased towards specific cluster sizes, making it more robust in cases where there are imbalanced clusters, whereas ARI is sensitive to such imbalances, particularly when there are many small clusters.

Data	FunGP	FunHDDC	FunFEM	FunClust
ArrowHead	0.305	0.198	0.031	0.007
BirdChicken	0.066	0.002	0.066	-0.016
CBF	0.455	0.326	0.008	0.000
DiatomSizeReduction	0.875	0.431	0.759	-0.001
ECG200	0.199	0.251	0.066	-0.029
FaceFour	0.434	0.363	0.209	0.006
GunPoint	0.256	-0.005	-0.005	0.003
Meat	0.740	0.682	0.490	0.192
SmallKitchenAppliances	0.227	0.014	0.047	-0.000
Symbols	0.908	0.703	0.535	-0.001
SyntheticControl	0.530	0.524	0.034	0.166
Worms	0.123	0.012	0.006	-0.035

Table 4.20: AMI Scores for Real Data

Since FunGP has consistently achieved the highest or near-best AMI scores across most datasets, it suggests that the algorithm excels in capturing the overall clustering structure and distribution of the data, even in complex scenarios. A high AMI score means that FunGP aligns well with the true clustering, preserving the essential relationships between clusters.

However, AMI's high scores must also be considered alongside ARI. For instance, even if FunGP has the best AMI scores, ARI scores should be examined to assess how well the algorithm places pairs of points into the same or different clusters. The fact that FunGP has almost the best AMI scores indicates that it is proficient at finding clusters that closely align with the true underlying structure of the data, capturing the overall distribution and relationship between clusters. This makes FunGP a robust choice for functional data clustering, especially in scenarios where the global structure is more informative than local pairwise relationships.

FunGP excels at identifying the broad clustering structure (as seen in its high AMI scores) but may fall short in handling finer pairwise distinctions (reflected in its lower ARI scores). This suggests it is best suited for tasks where global patterns matter most, but may require additional refinement or complementary methods in cases where local relationships are critical.

In the application of FunGP, parameter adjustment plays a crucial role in determining the final clustering results. While this flexibility allows FunGP to adapt to various datasets, it also increases the time cost. Our strategy for parameter selection begins by identifying three relatively effective smoothers. This is necessary because certain smoothers may perform poorly in specific scenarios. For instance, using B-spline as a smoother in the highly volatile Face Four dataset results in suboptimal performance due to its inability to capture the

data’s sharp fluctuations.

After selecting the three most suitable smoothers, we systematically perform permutations and combinations with different projectors to determine the optimal pairing. This approach ensures that the best combination of smoothers and projectors is selected for further detailed hyperparameter tuning. Interestingly, in practice, we observed that the choice of smoother has a limited impact on the final result. The fine-tuning of specific internal parameters within the smoother, rarely leads to significant changes in clustering performance. I discuss this in detail in Appendix A2 sensitivity analysis. This suggests that while selecting a general class of smoother is important, the detailed adjustments of smoothers may not be as critical to the outcome as the choice of projector or other parameters. The results of FunGP are more significantly influenced during the projector stage. As a result, we invest considerable effort into performing exhaustive enumeration and fine-tuning at this stage to ensure optimal performance. This detailed tuning process, while necessary for achieving the best results, represents a key limitation of our method. The flexibility in choosing and adjusting projectors allows FunGP to adapt well to different datasets, but it also introduces increased complexity and time costs. However, even with the default settings, our algorithm still provides good results. This makes FunGP a practical choice for users with low performance requirements or time constraints. In such cases, the algorithm can be deployed quickly and efficiently, providing reasonable clustering results without extensive parameter tuning.

4.3 Conclusion

In conclusion, our investigation into algorithmic data analysis, specifically focused on functional data clustering using the FunGP algorithm, reveals promising outcomes across both simulated and real-world data scenarios. Through strategic adjustments in smoother and projector configurations, FunGP demonstrates a remarkable capacity to yield favorable results across diverse datasets. Unlike alternative methods such as FunHDDC, FunFEM, and FunClust, which primarily rely on smoother adjustments for fitting enhancements, FunGP stands out for its comprehensive approach. By offering a unique blend of flexibility and robustness, FunGP surpasses these alternatives by enabling dynamic adaptations that significantly enhance clustering outcomes. This underscores the pivotal role of FunGP in advancing the efficacy and versatility of functional data clustering methodologies, marking a significant step forward in algorithmic data analysis.

5 Discussion

The model presented in this research introduces a nuanced approach to clustering functional data, leveraging the strengths of the Gaussian Mixture Model. This section explores the concerns, challenges, impact, and contributions of our model and outlines directions for future work to enhance its applicability and performance.

