Pattern Recognition 134 (2023) 109129

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Dirichlet process mixture of Gaussian process functional regressions and its variational EM algorithm

Tao Li, Jinwen Ma*

Department of Information and Computational Sciences, School of Mathematical Sciences and LMAM, Peking University, Beijing 100871, China

ARTICLE INFO

Article history:
Received 20 November 2021
Revised 18 August 2022
Accepted 22 October 2022
Available online 26 October 2022

Keywords:
Gaussian process
Dirichlet process
Functional data
Non-parametric Bayesian model
EM algorithm
Variational inference
Multi-modal data

ABSTRACT

Gaussian Process Functional Regression (GPFR) is a powerful tool in functional data analysis. In practical applications, functional data may be generated from different signal sources, and a single GPFR is not flexible enough to accurately model the data. To tackle the heterogeneity problem, a finite mixture of Gaussian Process Functional Regressions (mix-GPFR) was suggested. However, the number of components in mix-GPFR needs to be specified a priori, which is difficult to determine in practice. In this paper, we propose a Dirichlet Process Mixture of Gaussian Process Functional Regressions (DPM-GPFR), in which there are potentially infinite many GPFR components dominated by a Dirichlet process. Thus, DPM-GPFR is far more flexible than a single GPFR, and sidestep the model selection problem in mix-GPFR. We further develop a fully Bayesian treatment for learning DPM-GPFR based on the Variational Expectation-Maximization (VEM) algorithm. Experimental results on both synthetic datasets and real-world datasets demonstrate the effectiveness of our proposed method.

1. Introduction

Gaussian process (GP) [1] is the dominant non-parametric Bayesian model to learn and infer over temporal data or uncertain functions, which has been widely used in many fields such as machine learning, pattern recognition, and data mining. It consists of two basic elements, mean function and covariance function. One attractive property of Gaussian processes is that there are a variety of covariance functions for us to choose from, which enable us to model the function with different smoothness and structures. On the other hand, the mean function is often assumed to be a linear function of input variables, or even zero for convenience.

Usually, Gaussian processes are applied to regression and classification tasks [2,3], but they can also be applied to functional data (or batch data) problems [4–6] such as curve prediction, curve clustering, *etc*.. To see the difference between curve prediction and ordinary prediction, we consider the electricity load prediction problem, which is an intuitive and important example throughout this paper. Suppose that we have an electricity load dataset, which consists of 100 curves corresponding to 100 days, respectively, and each curve contains 96 (24×4) samples recorded every quarter-hour during one day. Suppose that we have recorded 48 electricity loads of today from 00:00 to 11:15, and we wish to predict the future trend of today's electricity load until 23:59. In ordinary prediction, we only use these 48 samples to predict. However, it is reasonable to assume that electricity loads of different days have a strong correlation and exhibit certain trends, thus the 100 curves of historical electricity load records may also be helpful to predict today's future electricity loads. Therefore, a better choice is to use the information from other curves to improve the prediction performance, which corresponds to curve prediction. Besides prediction, we hope to cluster the electricity load curves into groups to discover certain patterns, and this is referred to as the curve clustering problem. Although we can perform classical clustering algorithms such as k-means and hierarchical clustering to the 96-dimensional vectors, the time-dependent properties may be destroyed since most clustering algorithms assume the features are exchangeable. Furthermore, if there are missing values in these curves or the loads are recorded at different times each day, the classical clustering methods are invalid in this circumstance.

One seminal work of applying Gaussian processes to batch data is Gaussian Process Functional Regression (GPFR) [4]. In GPFR, the mean function is represented by a linear function of exogenous covariates, and the covariance structure is modeled by a Gaussian process. When there are no exogenous covariates and the inputs have temporal relationships, GPFR is equivalent to model the curves with a single Gaussian process whose mean function is estimated by a linear combination of spline basis functions [7–9]. Compared with the Gaussian process model, GPFR enables us to learn the non-parametric global trend. Compared with other func-

* Corresponding author.
E-mail address: jwma@math.pku.edu.cn (J. Ma).

https://doi.org/10.1016/j.patcog.2022.109129
0031-3203/

tional regression methods [10,11], GPFR has a more flexible covariance structure. However, GPFR fails to model heterogeneous/multi-modal data accurately [5]. In real applications, the curves may be generated from different signal sources, thus a single GPFR is not flexible enough to model all the curves, especially when there are a variety of evolving trends. Back to the electricity load prediction example, the curves corresponding to winter and summer are very likely to have significantly different trends and shapes. Just as in ordinary prediction and clustering tasks, the mixture model is a powerful technique to deal with heterogeneity and multi-modality [12]. Shi and Wang [5] proposed a mixture of Gaussian Process Functional Regressions (mix-GPFR). Moreover, Wu and Ma [9] further extended it to a Two-layer Mixture of Gaussian Process Functional Regressions (TMGPFR) for modeling general stochastic processes from different sources.

However, mix-GPFR requires the user to specify the number of components in advance, which is difficult to determine in practical applications. An underestimated number of components may result in under-fitting, while an overestimated number of components leads to over-fitting, and the performances are usually frustrating in both circumstances. One can run the algorithms with different numbers of components and use cross-validation to choose the best one, but this procedure is time-consuming, and the obtained result is generally unstable. From the non-parametric Bayesian perspective, it is possible to infer the necessary number of components automatically if the prior distribution over mixing proportions is suitable. One of the most prominent distributions in Bayesian mixture modeling is the Dirichlet process [13]. The Dirichlet process mixture model can be regarded as an extension of the finite mixture model, which allows potentially infinite many components [14,15] to model the data, and the number of components we need to interpret the observational data is determined automatically. In this work, we propose a mixture model of Gaussian process functional regressions based on the Dirichlet process, and we refer to this model as DPM-GPFR. We further design an effective learning algorithm for the model based on variational inference [16] and the EM algorithm [17].

The rest of this paper is organized as follows. We introduce notations and background in Section 2, including GP, GPFR, Dirichlet process, and Variational EM algorithm. We propose the model and derive the variational EM algorithm in Section 3. To validate the effectiveness of DPM-GPFR, we compare it with other competing methods on both synthetic datasets and real-world datasets in Section 4. Finally, we draw a brief conclusion and point out several possible directions for future research in Section 5.

2. Preliminaries

2.1. Gaussian process

In Gaussian process regression, one assumes a priori that the underlying function is generated by a Gaussian process. Equivalently, this assumption indicates any finite-dimensional distribution of samples is Gaussian. Formally, suppose that we have a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ and let $\mathbf{x} = [x_1, x_2, \ldots, x_N]^\intercal, \mathbf{y} = [y_1, y_2, \ldots, y_N]^\intercal$, then we assume $\mathbf{y}|\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ where $\boldsymbol{\mu} = [\mu(x_1), \mu(x_2), \ldots, \mu(x_N)]^\intercal$ and $\mathbf{C}_{ij} = c(x_i, x_j; \boldsymbol{\theta})$ with covariance function $c(\cdot, \cdot; \boldsymbol{\theta})$. From the function perspective [1], we assume that the function relating x_i and y_i is $y(x)$, *i.e.*, $y(x_i) = y_i$, and it is a sample path of a Gaussian process,

$$y(x) \sim \mathcal{GP}(\mu(x), c(x, x'; \boldsymbol{\theta})).$$

Here, we use the squared exponential covariance function which has been widely used [9,18], which is defined as

$$c(x_i, x_j|\boldsymbol{\theta}) = \theta_1^2 \exp\left(-\theta_2^2 \frac{(x_i - x_j)^2}{2}\right) + \theta_3^2 \delta_{ij},$$

where δ_{ij} is the Kronecker delta function and $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]$. The mean function is often assumed to be zero or a linear function dependent on x.

As for prediction, given a new input x_* and we want to predict the corresponding output $y*$, from the definition of Gaussian process we immediately know that $y_*|\mathcal{D}, x_*$ also obeys a Gaussian distribution [1] and

$$\begin{aligned} \mathbb{E}[y_*|\mathcal{D}, x_*] &= \mu(x_*) + \mathbf{c}(x_*, \mathbf{x})\mathbf{C}^{-1}(\mathbf{y} - \boldsymbol{\mu}), \\ \mathrm{Var}(y_*|\mathcal{D}, x_*) &= c(x_*, x_*) - \mathbf{c}(x_*, \mathbf{x})\mathbf{C}^{-1}\mathbf{c}(\mathbf{x}, x_*). \end{aligned} \tag{1}$$

Gaussian process model can also be applied to analyze functional data. Suppose that we have functional dataset $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^N$, which consists of N sub-datasets $\mathcal{D}_i = \{(x_{ij}, y_{ij})\}_{j=1}^{N_i}$, and each $\mathcal{D}_i$ can be regarded as a curve, or a finite collection of (input,output) pairs of certain function. Let the function underlying the i-th curve be $y_i(x)$, *i.e.*, $y_i(x_{ij}) = y_{ij}$, then Gaussian process for this functional dataset assumes

$$y_i(x) \sim \mathcal{GP}(\mu(x), c(x, x'; \boldsymbol{\theta})),\ i = 1, 2, \ldots, N.$$

That is to say, we assume these curves are generated by a common Gaussian process.

2.2. Gaussian process functional regression

Shi et al. [4] suggested to use a linear combination of B-spline basis functions to estimate the mean function. The corresponding model is called Gaussian Process Functional Regression (GPFR), which assumes

$$y(x) = \mu(x) + \tau(x), \tau(x) \sim \mathcal{GP}(0, c(x, x'; \boldsymbol{\theta})),$$

where $\mu(x)$ is the mean function estimated with B-spline functions. Theoretically, the models developed in this paper and other related literatures such as [5–7,9] do not rely on a set of specific B-spline basis functions and can be extended to a set of other basis functions straightforwardly, and we choose to use the B-spline basis functions here because it is effective for modeling nonlinear function and can be easily extended to high-dimensional cases. Suppose that we have D B-spline basis functions $\{\phi_d(x)\}_{d=1}^D$. Let $\mu(x) = \sum_{d=1}^D b_d \phi_d(x)$ and $\boldsymbol{\Phi}$ be an $N \times D$ matrix with $\boldsymbol{\Phi}_{id} = \phi_d(x_i), \mathbf{b} = [b_1, b_2, \ldots, b_D]^\intercal$, GPFR can be equivalently written as $\mathbf{y}|\mathbf{x} \sim \mathcal{N}(\boldsymbol{\Phi}\mathbf{b}, \mathbf{C})$. From the function perspective, this model can be denoted as

$$y(x) \sim \mathcal{GPFR}(x; \mathbf{b}, \boldsymbol{\theta}).$$

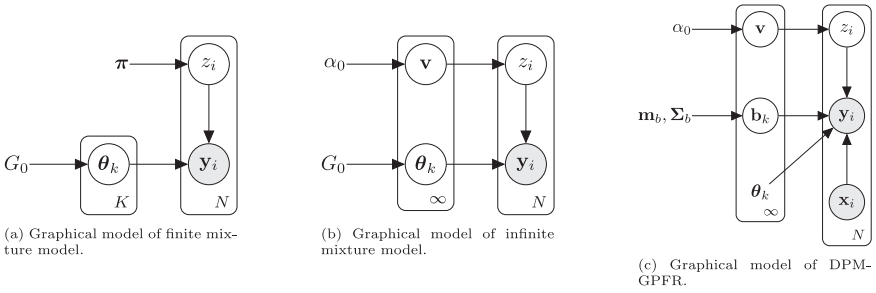
The prediction strategy is similar to the Gaussian process model. One distinguishing feature of GPFR is that it can deal with functional data more effectively by capturing their common trend. When applying the GPFR model to a common static regression problem or a conventional time-series regression problem, there is only one sample curve, thus introducing a spline function may lead to over-fitting or identification problems. On batch datasets, multiple realizations of the underlying stochastic process exhibit the common trend, which can be modeled by the spline function. By applying the GPFR model to batch data, we assume

$$y_i(x) \sim \mathcal{GPFR}(x; \mathbf{b}, \boldsymbol{\theta}), i = 1, 2, \ldots, N,$$

Furthermore, Shi and Wang [5] extended GPFR to the mixture model of GPFR (mix-GPFR). In mix-GPFR, each curve $\mathcal{D}_i$ is assumed to be generated from one of K GPFRs, given the latent indicator variable z_i,

$$y_i(x)|z_i = k \sim \mathcal{GPFR}(x; \mathbf{b}_k, \boldsymbol{\theta}_k), i = 1, 2, \ldots, N.$$

Employing neural networks to express GP mean function also enhances the flexibility of GPs. However, compared with spline functions, neural networks are far more difficult and time-consuming



(a) Graphical model of finite mixture model.

(b) Graphical model of infinite mixture model.

(c) Graphical model of DPM-GPFR.

Fig. 1. Graphical models of finite mixture model, infinite mixture model, and DPM-GPFR.

to train. On the other hand, the spline function naturally encodes prior information about the smoothness of functions, which attains a good balance between complexity and flexibility.

2.3. Dirichlet process based mixture model

Suppose that we have N data-points $\{\mathbf{y}_i\}_{i=1}^N$, and consider a finite mixture model with K components (see Fig. 1(a)) as follows:

$$\begin{aligned}
&\boldsymbol{\theta}_k \sim G_0, k = 1, 2, \ldots, K\\
&z_i|\boldsymbol{\pi} \sim \text{Categorical}(\boldsymbol{\pi}), i = 1, 2, \ldots, N\\
&\mathbf{y}_i|z_i, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K \sim F(\cdot|\boldsymbol{\theta}_{z_i}), i = 1, 2, \ldots, N.
\end{aligned}$$

Here, G_0 is the prior distribution for $\boldsymbol{\theta}$, $F(\cdot|\boldsymbol{\theta})$ is the distribution generating data parametrized by $\boldsymbol{\theta}$, and $\boldsymbol{\pi}$ denotes the distribution of mixing proportions. Since there are K components, $\boldsymbol{\pi}$ is a K-vector satisfying $\sum_{k=1}^K \pi_k = 1, \boldsymbol{\pi}_k \geq 0$. It is unnecessary to constrain K to be finite as long as the mixing proportions $\boldsymbol{\pi}$ are well-defined. In the stick-breaking construction, given the scaling parameter α_0, we draw an infinite collection of random variables, $\nu_k \sim \text{Beta}(1, \alpha_0)$ and define $\pi_k(\mathbf{v}) = \nu_k \prod_{l=1}^{k-1}(1-\nu_l)$, then it is easy to verify $\sum_{k=1}^K \pi_k(\mathbf{v}) = 1 - \prod_{l=1}^K (1-\nu_l)$ by induction. Taking $K \to \infty$ we know $\sum_{k=1}^\infty \pi_k(\mathbf{v}) = 1$ since $\prod_{l=1}^K (1-\nu_l)$ is a production of K variables in (0,1). Thus, we may consider the infinite mixture model as follows:

$$\nu_k|\alpha_0 \sim \text{Beta}(1, \alpha_0), k = 1, 2, \ldots, \infty \tag{2}$$

$$\boldsymbol{\theta}_k \sim G_0, k = 1, 2, \ldots, \infty \tag{3}$$

$$z_i|\boldsymbol{\pi} \sim \text{Categorical}(\boldsymbol{\pi}(\mathbf{v})), i = 1, 2, \ldots, N \tag{4}$$

$$\mathbf{y}_i|z_i, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_\infty \sim F(\cdot|\boldsymbol{\theta}_{z_i}), i = 1, 2, \ldots, N. \tag{5}$$

The corresponding graphical model is shown in Fig. 1(b). The infinite mixture model is very similar to the finite mixture model, except that there are countably infinite $\{\boldsymbol{\theta}_k\}_{k=1}^\infty$ and the mixing proportions $\boldsymbol{\pi}$ are defined with the help of Beta random variables $\{\nu_k\}_{k=1}^\infty$.

2.4. Variational EM algorithm

Suppose that $\mathbf{y}$ is the observed data, $\mathbf{z}$ is the latent variable and $\boldsymbol{\Theta}$ is the hyper-parameter. The goal of parameter learning is to find out the maximum likelihood estimate of $\boldsymbol{\Theta}$ and obtain the posterior distribution of $\mathbf{z}$. However, the calculation of marginal likelihood $p(\mathbf{y}; \boldsymbol{\Theta})$ is usually intractable. Using the fact that $p(\mathbf{y}; \boldsymbol{\Theta}) = p(\mathbf{y}, \mathbf{z}; \boldsymbol{\Theta})/p(\mathbf{z}|\mathbf{y}; \boldsymbol{\Theta})$, we can decompose the marginal likelihood with respect to arbitrary distribution $q(\mathbf{z})$ as

$$\begin{aligned}
\log p(\mathbf{y}; \boldsymbol{\Theta}) &= \int q(\mathbf{z}) \log \frac{p(\mathbf{y}, \mathbf{z}; \boldsymbol{\Theta})}{q(\mathbf{z})} \mathrm{d}\mathbf{z} - \int q(\mathbf{z}) \log \frac{p(\mathbf{z}|\mathbf{y}; \boldsymbol{\Theta})}{q(\mathbf{z})} \mathrm{d}\mathbf{z}\\
&= \mathcal{L}(q(\mathbf{z}); \boldsymbol{\Theta}) + \mathcal{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{y}; \boldsymbol{\Theta})).
\end{aligned}$$

The first term $\mathcal{L}(q(\mathbf{z}); \boldsymbol{\Theta})$ is called evidence lower bound (ELBO), which is a lower bound of $\log p(\mathbf{y}; \boldsymbol{\Theta})$ since $\mathcal{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{y}; \boldsymbol{\Theta})) \geq 0$. Following the idea of Minorization-Maximization, we can maximize $\mathcal{L}(q(\mathbf{z}); \boldsymbol{\Theta})$ instead of maximizing $\log p(\mathbf{y}; \boldsymbol{\Theta})$. To achieve this, we maximize $\mathcal{L}(q(\mathbf{z}); \boldsymbol{\Theta})$ with respect to $q(\mathbf{z})$ and $\boldsymbol{\Theta}$ alternately. When $\boldsymbol{\Theta}$ is fixed, because $\log p(\mathbf{y}; \boldsymbol{\Theta})$ is a constant, maximizing $\mathcal{L}(q(\mathbf{z}); \boldsymbol{\Theta})$ boils down to minimizing $\mathcal{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{y}; \boldsymbol{\Theta}))$. We usually make structural assumptions on $q(\mathbf{z})$ to simplify the problem. The most popular assumption is the mean-field family, which partitions $\mathbf{z}$ to groups $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_M\}$ and restricts $q(\mathbf{z})$ factorizes with respect to groups, $q(\mathbf{z}) = \prod_{j=1}^M q_j(\mathbf{z}_j)$. Let $\mathbf{z}_{-j} = \mathbf{z} - \{\mathbf{z}_j\}$, $q_{-j}(\mathbf{z}_{-j}) = q(\mathbf{z})/q_j(\mathbf{z}_j)$ and $\mathbb{E}_j = \mathbb{E}_{q_j(\mathbf{z}_j)}, \mathbb{E}_{-j} = \mathbb{E}_{q_{-j}(\mathbf{z}_{-j})}$ then given $q_{-j}(\mathbf{z}_{-j})$, the maximum of ELBO is attained at

$$q_j(\mathbf{z}_j) \propto \exp\left(\mathbb{E}_{-j}\big[\log p(\mathbf{y}, \mathbf{z}_j, \mathbf{z}_{-j})\big]\right). \tag{6}$$

We iteratively update $q_j(\mathbf{z}_j)$ until convergence. Then we fix $q(\mathbf{z})$, integrate out $q(z)$ in $\mathcal{L}(q(\mathbf{z}); \boldsymbol{\Theta})$ to obtain the Q-function and maximize it with respect to $\boldsymbol{\Theta}$. Compared with the original EM algorithm, the only difference is that we approximate the posterior distribution of latent variables via variational inference [19] rather than use the ground-truth posterior.