5.1 Research Challenges

One of our initial goals was to design a model that simplifies user interaction to the point where data input leads directly to automatic clustering. However, this ideal was met with practical limitations. We recognized that such simplicity while appealing, could potentially compromise the model's ability to produce results that faithfully represent the underlying data structures. The essence of the challenge lies in the inherent trade-off between user convenience and the nuanced precision required for effective functional data analysis. Specifically, on the topic of smoothers, we explored the use of Generalized Cross Validation (GCV) as a criterion for assessing smoothing quality. While GCV provides valuable insights, it falls short of serving as a definitive guide for several reasons: 1. GCV offers a quantifiable measure of smoothing effectiveness by balancing the fidelity to the data against the smoothness of the function. However, it does not account for all aspects of functional data complexity, such as varying noise levels across the domain or the presence of underlying patterns that might be obscured by over-smoothing; 2. Sensitivity to Data Characteristics: Functional data can exhibit diverse characteristics, including but not limited to periodicity, abrupt changes, and multimodal distributions. These features require more nuanced consideration than a single metric like GCV can provide. The challenges compound when evaluating the effectiveness of projectors, where no universally accepted metric exists. The absence of a clear benchmark underscores the difficulty in automating the selection of projection methods. So our final choice is to provide all methods for users to choose. When solving practical problems, researchers have a certain sensitivity to data.

Verifying the FunGP algorithm posed significant challenges due to its high flexibility and numerous combinations of smoothers and projectors, each with tunable parameters. With

four smoothers and dozens of projectors, systematically evaluating performance was complex. To manage this, we first selected an appropriate smoother based on fit quality, then explored the impact of different projectors. A key finding was the relative insensitivity of clustering results to the choice of smoother, as long as it achieved sufficient data fit. This insight streamlined the process, reducing the need to explore the entire parameter space. However, fine-tuning the remaining parameters remains crucial for achieving optimal clustering outcomes, as the results are highly dependent on dataset-specific characteristics.

In conclusion, the high flexibility of our model is both our advantage and our disadvantage. Users are faced with the task of selecting from an array of smoothers and projectors, each with its own set of parameters. This selection process requires a foundational understanding of the data's characteristics and how different methods might interact with these characteristics. We do provide recommended initial parameters based on common situations, but real-world problems are typically more challenging than this. But in any case, after our evaluation, the fitness and clustering results of our model are better than those of the algorithms we compared in most cases.

The choices for recommend smoother and projector are as follows:

```
1 smoother(smoother = 'bspline', smoother_args = {'degree': 3, '
    n_basis': 10})
2 projector(projection_method = 'fpca', projection_args = {'
    n_components': 10})
```

Listing 5.1: Smoother and Projector Arguments for Default Setting

5.2 Contribution

The first contribution of our model is we build a sophisticated clustering framework specifically designed for functional data. By integrating Gaussian processes with a mixture model approach, FunGP offers a robust solution to the clustering of data characterized by complex, continuous, and high-dimensional structures. This methodological innovation enables more nuanced and accurate clustering than traditional approaches.

The second contribution of our model is the model's structure. We build distinct layers for smoothing, projecting, and clustering, representing a comprehensive approach to preprocessing and analyzing functional data. At each layer, we can analyze the data individually and better understand the data. At the same time, if there are certain needs, each layer of our model can also be used independently, which increases the usability of our model. The flexibility to choose from multiple smoothers and projectors allows for customized processing of the data to meet its specific properties and ensures that the clustering process is efficient.

Finally, we also pioneered several innovative ideas for clustering functional data. Except for the conventional use of smoothing and projection techniques, we incorporate UniGMM as a clustering method. Referred to the method of Dasgupta [35], Maitra and Melnykov [36], we incorporate Dasgupta's insights into learning mixtures of Gaussians and Maitra and Melnykov's methods to obtain overlap. Our model supports this theory is referable and may become an auxiliary method for clustering functional data.

5.3 Future Work

Improving the Functional Gaussian Process Mixture Model encompasses enhancements across three critical dimensions: breadth, depth, and interpretability. These dimensions collectively contribute to the model's efficacy, usability, and applicability in addressing complex functional data analysis challenges. By focusing on these areas, the model can be significantly refined to better meet the needs of researchers and practitioners in various domains.

Breadth: Expanding the Toolkit. Incorporating a wider array of smoothers and projectors can cater to a broader set of data characteristics. By offering a more diverse set of tools, users can select the most appropriate methods for their specific data, enhancing the model's versatility. Introducing alternative clustering methods, including those based on non-Gaussian assumptions or capable of capturing more complex data relationships, can broaden the model's applicability. This diversification allows FunGP to accommodate a wider variety of functional data types and clustering objectives. In its current form, our algorithm assumes that the number of classes is fixed and known, and does not account for scenarios where the number of classes is uncertain. This limitation has not been addressed in the present work. Future extensions could explore approaches to asynchronously estimate the number of classes and incorporate this into the clustering process. By allowing dynamic estimation of class numbers, the algorithm could generate multiple potential clustering solutions, offering a range of outcomes for user evaluation and selection. This would enhance the flexibility and applicability of the algorithm in real-world scenarios where class structures are not predefined.