3. Infinite mixture of Gaussian process functional regressions

3.1. Model design

Suppose that we have batch data $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^N$, $\mathcal{D}_i = \{(x_{im}, y_{im})\}_{m=1}^{N_i}$, and let $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{iN_i}]^\intercal, \mathbf{y}_i = [y_{i1}, y_{i2}, \ldots, y_{iN_i}]^\intercal$. We suppose the function underlying ith curve is $y_i(x)$, *i.e.*, $y_i(x_{im}) = y_{im}, \forall m = 1, \ldots, N_i$. We utilize an infinite mixture of GPFRs to model the batch data. The corresponding probabilistic graphical model is shown in Fig. 1(c), and we refer this model as DPM-GPFR (Dirichlet Process Mixture of Gaussian Process Functional Regression). In this model, $\{\nu_k\}_{k=1}^\infty$ are stick-breaking variables and $\nu_k \sim \text{Beta}(1, \alpha_0)$ where α_0 is the scaling parameter. Latent variable z_i is the indicator variable indicating which component the i-th curve belongs to. In stick-breaking representation of Dirichlet process, the prior of z_i is a categorical distribution with

$$p(z_i|\mathbf{v}) = \nu_{z_i} \prod_{k=1}^{z_i-1}(1-\nu_k) = \prod_{k=1}^\infty \nu_k^{\mathbb{I}(z_i=k)}(1-\nu_k)^{\mathbb{I}(z_i>k)}.$$

Given indicator z_i, the ith curve is assumed to be generated from the z_i-th component, which is a GPFR model with parameters $\mathbf{b}_{c_i}$

and $\boldsymbol{\theta}_{z_i}$:

$$y_i(x)|\{\mathbf{b}_k\}_{k=1}^{\infty}, \{\boldsymbol{\theta}_k\}_{k=1}^{\infty}, z_i \sim \mathcal{GPFR}(\mathbf{x}_i; \mathbf{b}_{z_i}, \boldsymbol{\theta}_{z_i}).$$

We further impose a multi-variate Gaussian prior on the B-spline coefficients $\mathbf{b}_k$, *i.e.*, $\mathbf{b}_k \sim \mathcal{N}(\mathbf{m}_b, \boldsymbol{\Sigma}_b)$ where $\mathbf{m}_b$ and $\boldsymbol{\Sigma}_b$ are regarded as parameters. Learning $\mathbf{b}_k$ is similar to learning the coefficients in the least square regression problem. Since we expand x to a D dimensional vector $[\phi_1(x), \ldots, \phi_D(x)]^T$, the model tends to overfit as D becomes larger. From the frequentist perspective, a common way to prevent overfitting is to penalize the coefficients $\mathbf{b}_k$ via ℓ_2 or ℓ_1 norms. From the Bayesian perspective, penalizing $\|\mathbf{b}_k\|_2$ is equivalent to place a Gaussian prior $\mathcal{N}(0, \sigma^2\mathbf{I}_D)$ over $\mathbf{b}_k$, and penalizing $\|\mathbf{b}_k\|_1$ is equivalent to place a Laplacian prior over $\mathbf{b}_k$. In this paper, the prior $\mathcal{N}(\mathbf{m}_b, \boldsymbol{\Sigma}_b)$ over $\mathbf{b}_k$ can be regarded as a generalized version of ℓ_2 norm regularization. Besides, the parameters the prior $\mathbf{m}_b, \boldsymbol{\Sigma}_b$ are also learned during the learning process in an empirical Bayes way [20]. The main reason we use a Gaussian prior on B-spline coefficients is that this is the conjugate prior, which means the posterior of $\mathbf{b}_k$ is also Gaussian and we can update the parameters of posterior distribution effectively. In addition, ℓ_2 regularization is so common in machine learning that we believe this is a proper choice for the model. Theoretically, we can place arbitrary prior distribution over $\mathbf{b}_k$ (corresponding to different regularization terms from the frequentist perspective), however, the other choices do not have the conjugate property, and the inference over $\mathbf{b}_k$ would be much more challenging. In Gaussian processes, parameters of covariance functions are usually viewed as hyper-parameters rather than latent variables and they are estimated by Type-II maximum likelihood estimation [1,6]. Thus, we assume there are infinite many $\boldsymbol{\theta}_k = [\theta_{k,1}, \theta_{k,2}, \theta_{k,3}]^\intercal$ and update them in M-step of variational EM algorithm.

The mixture of hierarchical Gaussian processes (MOHGP) [21] also places a Dirichlet process prior over a mixture of Gaussian processes. In MOHGP, the mean function of a Gaussian process component is again a Gaussian process. This model was developed to cluster structured time series. Since GPs usually perform poorly on extrapolation, MOHGP is not suitable for prediction. In DPM-GPFR, the mean functions are B-spline functions, thus it can forecast future trends given a new curve. There are two main differences between mix-GPFR and DPM-GPFR. First, the priors over mixture proportions are different. Second, mix-GPFR treats $\mathbf{b}_k$ as parameters, while in DPM-GPFR we view $\mathbf{b}_k$ as latent variables as in Fig. 1c. When D is large, there are too many parameters in mix-GPFR and it tends to over-fit. On the other hand, DPM-GPFR alleviates this problem by placing a prior over $\mathbf{b}_k$. Recently, deep GPs [22] have been an active topic in the Gaussian process community. Compared with DPM-GPFR, deep GPs model multi-modality in the *input space*, while the proposed method models multi-modality in the *output space*.

3.2. Posterior inference and parameter learning

We employ the variational EM algorithm for learning. In the E-step, we need to calculate the approximate posterior of latent variables $\boldsymbol{\Omega} = \{z_i\}_{i=1}^N \cup \{\nu_k\}_{k=1}^{\infty} \cup \{\mathbf{b}_k\}_{k=1}^{\infty}$ using variational inference. The complete log-likelihood is

$$\begin{aligned}\log p(\mathbf{Y}, \boldsymbol{\Omega}|\mathbf{X}; \boldsymbol{\Theta}) = &\sum_{k=1}^{\infty} \log p(\nu_k; \alpha_0) + \sum_{i=1}^{N} \log p(z_i|\mathbf{v}) \\ &+ \sum_{k=1}^{\infty} \log p(\mathbf{b}_k; \mathbf{m}_b, \boldsymbol{\Sigma}_b) \\ &+ \sum_{i=1}^{N} \log p(\mathbf{y}_i|\mathbf{x}_i, z_i, \{\mathbf{b}_k\}_{k=1}^{\infty}; \{\boldsymbol{\theta}_k\}_{k=1}^{\infty}). \end{aligned} \tag{7}$$

The first three terms are relatively easy:

$$\begin{aligned}\log p(\nu_k; \alpha_0) &= (\alpha_0 - 1)\log(1-\nu_k) + \texttt{const}, \\ \log p(z_i|\mathbf{v}) &= \sum_{k=1}^{\infty} [\mathbb{I}(z_i > k)\log(1-\nu_k) + \mathbb{I}(z_i = k)\log\nu_k], \\ \log p(\mathbf{b_k}; \mathbf{m_b}, \boldsymbol{\Sigma_b}) &= -\frac{1}{2}(\mathbf{b_k} - \mathbf{m_b})^\intercal \boldsymbol{\Sigma_b}^{-1}(\mathbf{b_k} - \mathbf{m_b}) - \frac{\mathbf{1}}{2}\log|\boldsymbol{\Sigma_b}| \\ &\quad + \texttt{const}.\end{aligned}$$

As for the last term, let $\mathbf{C}_{i,k}$ denotes the covariance matrix of the i-th curve using parameters $\boldsymbol{\theta}_k$ and $\boldsymbol{\Phi}_i$ denotes the $N_i \times D$ B-spline basis matrix with $(\boldsymbol{\Phi}_i)_{ld} = \phi_d(x_{il})$, then

$$\begin{aligned}&\log p(\mathbf{y}_i|\mathbf{x}_i, z_i, \{\mathbf{b}_k\}_{k=1}^{\infty}; \{\boldsymbol{\theta}_k\}_{k=1}^{\infty}) \\ &= \sum_{k=1}^{\infty} \mathbb{I}(z_i = k)\Big[-\frac{N_i}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{C}_{i,k}| \\ &-\frac{1}{2}(\mathbf{y}_i - \boldsymbol{\Phi}_i\mathbf{b}_k)^\intercal \mathbf{C}_{i,k}^{-1}(\mathbf{y}_i - \boldsymbol{\Phi}_i\mathbf{b}_k)\Big].\end{aligned}$$

The goal is to find out an approximate posterior distribution $q(\boldsymbol{\Omega})$. We restrict $q(\boldsymbol{\Omega})$ in the mean-field family, and truncate the stick-breaking representations at level `max_K`, as in [13,23]. Equivalently, we fix a level `max_K` and let $\nu_{\texttt{max\_K}} = 1$. This implies that mixing proportions π_k are 0 when $k >$ `max_K`, and there are at most `max_K` components. With this assumption, the approximate posterior is

$$q(\boldsymbol{\Omega}) = \prod_{k=1}^{\texttt{max\_K}-1} q(\nu_k) \prod_{k=1}^{\texttt{max\_K}} q(\mathbf{b}_k) \prod_{i=1}^{N} q(z_i).$$

Furthermore, due to the property of exponential families [16,24,25], we assume $q(\nu_k)$ is a Beta distribution $q(\nu_k; \alpha_k, \beta_k)$, $q(\mathbf{b}_k)$ is a multivariate normal distribution $q(\mathbf{b}_k; \mathbf{m}_k, \boldsymbol{\Sigma}_k)$ and $q(z_i)$ is a categorical distribution $q(z_i; \varphi_{i,1}, \ldots, \varphi_{i,K})$. With these free variational parameters, we apply Eq. (6) to update the variational distribution. We first introduce some useful formulas:

$$\begin{aligned}&\mathbb{E}_{q(\nu_k;\alpha_k,\beta_k)}[\log\nu_k] = \Psi(\alpha_k) - \Psi(\alpha_k + \beta_k), \\ &\mathbb{E}_{q(\nu_k;\alpha_k,\beta_k)}[\log(1-\nu_k)] = \Psi(\beta_k) - \Psi(\alpha_k + \beta_k), \\ &\mathbb{E}_{q(\mathbf{b}_k;\mathbf{m}_k,\boldsymbol{\Sigma}_k)}[\mathbf{b}_k^\intercal \mathbf{t}] = \mathbf{m}_k^\intercal \mathbf{t}, \quad \forall \mathbf{t} \in \mathbb{R}^D, \\ &\mathbb{E}_{q(\mathbf{b}_k;\mathbf{m}_k,\boldsymbol{\Sigma}_k)}[\mathbf{b}_k^\intercal \mathbf{Q}\mathbf{b}_k] = \mathbf{m}_k^\intercal \mathbf{Q}\mathbf{m}_k + \mathrm{tr}(\mathbf{Q}\boldsymbol{\Sigma}_k)\ \forall \mathbf{Q} \in \mathbb{S}_{++}^D, \\ &\mathbb{E}_{q(z_i;\varphi_{i,1},\ldots,\varphi_{i,\texttt{max\_K}})}[\mathbb{I}(z_i = k)] = \varphi_{i,k},\end{aligned}$$

where Ψ is the digamma function, and $\mathbb{S}_{++}^D$ denotes the set of $D \times D$ positive definite matrices.

For ν_k, only first two terms in Eq. (7) are involved, and we only need to take expectation with respect to $\{z_i\}_{i=1}^N$.

$$\begin{aligned}&q(\nu_k; \alpha_k, \beta_k) \propto \exp\Big((\alpha_0 - 1)\log(1-\nu_k) \\ &\quad + \sum_{i=1}^{N} \mathbb{E}_{q(z_i;\varphi_{i,1},\ldots,\varphi_{i,K}))}[\mathbb{I}(z_i > k)\log(1-\nu_k) + \mathbb{I}(z_i = k)\log\nu_k]\Big) \\ &= \left(\alpha_0 + \sum_{i=1}^{N}\sum_{l=k+1}^{\texttt{max\_K}} \varphi_{i,j} - 1\right)\log(1-\nu_k) + \sum_{i=1}^{N} \varphi_{i,k}\log\nu_k.\end{aligned}$$

Therefore, we have

$$\alpha_k = 1 + \sum_{i=1}^{N} \varphi_{i,k}, \ \beta_k = \alpha_0 + \sum_{i=1}^{N}\sum_{l=k+1}^{\texttt{max\_K}} \varphi_{i,j}.$$

For $\mathbf{b}_k$, only last two terms in Eq. (7) are involved. Besides, the third term is fixed, and we only need to take expectation of the

fourth term with respect to $\{z_i\}_{i=1}^N$. Ignoring the terms not related to $\mathbf{b}_k$, we obtain

$$\begin{aligned}&\mathbb{E}_{q(z_i)}[\log p(\mathbf{y}_i|\mathbf{x}_i, z_i, \{\mathbf{b}_k\}_{k=1}^\infty; \{\boldsymbol{\theta}_k\}_{k=1}^\infty)]\\&= \sum_{k=1}^{\texttt{max\_K}} \varphi_{i,k}\Big(-\frac{1}{2}\mathbf{b}_k^\intercal \boldsymbol{\Phi}_i^\intercal \mathbf{C}_{i,k}^{-1}\boldsymbol{\Phi}_i\mathbf{b}_k + \mathbf{y}_i^\intercal \mathbf{C}_{i,k}^{-1}\boldsymbol{\Phi}_i\mathbf{b}_k\Big)\\&= -\frac{1}{2}\mathbf{b}_k^\intercal\big(\varphi_{i,k}\boldsymbol{\Phi}_i^\intercal\mathbf{C}_{i,k}^{-1}\boldsymbol{\Phi}_i\big)\mathbf{b}_k + \big(\varphi_{i,k}\boldsymbol{\Phi}_i^\intercal\mathbf{C}_{i,k}^{-1}\mathbf{y}_i\big)^\intercal\mathbf{b}_k + \texttt{const}.\end{aligned}$$

Summing up all N terms and the prior term, we have

$$\begin{aligned}&\log q(\mathbf{b}_k; \mathbf{m}_k, \boldsymbol{\Sigma}_k) = \log p(\mathbf{b}_k; \mathbf{m}_b, \boldsymbol{\Sigma}_b)\\&+\sum_{i=1}^N \mathbb{E}_{q(z_i)}[\log p(\mathbf{y}_i|\mathbf{x}_i, z_i, \{\mathbf{b}_k\}_{k=1}^\infty; \{\boldsymbol{\theta}_k\}_{k=1}^\infty)] + \texttt{const}\\&= -\frac{1}{2}\mathbf{b}_k^\intercal\left(\boldsymbol{\Sigma}_b^{-1} + \sum_{i=1}^N \varphi_{i,k}\boldsymbol{\Phi}_i^\intercal\mathbf{C}_{i,k}^{-1}\boldsymbol{\Phi}_i\right)\mathbf{b}_k\\&+\left(\boldsymbol{\Sigma}_b^{-1}\mathbf{m}_b + \sum_{i=1}^N \varphi_{i,k}\boldsymbol{\Phi}_i^\intercal\mathbf{C}_{i,k}^{-1}\mathbf{y}_i\right)^\intercal\mathbf{b}_k + \texttt{const}.\end{aligned}$$

By completing the square, we obtain the update formula for $q(\mathbf{b}_k; \mathbf{m}_k, \boldsymbol{\Sigma}_k)$:

$$\boldsymbol{\Sigma}_k = \left(\boldsymbol{\Sigma}_b^{-1} + \sum_{i=1}^N \varphi_{i,k}\boldsymbol{\Phi}_i^\intercal\mathbf{C}_{i,k}^{-1}\boldsymbol{\Phi}_i\right)^{-1},$$

$$\mathbf{m}_k = \boldsymbol{\Sigma}_k\left(\boldsymbol{\Sigma}_b^{-1}\mathbf{m}_b + \sum_{i=1}^N \varphi_{i,k}\boldsymbol{\Phi}_i^\intercal\mathbf{C}_{i,k}^{-1}\mathbf{y}_i\right).$$

For z_i, only the second and fourth terms in Eq. (7) are involved. We need to take expectation of $\log p(z_i|\mathbf{v})$ with respect to $\prod_{k=1}^{\texttt{max\_K}} q(v_k; \alpha_k, \beta_k)$ and $\log p(\mathbf{y}_i|\mathbf{x}_i, z_i, \{\mathbf{b}_k\}_{k=1}^\infty; \{\boldsymbol{\theta}_k\}_{k=1}^\infty)$ with respect to $\prod_{k=1}^{\texttt{max\_K}} q(\mathbf{b}_k; \mathbf{m}_k, \boldsymbol{\Sigma}_k)$.

$$\begin{aligned}&\mathbb{E}_{\prod_{k=1}^{\texttt{max\_K}} q(\mathbf{b}_k)}[\log p(\mathbf{y}_i|\mathbf{x}_i, z_i, \{\mathbf{b}_k\}_{k=1}^\infty; \{\boldsymbol{\theta}_k\}_{k=1}^\infty)]\\&= \sum_{k=1}^{\texttt{max\_K}} \mathbb{I}(z_i = k)\Big[-\frac{N_i}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{C}_{i,k}|\\&-\frac{1}{2}\mathbf{y}_i^\intercal\mathbf{C}_{i,k}^{-1}\mathbf{y}_i + \mathbb{E}_{q(\mathbf{b}_k)}[-\frac{1}{2}\mathbf{y}_i\boldsymbol{\Phi}_i^\intercal\mathbf{C}_{i,k}^{-1}\boldsymbol{\Phi}_i\mathbf{b}_k + (\boldsymbol{\Phi}_i^\intercal\mathbf{C}_{i,k}^{-1}\mathbf{y}_i)^\intercal\mathbf{b}_k]\Big]\\&= \sum_{k=1}^{\texttt{max\_K}} \mathbb{I}(z_i = k)\Big[-\frac{N_i}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{C}_{i,k}| - \frac{1}{2}\mathbf{y}_i^\intercal\mathbf{C}_{i,k}^{-1}\mathbf{y}_i\\&-\frac{1}{2}\mathbf{m}_k^\intercal\boldsymbol{\Phi}_i^\intercal\mathbf{C}_{i,k}^{-1}\boldsymbol{\Phi}_i\mathbf{m}_k - \frac{1}{2}\mathrm{tr}(\boldsymbol{\Phi}_i^\intercal\mathbf{C}_{i,k}^{-1}\boldsymbol{\Phi}_i\boldsymbol{\Sigma}_k) + (\boldsymbol{\Phi}_i^\intercal\mathbf{C}_{i,k}^{-1}\mathbf{y}_i)^\intercal\mathbf{m}_k\Big]\end{aligned}$$