Depth: Enhancing Efficiency and Optimization. Focusing on the optimization of existing algorithms within FunGP, including smoothers, projectors, and clustering methods, can lead to significant improvements in computational efficiency. Techniques such as algorithmic refinement, parallel processing, and resource allocation optimization are key to reducing computational demands. Developing methods for adaptive parameter selection that dynamically adjust based on data characteristics can streamline the model setup process and enhance performance. This approach minimizes the need for manual parameter tuning, making the model more accessible to users with varying levels of expertise.

Interpretability: Improving Understanding and Visualization. Incorporating additional analysis modules that can automatically identify and highlight key features of the data, such as trends, patterns, and anomalies, can aid in the pre-clustering exploration phase. These tools can provide valuable insights that inform the selection of smoothers, projectors, and clustering methods. Expanding the model's visualization capabilities to include more intuitive and informative plots can greatly enhance interpretability. Visualizations that clearly depict the effects of different preprocessing choices, the structure of identified clusters, and the relationships between clusters can facilitate a deeper understanding of the data and the clustering outcomes. Establishing a comprehensive framework for model interpretability that guides users through understanding how and why specific clustering decisions are made can reinforce trust in the model. Similar to the Scikit-fda extension [29], which offers a wide range of useful tools and visualization techniques for functional data analysis,

6 Conclusion

In conclusion, this thesis introduced the Functional Gaussian Process Mixture Model, a novel and comprehensive model designed to address the intricate challenges of clustering functional data. By innovatively combining Gaussian processes with mixture models, FunGP offers a sophisticated approach to understanding and categorizing complex data structures inherent in functional datasets. Throughout this work, we have explored the model's architecture, which is distinguished by its layers for smoothing, projecting, and clustering, each contributing to the model's overall efficacy and adaptability.

The development of FunGP was motivated by the need for a more nuanced and flexible approach to functional data analysis, one that accommodates the unique characteristics of such data, including its high-dimensionality and the intricate correlations within. By providing a versatile framework that integrates various smoothing techniques, projection methods, and the Unigmm clustering approach, FunGP provides a new framework that can be used in functional data analysis.

In our comprehensive evaluation of the FunGP, we employed an extensive dataset comprising both real-world and simulated data. This methodical approach allowed us to rigorously test the model's capabilities and assess its effectiveness across a variety of scenarios. The use of real data grounded our evaluation in practical, real-world applications, ensuring that our findings are relevant and applicable to the challenges faced by practitioners in the field.

Furthermore, we outlined strategic directions for future enhancements of FunGP, focusing on expanding the model's breadth through the inclusion of additional methods, deepening its efficiency and optimization, and enhancing its interpretability. These proposed improvements aim to augment the model's applicability, ease of use, and the insightfulness of its clustering outcomes.

In the end, FunGP offers a robust, flexible, and user-friendly tool for clustering functional data.

Bibliography

- [1] G. M. James and C. Sugar, "Clustering for sparsely sampled functional data," *Journal of the American Statistical Association*, vol. 98, pp. 397 – 408, 2003.
- [2] J. Jacques and C. Preda, "Model-based clustering for multivariate functional data," *Computational Statistics & Data Analysis*, vol. 71, pp. 92–106, 2014.
- [3] —, "Functional data clustering: a survey," *Advances in Data Analysis and Classification*, vol. 8, pp. 231–255, 2014.
- [4] J.-L. Wang, J.-M. Chiou, and H.-G. Müller, "Functional data analysis," *Annual Review of Statistics and its application*, vol. 3, pp. 257–295, 2016.
- [5] D. J. Levitin, R. L. Nuzzo, B. W. Vines, and J. Ramsay, "Introduction to functional data analysis." *Canadian Psychology/Psychologie canadienne*, vol. 48, no. 3, p. 135, 2007.
- [6] S. Ullah and C. F. Finch, "Applications of functional data analysis: A systematic review," *BMC medical research methodology*, vol. 13, pp. 1–12, 2013.
- [7] S. Sakakibara, "A practice of data smoothing by b-spline wavelets," vol. 5, pp. 179–196, 1994.
- [8] M. Jeong, J.-C. Lu, and N. Wang, "Wavelet-based spc procedure for complicated functional data," *International Journal of Production Research*, vol. 44, pp. 729 – 744, 2006.
- [9] M.-H. Descary and V. Panaretos, "Functional data analysis by matrix completion," *The Annals of Statistics*, 2016.
- [10] J. Yang, H. Zhu, T. Choi, and D. Cox, "Smoothing and mean-covariance estimation of functional data with a bayesian hierarchical model." *Bayesian analysis*, vol. 11 3, pp. 649–670, 2014.
- [11] D. B. Hitchcock, J. Booth, and G. Casella, "The effect of pre-smoothing functional data on cluster analysis," *Journal of Statistical Computation and Simulation*, vol. 77, pp. 1043 – 1055, 2007.