With a little abuse of notation, we use $\mathcal{N}(\mathbf{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ to denote the probability density of $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ at $\mathbf{y}$. Taking exponential of the above equation, we immediately get

$$\begin{aligned}&\exp\mathbb{E}_{\prod_{k=1}^{\texttt{max\_K}} q(\mathbf{b}_k)}[\log p(\mathbf{y}_i|\mathbf{x}_i, z_i, \{\mathbf{b}_k\}_{k=1}^\infty; \{\boldsymbol{\theta}_k\}_{k=1}^\infty)]\\&\propto \prod_{k=1}^{\texttt{max\_K}}\left(\mathcal{N}(\mathbf{y}_i; \boldsymbol{\Phi}_i\mathbf{m}_k, \mathbf{C}_{i,k})\exp\left(-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_k\boldsymbol{\Phi}_i\mathbf{C}_{i,k}^{-1}\boldsymbol{\Phi}_i^\intercal)\right)\right)^{\mathbb{I}(z_i=k)}.\end{aligned}$$

The other expectation is

$$\begin{aligned}&\mathbb{E}_{\prod_{k=1}^{\texttt{max\_K}} q(v_k)}[\log p(z_i|\mathbf{v})]\\&= \sum_{k=1}^{\texttt{max\_K}} \mathbb{I}(z_i > k)\mathbb{E}_{q(v_k)}[\log(1 - v_k)] + \mathbb{I}(z_i = k)\mathbb{E}_{q(v_k)}[\log v_k]\\&= \sum_{k=1}^{\texttt{max\_K}} \mathbb{I}(z_i = k)[\sum_{l=1}^{k-1}\mathbb{E}_{q(v_k)}[\log(1 - v_k)] + \mathbb{E}_{q(v_k)}[\log v_k]]\\&= \sum_{k=1}^{\texttt{max\_K}} \mathbb{I}(z_i = k)\\&\left[\sum_{l=1}^{k-1}\big(\Psi(\beta_l) - \Psi(\alpha_l + \beta_l)\big) + \Psi(\alpha_k) - \Psi(\alpha_k + \beta_k)\right].\end{aligned}$$

Therefore, the update formula of $q(z_i; \varphi_{i,1}, \ldots, \varphi_{i,K})$ is

$$\begin{aligned}\varphi_{i,k} \propto \mathcal{N}(\mathbf{y}_i; \boldsymbol{\Phi}_i\mathbf{m}_k, \mathbf{C}_{i,k})\exp(-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_k\boldsymbol{\Phi}_i\mathbf{C}_{i,k}^{-1}\boldsymbol{\Phi}_i^\intercal))\\\exp\left(\sum_{l=1}^{k-1}\big(\Psi(\beta_l) - \Psi(\alpha_l + \beta_l)\big) + \Psi(\alpha_k) - \Psi(\alpha_k + \beta_k)\right).\end{aligned}$$

In the M-step, we optimize the expected log complete likelihood with respect to parameters $\boldsymbol{\Theta} = \{\mathbf{m}_b, \boldsymbol{\Sigma}_b\} \cup \{\boldsymbol{\theta}_k\}_{k=1}^{\texttt{max\_K}}$. Note that we keep α_0 fixed as in [23]. With the derived distribution $q(\boldsymbol{\Omega})$ in the E-step, the Q-function is

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\Theta}) = &\sum_{k=1}^{\texttt{max\_K}}(\log\mathcal{N}(\mathbf{m}_k; \mathbf{m}_b, \boldsymbol{\Sigma}_b) - \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_k\boldsymbol{\Sigma}_b^{-1}))\\&+\sum_{i=1}^N\sum_{k=1}^{\texttt{max\_K}}\varphi_{i,k}\Big(\log\mathcal{N}(\mathbf{y}_i; \boldsymbol{\Phi}_i\mathbf{m}_k, \mathbf{C}_{i,k})\\&-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_k\boldsymbol{\Phi}_i\mathbf{C}_{i,k}^{-1}\boldsymbol{\Phi}_i^\intercal)\Big) + \texttt{const}.\end{aligned}$$

We have omitted the terms that is independent with $\boldsymbol{\Theta}$. Only first two terms (except constant term) are related to $\mathbf{m}_b$ and $\boldsymbol{\Sigma}_b$. The derivatives of $\mathcal{Q}$ with respect to $\boldsymbol{\Sigma}_b$ and $\mathbf{m}_b$ are

$$\begin{aligned}\frac{\partial\mathcal{Q}}{\partial\boldsymbol{\Sigma}_b} = &\frac{1}{2}\boldsymbol{\Sigma}_b^{-1}\left(\boldsymbol{\Sigma}_k + \sum_{k=1}^{\texttt{max\_K}}(\mathbf{m}_k - \mathbf{m}_b)(\mathbf{m}_k - \mathbf{m}_b)^\intercal\right)\boldsymbol{\Sigma}_b^{-1}\\&-\frac{\texttt{max\_K}}{2}\boldsymbol{\Sigma}_b^{-1},\ \frac{\partial\mathcal{Q}}{\partial\mathbf{m}_b} = \sum_{k=1}^{\texttt{max\_K}}\boldsymbol{\Sigma}_b^{-1}(\mathbf{m}_k - \mathbf{m}_b).\end{aligned}$$

Setting the derivatives to zero, we obtain

$$\boldsymbol{\Sigma}_b = \frac{\sum_{k=1}^{\texttt{max\_K}}(\boldsymbol{\Sigma}_k + (\mathbf{m}_k - \mathbf{m}_b)(\mathbf{m}_k - \mathbf{m}_b)^\intercal)}{\texttt{max\_K}},$$

$$\mathbf{m}_b = \frac{\sum_{k=1}^{\texttt{max\_K}}\mathbf{m}_k}{\texttt{max\_K}}.$$

There is no closed-form solutions to estimate $\{\boldsymbol{\theta}_k\}_{k=1}^{\texttt{max\_K}}$, thus we use first-order optimization methods (such as conjugate gradient methods or quasi-Newton methods) to optimize $\mathcal{Q}(\boldsymbol{\Theta})$ with respect to $\{\boldsymbol{\theta}_k\}_{k=1}^{\texttt{max\_K}}$. Note that the objective is separable, thus we consider each $\boldsymbol{\theta}_k$ individually. According to [6], the gradient of $\mathcal{Q}(\boldsymbol{\Theta})$ with respect to $\theta_{k,s}$ where $s = 1, 2, 3$ is

$$\begin{aligned}\frac{\partial\mathcal{Q}(\boldsymbol{\Theta})}{\partial\theta_{k,s}} = &\sum_{i=1}^N\frac{1}{2}\varphi_{i,k}\mathrm{tr}\left(\mathbf{C}_{i,k}^{-1}\mathbf{S}_{i,k}\mathbf{C}_{i,k}^{-1}\frac{\partial\mathbf{C}_{i,k}}{\partial\theta_{k,s}}\right),\\&\mathbf{S}_{i,k} = (\mathbf{y}_i - \boldsymbol{\Phi}_i\mathbf{m}_k)(\mathbf{y}_i - \boldsymbol{\Phi}_i\mathbf{m}_k)^\intercal + \boldsymbol{\Phi}_i^\intercal\boldsymbol{\Sigma}_k\boldsymbol{\Phi}_i - \mathbf{C}_{i,k}.\end{aligned}\quad(8)$$

We can optimize $\mathcal{Q}(\boldsymbol{\Theta})$ according to the gradients derived above. In practice, the truncation level $\texttt{max\_K}$ is usually larger than the number of components needed to model the data. Therefore, we can abandon the kth component if $\sum_{i=1}^N\varphi_{i,k}$ is smaller than a threshold, thus the number of remaining components is reduced by 1. In this way, the algorithm automatically selects the proper number of components to fit the data. The complete variational EM algorithm for DPM-GPFR is summarized in Algorithm 1.

3.3. Complexity analysis of the variational EM algoritm

The main bottleneck of the computational cost lies in Line 7 and Line 18 in Algorithm 1. These two lines involve calculating the

Algorithm 1: The variational EM algorithm for learning DPM-GPFR.

while *not converged* **do**
  // Variational E-step
  Initialize variational parameters $\{\alpha_k, \beta_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \varphi_{1,k}, \ldots, \varphi_{N,k}\}_{k=1}^{\texttt{max\_K}}$;
  while *not converged* **do**
    for $k = 1, 2, \ldots, \texttt{max\_K}$ **do**
      if *k-th component is remained* **then**
        Update $\alpha_k = 1 + \sum_{i=1}^{N} \varphi_{i,k}$, $\beta_k = \alpha_0 + \sum_{i=1}^{N} \sum_{l=k+1}^{\texttt{max\_K}} \varphi_{i,j}$;
        Update $\boldsymbol{\Sigma}_k = \left(\boldsymbol{\Sigma}_b^{-1} + \sum_{i=1}^{N} \varphi_{i,k} \boldsymbol{\Phi}_i^{\intercal} \mathbf{C}_{i,k}^{-1} \boldsymbol{\Phi}_i\right)^{-1}$, $\mathbf{m}_k = \boldsymbol{\Sigma}_k \left(\boldsymbol{\Sigma}_b^{-1} \mathbf{m}_b + \sum_{i=1}^{N} \varphi_{i,k} \boldsymbol{\Phi}_i^{\intercal} \mathbf{C}_{i,k}^{-1} \mathbf{y}_i\right)$;
        for $i = 1, 2, \ldots, N$ **do**
          Update $\varphi_{i,k} \propto \mathcal{N}(\mathbf{y}_i; \boldsymbol{\Phi}_i \mathbf{m}_k, \mathbf{C}_{i,k}) \exp(-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}_k \boldsymbol{\Phi}_i \mathbf{C}_{i,k}^{-1} \boldsymbol{\Phi}_i^{\intercal}))$
          $\exp\left(\sum_{l=1}^{k-1} \left(\Psi(\beta_l) - \Psi(\alpha_l + \beta_l)\right)\right) + \Psi(\alpha_k) - \Psi(\alpha_k + \beta_k)\right)$
        end
      end
      if $\sum_{i=1}^{N} \varphi_{i,k} <$ threshold **then**
        Disgard *k*-th comonent;
      end
    end
  end
  // M-step
  for $k = 1, 2, \ldots, \texttt{max\_K}$ **do**
    if *k-th component is remained* **then**
      Update parameters $\boldsymbol{\Sigma}_b = \frac{\sum_{k=1}^{\texttt{max\_K}} (\boldsymbol{\Sigma}_k + (\mathbf{m}_k - \mathbf{m}_b)(\mathbf{m}_k - \mathbf{m}_b)^{\intercal})}{\texttt{max\_K}}$, $\mathbf{m}_b = \frac{\sum_{k=1}^{\texttt{max\_K}} \mathbf{m}_k}{\texttt{max\_K}}$;
      Using gradient ascent algorithm to optimize $\mathcal{Q}(\boldsymbol{\Theta})$ with respect to $\boldsymbol{\theta}_k$ according to Eq. (8);
      for $i = 1, 2, \ldots, N$ **do**
        Update $\mathbf{C}_{i,k}$ with new parameters $\boldsymbol{\theta}_k$;
      end
    end
  end
end

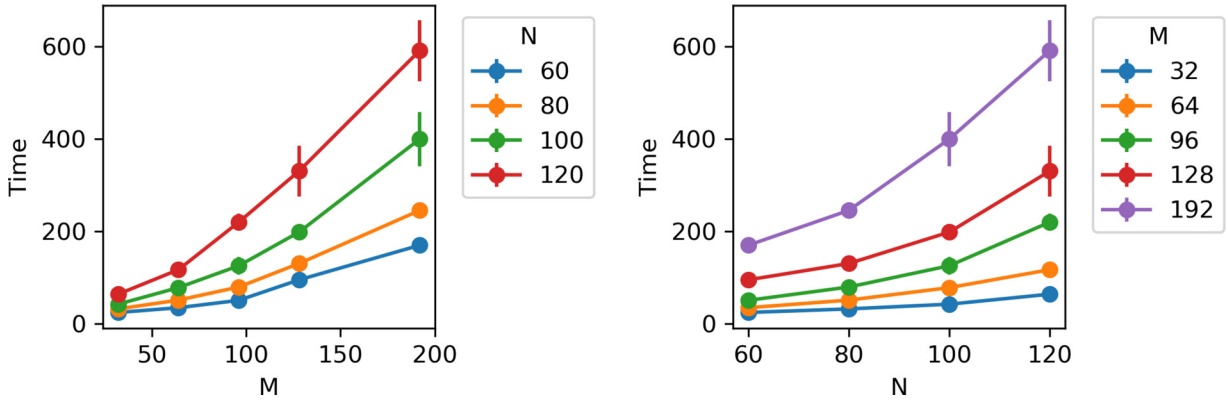


Fig. 2. Time cost of DPM-GPFR with respect to the dataset size N and the number of observations in each curve M. Here, we set $D = 20, \max_K = 30, \alpha_0 = 1$. There are for dataset size N there are $N/20$ components in total.

inverse of an $N_i \times N_i$ matrix, which require $\mathcal{O}(N_i^3)$. Since we need to iterate for N times, the total computational cost is approximately $\mathcal{O}(\sum_{i=1}^{N} N_i^3)$. If we further assume that each $\mathcal{D}_i$ has an equal length M, then the computational cost is $\mathcal{O}(NM^3)$, which grows linearly to dataset size N and cubically to M. However, such theoretical analysis provides limited information on practical running times, because the running time is significantly affected by the number of outer EM-iterations and the number of inner variational E-step iterations. From Fig. 2, we observe that the time cost of DPM-GPFR grows with both N and M, but the rate of growth is not exactly consistent with the theoretical results. Usually, the rate of growth of time cost with respect to N is larger than $\mathcal{O}(N)$. One possible explanation is when there are more curves, the algorithm is more difficult to converge, thus we need more iterations for the algorithm to terminate. Besides, the rate of growth of time cost with respect to M is smaller than $\mathcal{O}(M^3)$, which is due to fast hardware implementation of matrix operations.

In practice, the parameter α_0 and $\texttt{max\_K}$ influence the time cost, because they affect the number of remaining components during the learning process. The prior of stick-breaking variables influence the distribution of mixing proportions significantly. Some theoretical justifications can help us to better understand this ten-

dency. Note that the mean of Beta$(1,\alpha_0)$ is $\frac{1}{1+\alpha_0}$. When α_0 is small, we can expect that $\{v_k\}_{k=1}^{\infty}$ are closed to 1, thus π_k decrease to 0 rapidly since $\pi_k = v_k\prod_{l<k}(1-v_l)$, and there tends to be fewer components. However, when α_0 is large, we can expect that $\{v_k\}_{k=1}^{\infty}$ are close to 0, and the speed of π_k decreasing to 0 is rather slow, so there tend to be more components. In summary, the time cost is positively correlated to the scaling parameter α_0 provided that there are possibly infinite components. Taking into account the role of max_K, the situation is more complicated. If α_0 is small, the conclusion remains unchanged, and most mixing proportions are concentrating on the first few components. If α_0 is large, we do not have enough components to wait for π_k decreasing to 0, so mixing proportions tend to concentrate on the last few components due to $v_{\texttt{max\_K}} = 1$. This is similar to the case when α_0 is small. Therefore, in consideration of max_K, both large and small α_0 lead to a relatively fewer number of components.

For big functional data, we consider two cases. In the first case, the dataset size N is large. Since the computational cost grows linearly to N, thus the proposed method is scalable to N. In the second case, the lengths $\{N_i\}_{i=1}^N$ are large. Since the computational cost grows cubically to N_i, we should apply sparse approximation techniques in the Gaussian process community [26,27] to reduce the cost.

3.4. Prediction strategies

In the curve prediction problem, a test curve can be seen as the $(N+1)$-th curve, with known observations $\mathcal{D}_{N+1} = \{(x_{N+1,j}, y_{N+1,j})\}_{j=1}^{T}$. We wish to predict the response y_* at a new input x_* in the $(N+1)$-curve. Theoretically, after we obtain estimated parameters $\hat{\mathbf{\Theta}}$ and approximate posterior $q(\mathbf{\Omega})$, we can predict y_* via

$$p(y_*|x_*, \mathcal{D}, \mathcal{D}_{N+1}; \hat{\mathbf{\Theta}}) = \int p(y_*|x_*, \mathcal{D}_{N+1}, z_{N+1}, \mathbf{\Omega}; \hat{\mathbf{\Theta}}) p(z_{N+1}|\mathcal{D}_{N+1}, \mathbf{\Omega}; \hat{\mathbf{\Theta}}) q(\mathbf{\Omega}) \mathrm{d}z_{N+1} \mathrm{d}\mathbf{\Omega}.$$

However, this integral is intractable. Instead of using $q(\mathbf{\Omega})$, we use the mode as a point estimation $\hat{\mathbf{\Omega}}$ of $\mathbf{\Omega}$. First, the probability that $(N+1)$-curve belongs to kth component is

$$p(z_{N+1} = k|\mathcal{D}_{N+1}; \hat{\Omega}, \hat{\mathbf{\Theta}}) \propto \hat{v}_k \prod_{l=1}^{k-1}(1-\hat{v}_l)\mathcal{N}(\mathbf{y}_{N+1}; \mathbf{\Phi}_{N+1}\mathbf{m}_k, \mathbf{C}_{N+1,k}).$$

Condition on $z_{N+1} = k$, we can predict $\hat{y}^{(k)}$ using Gaussian process prediction Eq. (1),

$$\hat{y}^{(k)} = \boldsymbol{\phi}_*^{\intercal}\mathbf{m}_k + \mathbf{c}_k\mathbf{C}_{N+1,k}^{-1}(\mathbf{y}_{N+1} - \mathbf{\Phi}_{N+1}\mathbf{m}_k),$$

where $\boldsymbol{\phi}_* = [\phi_1(x_*), \ldots, \phi_D(x_*)]^{\intercal}$ and $\mathbf{c}_k = [c(x_{N+1,1}, x_*; \boldsymbol{\theta}_k), \ldots, c(x_{N+1,T}, x_*; \boldsymbol{\theta}_k)]^{\intercal}$. The final prediction is averaged over max_K components,

$$\hat{y}_* = \sum_{k=1}^{\texttt{max\_K}} p(z_{N+1} = k|\mathcal{D}_{N+1}; \hat{\Omega}, \hat{\mathbf{\Theta}})\hat{y}^{(k)}.$$

We refer to the prediction strategy described above as "fused prediction", since we integrate predictions from all components. On the other hand, if we set

$$\hat{y}_* = \hat{y}^{(k)}, \text{ where } k = \underset{l=1,\ldots,\texttt{max\_K}}{\arg\max}\ p(z_{N+1} = l|\mathcal{D}_{N+1}; \hat{\Omega}, \hat{\mathbf{\Theta}}),$$

then we call this strategy as "MAP prediction", since only the component corresponding to the MAP estimation of z_{N+1} is utilized. We can also ignore the structured noise and only consider the spline mean functions, which gives the "spline prediction",

$$\hat{y}_* = \sum_{k=1}^{\texttt{max\_K}} p(z_{N+1} = k|\mathcal{D}_{N+1}; \hat{\Omega}, \hat{\mathbf{\Theta}})\boldsymbol{\phi}_*^{\intercal}\mathbf{m}_k.$$

If we further simplify the predictions via using the MAP estimation of z_{N+1} to make spline prediction, *i.e.*,

$$\hat{y}_* = \boldsymbol{\phi}_*^{\intercal}\mathbf{m}_k, \text{ where } k = \underset{l=1,\ldots,\texttt{max\_K}}{\arg\max}\ p(z_{N+1} = l|\mathcal{D}_{N+1}; \hat{\Omega}, \hat{\mathbf{\Theta}}),$$

we call this prediction strategy "MAP spline prediction". Unless otherwise specified, we always use the fused prediction strategy in the experiments.