- [12] G. Wahba, *Spline models for observational data*. SIAM, 1990.
- [13] Y. Xi-xi, "Principal component analysis in functional data analysis," *Journal of Shanxi University*, 2011.
- [14] P. Hall and M. Hosseini-Nasab, "On properties of functional principal components analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, 2006.
- [15] H. Shang, "A survey of functional principal component analysis," *AStA Advances in Statistical Analysis*, vol. 98, pp. 121–142, 2014.
- [16] Z. Lin, L. Wang, and J. Cao, "Interpretable functional principal component analysis," *Biometrics*, vol. 72, 2016.
- [17] J. C. Lemm, "Mixtures of gaussian process priors," vol. 1, pp. 292–297, 1999.
- [18] C. Luo and S. Sun, "Variational mixtures of gaussian processes for classification," pp. 4603–4609, 2017.
- [19] J. Wilson, V. Borovitskiy, A. Terenin, P. Mostowsky, and M. Deisenroth, "Efficiently sampling functions from gaussian process posteriors," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 292–10 302.
- [20] C. Stachniss, C. Plagemann, and A. Lilienthal, "Learning gas distribution models using sparse gaussian process mixtures," *Autonomous Robots*, vol. 26, pp. 187–202, 2009.
- [21] X. Ou and E. Martin, "Batch process modelling with mixtures of gaussian processes," *Neural Computing and Applications*, vol. 17, pp. 471–479, 2008.
- [22] B. Everitt, "Cluster analysis," *Quality and Quantity*, 1974.
- [23] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognit. Lett.*, 2010.
- [24] M. Zhang and A. C. Parnell, "Review of clustering methods for functional data," *ACM Transactions on Knowledge Discovery from Data*, vol. 17, pp. 1 – 34, 2022.
- [25] E. Akeweje and M. Zhang, "Learning mixtures of gaussian processes through random projection," in *Forty-first International Conference on Machine Learning*, 2024.
- [26] C. Bouveyron and J. Jacques, "Model-based clustering of time series in group-specific functional subspaces," *Advances in Data Analysis and Classification*, vol. 5, pp. 281–300, 2011.
- [27] M. Soueidatt and *, *Funclustering: Functional Data Clustering*, 2018, r package version 1.0.2. [Online]. Available: <https://CRAN.R-project.org/package=Funclustering>

- [28] C. Bouveyron and J. Jacques, “funfem: an r package for functional data clustering,” in *Quatrième Rencontres R*, 2015.
- [29] C. Ramos-Carreño, J. L. Torrecilla, M. Carbajo-Berrocal, P. Marcos, and A. Suárez, “scikit-fda: A python package for functional data analysis,” 2023.
- [30] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.
- [31] L. Wasserman, *All of nonparametric statistics*. Springer Science & Business Media, 2006.
- [32] D. Borthwick, *Spectral theory*. Springer, 2020.
- [33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [34] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009.
- [35] S. Dasgupta, “Learning mixtures of gaussians,” *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, pp. 634–644, 1999.
- [36] R. Maitra and V. Melnykov, “Simulating data to study performance of finite mixture modeling and clustering algorithms,” *Journal of Computational and Graphical Statistics*, vol. 19, pp. 354 – 376, 2010.
- [37] A. Delaigle and P. Hall, “Defining probability density for a distribution of random functions,” vol. 38, no. 2, 2010. [Online]. Available: <http://arxiv.org/abs/1002.4931>
- [38] L. A. Garcia-Escudero and A. Gordaliza, “A proposal for robust curve clustering,” vol. 22, no. 2, pp. 185–201, 2005. [Online]. Available: <https://doi.org/10.1007/s00357-005-0013-8>
- [39] S. Golovkine, N. Klutchnikoff, and V. Patilea, “Clustering multivariate functional data using unsupervised binary trees,” vol. 168, p. 107376, 2022. [Online]. Available: <http://arxiv.org/abs/2012.05973>
- [40] Y. Meng, J. Liang, F. Cao, and Y. He, “A new distance with derivative information for functional k -means clustering algorithm,” vol. 463-464, pp. 166–185, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025518304766>
- [41] L. Ye and E. Keogh, “Time series shapelets: a new primitive for data mining,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge*

- discovery and data mining*, ser. KDD '09. Association for Computing Machinery, 2009, pp. 947–956. [Online]. Available: <https://doi.org/10.1145/1557019.1557122>
- [42] N. Saito, *Local feature extraction and its applications using a library of bases*. Yale University, 1994. [Online]. Available: <https://scholar.google.com/scholar?cluster=15742796369939077514&hl=en&oi=scholar>
- [43] R. T. Olszewski, “Generalized feature extraction for structural pattern recognition in time-series data.”
- [44] C. A. Ratanamahatana and E. Keogh, “Making time-series classification more accurate using learned constraints,” in *Proceedings of the 2004 SIAM International Conference on Data Mining (SDM)*, ser. Proceedings. Society for Industrial and Applied Mathematics, 2004, pp. 11–22. [Online]. Available: <https://epubs.siam.org/doi/10.1137/1.9781611972740.2>
- [45] O. Al-Jowder, E. Kemsley, and R. Wilson, “Mid-infrared spectroscopy and authenticity problems in selected meats: A feasibility study,” *Food Chemistry*, vol. 59, no. 2, pp. 195–201, Jun. 1997.
- [46] A. E. X. Brown, E. I. Yemini, L. J. Grundy, T. Jucikas, and W. R. Schafer, “A dictionary of behavioral motifs reveals clusters of genes affecting *caenorhabditis elegans* locomotion,” vol. 110, no. 2, pp. 791–796, 2013.
- [47] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, “Time-series classification with COTE: The collective of transformation-based ensembles,” vol. 27, pp. 1–1, 2015.