3.5. Practical issues

Unequal data length. Unequal data length is an important issue in batch data. Our method is adaptive and applicable to the unequal data length case since GPFR does not require that the data are time-aligned. The main reason is that Gaussian processes are powerful at interpolation between observations. Note that these $\{N_i\}_{i=1}^N$ may be different in general, and the samples points $\{\mathbf{x}_i\}_{i=1}^N$ may also be different. In the Line 6 and Line 19 of Algorithm 1, we iterate each $\mathcal{D}_i$ separately and do not need to concatenate $\{\mathbf{x}_i\}_{i=1}^N$ or $\{\mathbf{y}_i\}_{i=1}^N$ together.

Input range. Different input range is also a practical issue. In practice, different $\mathcal{D}_i$ may have different range of $\mathbf{x}_i$. This is important because it affects how to set knots of B-spline basis functions. In practice, we set the left end to be $-1.1 * \min_{i=1,\ldots,N}(\min_{j=1,\ldots,N_i} x_{i,j})$ and the right end to be $1.1 * \max_{i=1,\ldots,N}(\max_{j=1,\ldots,N_i} x_{i,j})$ such that the considered interval covers all $\{\mathbf{x}_i\}_{i=1}^N$.

Generalization on new sources. We consider the case that the DPM-GPFR needs to make predictions on an unseen new source batch which has not been used for training. A GPFR source consists of a mean function and a Gaussian process noise. Temporarily, we ignore the GP part since the mean function plays a more important role in prediction. The problem boils down to: how well can we approximate an unseen new trend function $\mu_*(x)$ based on seen and learned mean functions $\mu_1(x), \ldots, \mu_K(x)$? According to the model assumption, we may further write these mean functions as a linear combination of D B-spline basis functions, *i.e.*, $\mu_k(x) = \boldsymbol{\phi}(x)^{\intercal}\mathbf{b}_k$ and $\mu_*(x) = \boldsymbol{\phi}(x)^{\intercal}\mathbf{b}_*$. Using the fused prediction strategy, the obtained predicting function is a linear combination of $\{\mu_k(x)\}_{k=1}^K$. *i.e.*, $\hat{\mu}(x) = \sum_{k=1}^K \lambda_k\mu_k(x)$. Let $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_K]^{\intercal}$, $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_K]$, then $\hat{\mu}(x)$ can be rewritten as $\boldsymbol{\phi}(x)^{\intercal}\mathbf{B}\boldsymbol{\lambda}$. For simplicity, we assume $\{\mathbf{b}_k\}_{k=1}^K$ are linearly independent. Intuitively, the performance would be satisfying if $\mu_*(x)$ is similar to $\{\mu_k(x)\}_{k=1}^K$. In this case, $\hat{\mu}(x)$ is able to approximate $\mu_*(x)$ well. The key-point is to measure the similarity between sources. The ℓ_2 norm between $\mu_*(x)$ and $\hat{\mu}(x)$ is

$$\begin{aligned}\|\mu_*(x) - \hat{\mu}(x)\|_2^2 &= \int (\mu_*(x) - \hat{\mu}(x))^2\mathrm{d}x \\ &= \int (\boldsymbol{\phi}(x)^{\intercal}\mathbf{b}_* - \boldsymbol{\phi}(x)^{\intercal}\mathbf{B}\boldsymbol{\lambda})^2\mathrm{d}x \\ &= \int \Big(\mathbf{b}_*^{\intercal}(\boldsymbol{\phi}(x)\boldsymbol{\phi}(x)^{\intercal})\mathbf{b}_* - 2\boldsymbol{\lambda}^{\intercal}\mathbf{B}^{\intercal}\boldsymbol{\phi}(x)\boldsymbol{\phi}(x)^{\intercal}\mathbf{b}_* \\ &\quad + \boldsymbol{\lambda}^{\intercal}\mathbf{B}^{\intercal}\boldsymbol{\phi}(x)\boldsymbol{\phi}(x)^{\intercal}\mathbf{B}\boldsymbol{\lambda}\Big)\mathrm{d}x \\ &= \mathbf{b}_*^{\intercal}\mathbf{W}\mathbf{b}_* - 2\boldsymbol{\lambda}^{\intercal}\mathbf{B}^{\intercal}\mathbf{W}\mathbf{b}_* + \boldsymbol{\lambda}^{\intercal}\mathbf{B}^{\intercal}\mathbf{W}\mathbf{B}\boldsymbol{\lambda}, \end{aligned} \tag{9}$$

where $\mathbf{W} = \int \boldsymbol{\phi}(x)\boldsymbol{\phi}(x)^{\intercal}\mathrm{d}x$, *i.e.*, $[\mathbf{W}]_{ij} = \int \phi_i(x)\phi_j(x)\mathrm{d}(x)$. This is a weighted least square problem, and the optimal value is $(\mathbf{b}_* - \mathbf{B}\boldsymbol{\lambda}_\star)^{\intercal}\mathbf{W}(\mathbf{b}_* - \mathbf{B}\boldsymbol{\lambda}_\star)$, where $\boldsymbol{\lambda}_\star = (\mathbf{B}^{\intercal}\mathbf{W}\mathbf{B})^{-1}\mathbf{B}^{\intercal}\mathbf{W}\mathbf{b}_*$. From this equation, we can see that the approximation performance is determined by the distance from $\mathbf{b}_*$ to the vector space spanned by $\{\mathbf{b}_1, \ldots, \mathbf{b}_K\}$, with a weighted inner product $\langle \mathbf{b}_*, \mathbf{b}_k\rangle = \mathbf{b}_*^{\intercal}\mathbf{W}\mathbf{b}_k$. In practice, however, there are other factors that influence the result. First, once D is given, we can only express the mean functions with a linear combination of D B-spline basis functions. How well can we approximate these mean functions depends on their intrinsic

properties and the choice of D. Second, the GP noise also influences the performance, especially when the signal-noise ratio is low. Third, when we use the fuse prediction strategy, the weights calculated as in Section 3.4 are not guaranteed to be $\boldsymbol{\lambda}_\star$. Therefore, the above analysis provides us some insights on the performance of DP-GPFR and related methods on data generated by a new source, but the practical performance also depends on many other factors.

Standardization. As for standardizing the batch data, we do not standardize $\mathbf{x}_i$. For $\mathbf{y}_i$, we choose to standardize them using

$$\text{min\_y} = \min_{i=1,\dots,N}(\min_{j=1,\dots,N_i} y_{i,j}),\ \text{max\_y} = \max_{i=1,\dots,N}(\max_{j=1,\dots,N_i} y_{i,j}),$$
$$y_{ij} \leftarrow \frac{y_{ij} - \text{min\_y}}{\text{max\_y} - \text{min\_y}}.$$

Other standardization methods can also be used.

Feature selection. Side-information may help to characterize the evolving patterns of the batch data. There may be some other covariates besides x. This direction has been explored by many excellent previous works [5,6]. In general, automatic relevance determination (ARD) and penalizing regression coefficients are the most important feature selection techniques in this area, and they can be applied in our model straightforwardly.

3.6. Possible variants

Each component of the DPM-GPFR model consists of two parts, the B-spline mean function, and the Gaussian process noise. One natural problem is: whether both parts are necessary? To solve this problem, we can consider two simple variants:

- DPM-GP: eliminating the B-spline mean functions in DPM-GPFR, *i.e.*, each curve is generated by a Gaussian process with zero mean function.
- DPM-FR: eliminating the Gaussian process noises in DPM-GPFR, *i.e.*, each curve is generated by the mean function with independent identically distributed Gaussian noises.

Besides, the scaling parameter α_0 in DPM-GPFR is treated as a hyper-parameter. Similar to [28], we can further impose a Gamma prior $\text{Gamma}(\eta_1, \eta_2)$ on α_0. In the variational E-step, the approximate posterior of α_0 is also a Gamma distribution $\text{Gamma}(\eta_1, \eta_2 - \sum_{k=1}^{K}(\Psi(\beta_k) - \Psi(\alpha_k + \beta_k)))$. We refer to this model as DPM-GPFR$_{\alpha_0}$.

We can also extend the Dirichlet process prior on latent variables to the Pitman-Yor process [28]. This can be achieved by sampling variables v_k from $\text{Beta}(1-\delta, \alpha_0 + k\delta)$ in Eq. (2), where δ is a parameter. Compared with the Dirichlet process, the Pitman-Yor process prior enjoys the power law property. We refer to this model as PYP-GPFR.

Variational tempering technique [29] can be utilized in the inference process. Given a finite collection of temperatures $1 = T_1 < T_2 < \ldots < T_M$, we can assign a temperature variable t_i to the ith curve $\mathcal{D}_i$ and deform its likelihood as $(p(\mathbf{y}_i|\mathbf{x}_i, z_i, \{\mathbf{b}_k\}_{k=1}^{\infty}; \{\boldsymbol{\theta}_k\}_{k=1}^{\infty})p(z_i|\mathbf{v}))^{1/t_i}$. This is equivalent to reweight log-likelihood terms in Eq. (7) by $\frac{1}{t_i}$. We refer to this model as DPM-GPFR-VT.