A1 Appendix: FunGP Introduction

A1.1 Initialisation

This section is about initialization before use. First we need to ensure that python, our model and the libraries listed below are installed in the running environment. And then we need to load some dependent libraries.

```
1 import numpy as np
2 import skfda
3 import pywt
4 import scipy
5 import sklearn
6 from smother.gpmix_smoother import smother
7 from projector.gpmix_projector import projector
8 from unigmm.gpmix_unigmm import unigmm
```

A1.2 Smoother

In order to use Smoother, that is, the data is smoothed by conversion and converted into a more suitable data type FDataGrid. FdataGrid is a data type from the skfda library, which is more convenient for the following operations.

```
1 smother1 = smother(smoother = 'bspline', smother_args = {'
    degree': 3, 'n_basis': 50})
2 fdate_smooth1 = smother1.fit(data)
```

```
1 smother2 = smother(smoother = 'fourier', smother_args = {'
    n_basis': 10, 'period': 3})
2 fdate_smooth2 = smother2.fit(data)
```

```
1 smother3 = smother(smoother = 'kernel', smother_args = {'
    bandwidth': 0.01})
2 fdate_smooth3 = smother3.fit(data)
```

```

1 smoother4 = smoother(smoother = 'wavelet', smoother_args = {'
    wavelet': 'db4', 'level': 4})
2 fdate_smooth4 = smoother4.fit(data)

```

```

1 smoother (str): The smoother to use (e.g., 'bspline', 'fourier', '
    kernel').
2 smoother_args (dict): Additional arguments for the smoother (e.g.,
    {'bandwidth': 0.1} for a kernel smoother).
3 domain_range (tuple): The domain range of the functional data.
4 X (ndarray): The input data to transform.

```

A1.3 Projector

A class that performs projection of an FDataGrid object using a specified projection method and basis.

```

1 projector = projector(projection_method = 'fpca', projection_args
    = {'n_components': 15, 'basis': 'bspline', 'n_basis': 15, '
    order': 3})
2 coefficients = projector.fit(fdate_smooth)

```

```

1 projection_method (str): The projection method to use (e.g., 'fpca
    ', 'pca', 'kpca', etc.).
2 projection_args (dict): Additional arguments for the projection
    method (e.g., {'n_components': 15, 'basis': 'bspline', 'n_basis
    ': 15, 'order': 3}).

```

A1.4 Unigmm

A class that performs clustering on projection coefficients using Gaussian Mixture Models (GMMs).

```

1 gmm = unigmm(n_components = n_labels)
2 gmms = gmm.fit(coefficients)
3 affinity_matrix = gmm.calculate_weights()
4 sc = SpectralClustering(n_clusters = n_labels, affinity='
    precomputed', assign_labels='discretize')
5 sc.fit(affinity_matrix)
6 labels = sc.labels_

```

A2 Appendix: Sensitivity Analysis

In this section, we perform a sensitivity analysis to demonstrate the robustness of FunGP's clustering results with respect to variations in smoother parameter settings. Specifically, we explore how changes in the internal parameters of the smoother affect the clustering performance when the same smoother and projector are used. This analysis supports the earlier claim that tuning the smoother's internal parameters has a limited impact on the final results.

A2.1 Experimental Setup

For this analysis, we use the same datasets as in the main experiments: Simulation Scenario 5. We fix the choice of smoother and projector and vary the internal parameters of the smoother. The choices for smoother and projector are as follows: The choices for smoother and projector are as follows:

```
1 smoother(smoother = 'fourier', smoother_args = {'n_basis': 10, '
   period': 3})
2 projector(projection_method = 'kpca', projection_args = {'
   n_components': 12})
```

Listing A2.1: Smoother and Projector Arguments for Simulation Scenario 5

In this experiment, we use the same parameters as those in the main experiment, but for the sensitivity analysis, we vary the internal parameters of the smoother. Specifically, the parameters *n_basis* (the number of basis functions) and *period* (the period of the Fourier basis) are tested across different values:

- *n_basis* is varied from 2 to 16, increasing by 2 at each step.
- *period* is varied from 1 to 5.

It should be noted that *n_basis* is increased by 2 each time because, for the Fourier basis function, due to its periodicity, for example, when the *period* remains unchanged, the basis functions obtained for *n_basis* = 1 and *n_basis* = 2 are identical.

A2.2 Results

The Adjusted Rand Index (ARI) and Adjusted Mutual Information (AMI) are used to measure the clustering performance for each combination of smoother parameters, In addition, we also use Generalized Cross-Validation (GCV) to help to evaluate the result. GCV is mentioned in the previous section 2.1.1 Functional Data Smoothing, which is a widely used method for selecting the optimal smoothing parameters. Generally speaking, the GCV score is a negative number, Lower GCV score suggests that the model fits the data well without being too complex, meaning the smoother has found an optimal balance between closely fitting the data and maintaining a good generalization to unseen data. Note that low GCV score does not always guarantee the best result for a given analysis. This is because GCV only focuses on balancing model fit and complexity and may not account for other aspects of model performance that are important in specific contexts.

Tables A2.1, A2.2, and A2.3 present the results for different values of n_basis and $Period$, showing the corresponding AMI, ARI, and GCV scores. The largest result in each row is bolded.

Table A2.1: AMI Score for Different n_basis and $Period$

n_basis	Period=1	Period=2	Period=3	Period=4	Period=5
2	0.662	0.590	0.588	0.590	0.612
4	0.674	0.634	0.619	0.606	0.758
6	0.756	0.647	0.671	0.671	0.696
8	0.754	0.679	0.746	0.713	0.479
10	0.668	0.655	0.730	0.644	0.642
12	0.460	0.764	0.572	0.485	0.486
14	0.611	0.346	0.434	0.322	0.305
16	0.640	0.114	0.186	0.334	0.263

Table A2.2: ARI Score for Different n_basis and $Period$

n_basis	Period=1	Period=2	Period=3	Period=4	Period=5
2	0.686	0.637	0.632	0.640	0.643
4	0.784	0.739	0.699	0.693	0.826
6	0.842	0.733	0.758	0.758	0.797
8	0.894	0.774	0.837	0.818	0.514
10	0.765	0.766	0.837	0.752	0.744
12	0.472	0.886	0.637	0.522	0.528
14	0.691	0.261	0.427	0.286	0.302
16	0.752	0.016	0.091	0.330	0.235

First, we can notice that the GCV Score is not necessarily the best in the case of the best clustering effect. It can be observed that the absolute value of the GCV value changes

Table A2.3: GCV Score for Different n_basis and Period

n_basis	Period 1	Period 2	Period 3	Period 4	Period 5
2	-10.670	-10.777	-10.768	-10.763	-10.761
4	-14.079	-14.078	-13.893	-13.824	-13.792
6	-18.971	-17.722	-17.150	-16.964	-16.882
8	-26.244	-22.266	-21.582	-21.461	-21.427
10	-37.672	-31.172	-34.471	-36.607	-37.806
12	-56.354	-65.889	-111.304	-137.857	-152.666
14	-88.834	-384.007	-1130.297	-1630.242	-1927.460
16	-150.563	-5562.483	-21310.274	-33192.428	-47922.672

greatly, while the changes of AMI and ARI are relatively small. Specifically, GCV represents a measure of the complexity or adaptability of the model. A large negative value means that the model may be overfitting or underfitting. This also inverts the poor results to a certain extent. For example, in the group of results with $n_basis = 16$, it can be observed that their clustering effect is generally worse than most other results, which is due to their poor smoothing effect.

In addition, it is noted that the best clustering result is achieved with $n_basis = 12$ and $Period = 2$, yielding AMI and ARI scores of 0.764 and 0.886, respectively. The parameters ultimately selected in Section 4, $n_basis = 10$ and $Period = 3$, also produced relatively good results, only slightly worse than the optimal case. It is important to note that, since GMM is used in the final step of the algorithm, the results may vary slightly due to the random initialization in GMM.

Picture A2.1, A2.2 and A2.3 present the results for different values of n_basis and $Period$, showing the corresponding AMI, ARI, and GCV scores. Where n_basis is used as the horizontal axis, and $Period$ is represented by color.

From these three figures, we can observe that the AMI and ARI scores do not change drastically. When n_basis is less than 10, the scores generally range between 0.6 and 0.8, with only a few results deviating from this range.

Figure A2.4 illustrates the relationship between the GCV score and the AMI and ARI scores. The horizontal axis represents the GCV score, while the vertical axis shows the corresponding AMI and ARI scores.

It is noticeable that when the GCV score is very small, it indicates poor smoothing results, often leading to poor clustering performance. While a GCV score close to zero does not necessarily guarantee better clustering results, it generally increases the likelihood of achieving higher AMI and ARI scores.

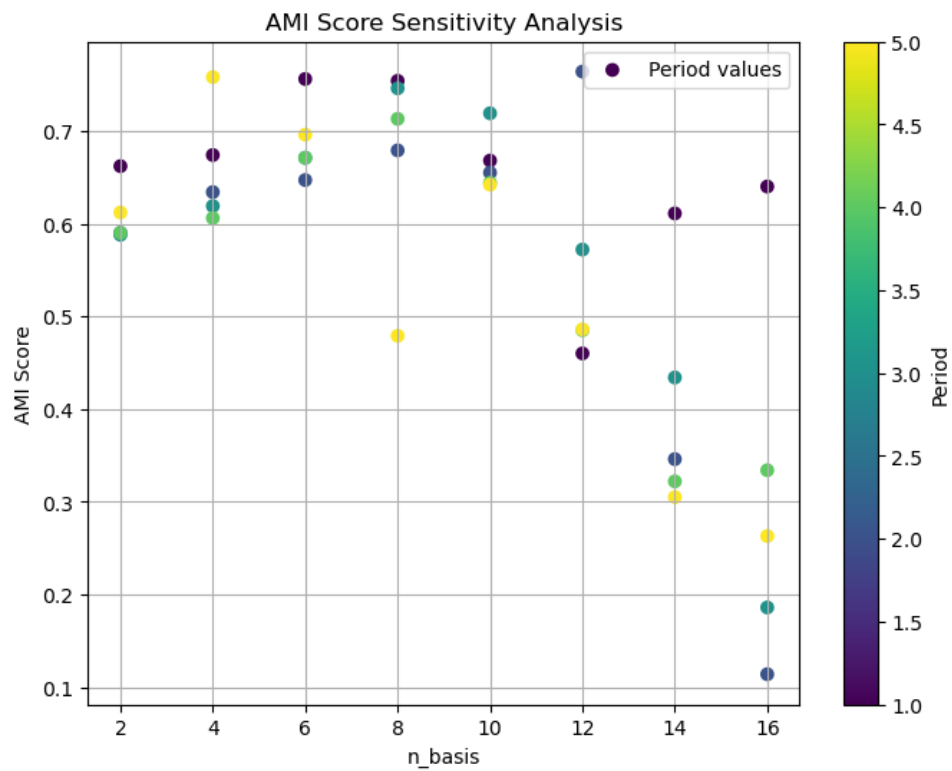


Figure A2.1: AMI Score against Varied Smoother Parameter

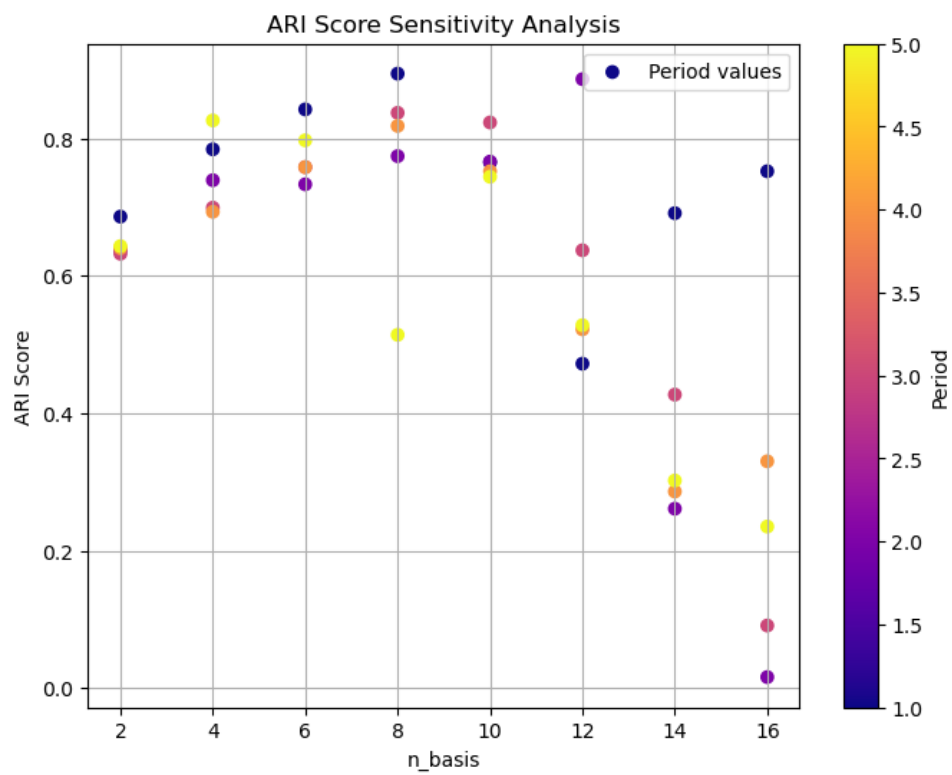


Figure A2.2: ARI Score against Varied Smoother Parameter

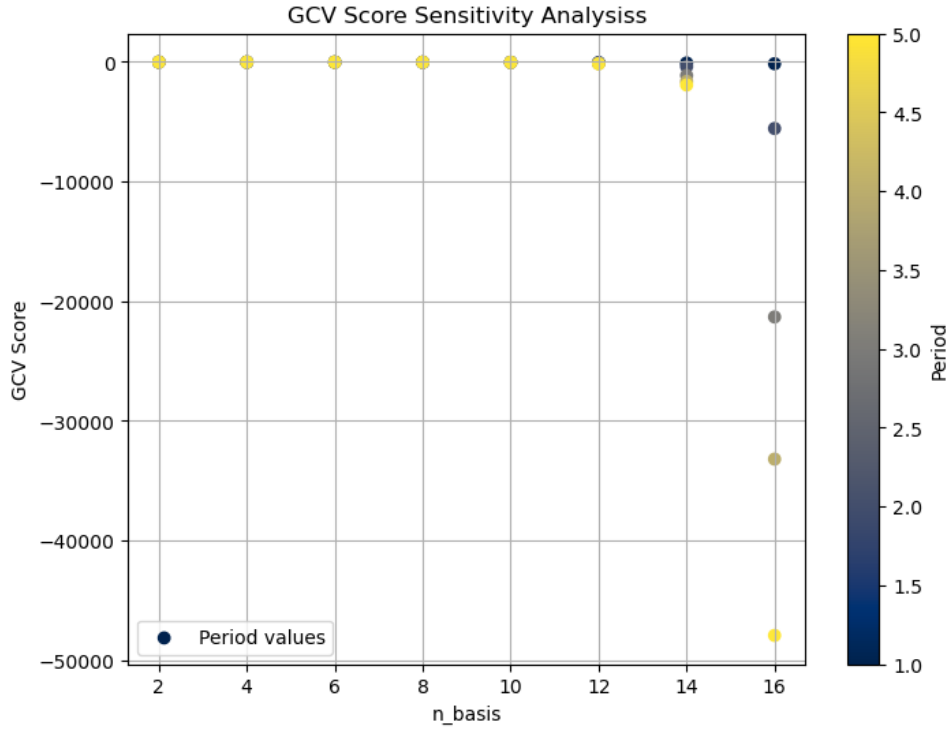


Figure A2.3: GCV Score against Varied Smoother Parameter

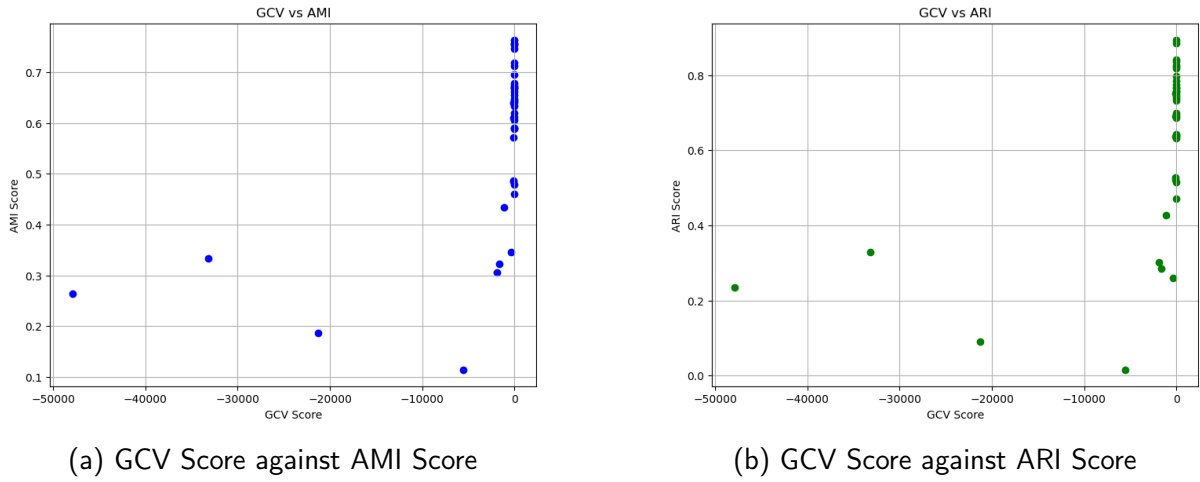


Figure A2.4: GCV Score against AMI and ARI Scores

A2.3 Conclusion

Based on the experimental results from the previous section, we can observe that once the smoother is selected, tuning its hyperparameters can affect the final clustering results, but the impact is not very significant if the smoothing effect remains consistent. For instance, when $n_{basis} = 2$, the GCV score ranges from -10.777 to -10.670, with AMI values between 0.588 and 0.662, showing minimal variation and remaining relatively high. The ARI score ranges from 0.632 to 0.686, indicating stable clustering performance. Overall, the fluctuation in the results is minor. However, when $n_{basis} = 16$, the GCV score increases

significantly in absolute value, reaching as high as -47922.672. In this case, both AMI and ARI drop drastically, with the worst-case values of 0.114 and 0.016, respectively, which are close to random clustering results.

The sensitivity analysis shows that while tuning the smoother's hyperparameters can influence the clustering results, the impact is generally small as long as the smoother is reasonably selected. However, when the hyperparameters are set to extreme values, the clustering performance can degrade significantly, with results approaching random clustering. Overall, within a normal range of parameter settings, the effect on clustering performance remains relatively minor.