4. Experiments

4.1. Dataset description and experimental settings

We conduct experiments on both synthetic datasets and real-world datasets. Since the DPM-GPFR model involves potentially infinite components, it is difficult to generate synthetic datasets based on it. Instead, we use the mix-GPFR model with K components to generate these datasets, with each component consisting of 20 curves for training and 10 curves for testing. We have 100 observed samples in each curve, randomly positioned in the interval $[-3, 3]$. For a testing curve, the first half of the samples are known and the task is to predict the rest points. We set $K = 3, 4, \ldots, 10$ to obtain 8 datasets with different number of components, and we refer to them as $\mathcal{S}_3, \mathcal{S}_4, \ldots, \mathcal{S}_{10}$, respectively. For each component, the curves are generated by a mean function together with Gaussian process noises. The mean functions and hyper-parameters of the Gaussian processes are listed in Table 2. These mean functions are specially designed to cover various common properties of functions, such as periodicity, monotonicity, and symmetry, and the parameters of Gaussian processes correspond to varying length scales, noise levels, *etc*.. The dataset with K components involves the first K rows of Table 2. Therefore, it becomes harder and harder to cluster the curves and make predictions as we increase K.

We use the electricity load dataset issued by the Northwest China Grid Company and the drosophila development dataset [21,30]. The electricity dataset records electricity loads in 2009 and 2010 every 15 min. Therefore, daily electricity loads can be regarded as a curve with 96 samples. We split the dataset according to the year, and refer to them as Electricity 2009/2010, respectively. Each sub-datasets consists of 200 curves for training and 165 curves for testing. The drosophila dataset records the gene expression of six species of Drosophila, measured at two hour intervals during embryonic development. Among the 500 records, 300 are used for training and 200 are used for testing.

We compare DPM-GPFR with several competing methods, which are summarized as follows:

- GP: a single Gaussian process with zero mean function to model batch data, as stated in Section 2.1.
- GPFR: a single Gaussian process with B-spline mean function to model curves, as stated in Section 2.2.
- mix-GP: an output-mixture of Gaussian processes with zero mean functions to model batch data.
- mix-GPFR: an output-mixture of Gaussian processes with B-spline mean functions to model batch data.
- MOHGP: a Dirichlet process based mixture of Gaussian processes with Gaussian process mean functions [21].
- Variants of DPM-GPFR as mentioned in Section 3.6.

GPFR, mix-GPFR and DPM-GPFR involve B-spline mean functions, and we set $D = 20$ on synthetic datasets and $D = 30$ on real-world datasets. For GP and GPFR, we learn the parameters via Type-II maximum likelihood estimation, which maximizes the marginal likelihood via the conjugate gradient method. For mix-GP and mix-GPFR, we learn the parameters via the hard-cut EM algorithm [31]. We denote the estimated number of components as $\hat{K}$. For DPM-GPFR, we have three main hyper-parameters: the scaling parameter α_0, the truncation level `max_K`, and the number of knots D. Unless otherwise specified, we set `max_K` $= 30$ and $\alpha_0 = 1$. The sensitivity of these hyper-parameters is analyzed in Section 4.6.

4.2. Evaluation metrics

We evaluate the performance from two aspects: curve prediction and curve clustering. Curve prediction concerns whether we can effectively predict the values at unknown inputs in the testing curves. We use the Rooted Mean Square Error (RMSE) between predicted values and ground-truths to measure the prediction performance. Besides, *testing* R-squared (R^2) is also used for evaluating the prediction performance. Curve clustering concerns whether we can reveal the mixture structure underlying the data. Besides the Adjusted Rand Index (ARI) [32], we also propose several novel metrics to further analyze the clustering results.

Definition 1 (Generalized Classification Accuracy Rate)**.** Suppose that there are N samples, let $c(i) \in \{1, \ldots, K\}$ be the ground-truth

Table 1
List of notations and symbols used in Section 3.

Notations	Descriptions
$\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^N$, $\mathcal{D}_i = \{(x_{im}, y_{im})\}_{m=1}^{N_i}$	$\mathcal{D}$ is the batch dataset, and $\mathcal{D}_i$ is the i-the curve consisting of $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{iN_i}]^\intercal$ and $\mathbf{y}_i = [y_{i1}, y_{i2}, \ldots, y_{iN_i}]^\intercal$, where N_i is the number of observations in the i-th curve.
$\mathbf{v} = \{v_k\}_{k=1}^\infty$	Stick-breaking variables, $v_k \sim \text{Beta}(1, \alpha_0)$.
α_0	The scaling parameter of the Dirichlet process.
z_i	The latent indicator variable associated with $\mathcal{D}_i$.
$\mathcal{GPFR}(x; \mathbf{b}_k, \boldsymbol{\theta}_k)$	The k-th Gaussian process functional regression (GPFR) component. Here, $\mathbf{b}_k$ is the coefficient of B-spline basis functions, and $\boldsymbol{\theta}_k = [\theta_{k,1}, \theta_{k,2}, \theta_{k,3}]^\intercal$ are the parameters.
$\mathcal{N}(\mathbf{m}_b, \boldsymbol{\Sigma}_b)$	The prior distribution for $\{\mathbf{b}_k\}_{k=1}^\infty$. $\mathcal{N}$ denotes multivariate Gaussian distribution, and $\mathbf{m}_b$, $\boldsymbol{\Sigma}_b$ are the mean and covariance matrix respectively.
max_K	The truncation level.
π_k	The mixing proportion of the k-th GPFR component.
$\mathbf{X}, \mathbf{Y}$	$\mathbf{X}$ is the concatenation of $\{\mathbf{x}_i\}_{i=1}^N$; $\mathbf{Y}$ is the concatenation of $\{\mathbf{y}_i\}_{i=1}^N$
$\boldsymbol{\Omega}, \boldsymbol{\Theta}$	$\boldsymbol{\Omega} = \{z_i\}_{i=1}^N \cup \{v_k\}_{k=1}^\infty \cup \{\mathbf{b}_k\}_{k=1}^\infty$ denotes all latent variables, while $\boldsymbol{\Theta} = \{\pi_k, \boldsymbol{\theta}_k\}_{k=1}^\infty \cup \{\mathbf{m}_b, \boldsymbol{\Sigma}_b\}$ denotes all parameters.
$\{\phi_d(x)\}_{d=1}^D$, $\boldsymbol{\Phi}_i$	$\{\phi_d(x)\}_{d=1}^D$ denotes D B-spline basis functions, $\boldsymbol{\Phi}_i \in \mathbb{R}^{N_i \times D}$ is the B-spline basis matrix, *i.e.*, the (l, d)-th element of $\boldsymbol{\Phi}_i$ is $\phi_d(\mathbf{x}_{il})$.
$\mathbf{C}_{i,k}$	$\mathbf{C}_{i,k}$ is the covariance matrix of $\mathbf{x}_i$ calculated by parameters $\boldsymbol{\theta}_k$, *i.e.*, the (p, q)-th element of $\mathbf{C}_{i,k}$ is $c(\mathbf{x}_{ip}, \mathbf{x}_{iq}; \boldsymbol{\theta}_k)$.
α_k, β_k	The variational parameters of v_k. We use $\text{Beta}(\alpha_k, \beta_k)$ to approximate the posterior distribution of v_k.
$\mathbf{m}_k, \boldsymbol{\Sigma}_k$	The variational parameters of $\mathbf{b}_k$. We use $\mathcal{N}(\mathbf{m}_k, \boldsymbol{\Sigma}_k)$ to approximate the posterior distribution of $\mathbf{b}_k$.
$\varphi_{i,1}, \ldots, \varphi_{i,K}$	The variational parameters of z_i. We use $\text{Categorical}(\varphi_{i,1}, \ldots, \varphi_{i,K})$ to approximate the posterior distribution of z_i.
Ψ	The digamma function.
$\mathcal{D}_{N+1} = \{(x_{N+1,j}, y_{N+1,j})\}_{j=1}^T, x_*, y_*$	$\mathcal{D}_{N+1}$ denotes a testing curve, while $\{(x_{N+1,j}, y_{N+1,j})\}_{j=1}^T$ are known samples and the aim is to predict the response y_* at x_*
$z_{N+1}, \boldsymbol{\phi}_*, \mathbf{c}_k$	z_{N+1} denotes the indicator variable of the testing curve, and $\boldsymbol{\phi}_* = [\phi_1(x_*), \ldots, \phi_D(x_*)]^\intercal$ and $\mathbf{c}_k = [c(x_{N+1,1}, x_*; \boldsymbol{\theta}_k), \ldots, c(x_{N+1,T}, x_*; \boldsymbol{\theta}_k)]^\intercal$.

Table 2
Mean functions and hyper-parameters of Gaussian processes of the synthetic datasets.

Mean function	Hyper-parameters of GPs		
	θ_1	θ_2	θ_3
x^2	0.500	2.000	0.150
$(-4(x+1.5)^2+9)\mathbb{I}(x<0)$ $+(4(x-1.5)^2-9)\mathbb{I}(x\geq 0)$	0.528	2.500	0.144
$8\sin(1.5x-1)$	0.556	3.333	0.139
$\sin(1.5x)+2x-5$	0.583	5.000	0.133
$-0.5x^2+\sin(4x)-2x$	0.611	10.000	0.128
$-x^2$	0.639	10.000	0.122
$(4(x+1.5)^2-9)\mathbb{I}(x<0)$ $+(-4(x-1.5)^2+9)\mathbb{I}(x\geq 0$	0.667	5.000	0.117
$5\cos(3x+2)$	0.694	3.333	0.111
$\cos(1.5x)-2x+5$	0.722	2.500	0.106
$0.5x^2+\cos(4x)+2x$	0.750	2.000	0.100

labels and $\hat{c}(i) \in \{1, \ldots, \hat{K}\}$ be the clustering result, then the generalized classification accuracy rate is defined as

$$\text{GCAR}(c, \hat{c}) = \max_{f:\{1,\ldots,\hat{K}\}\to\{1,\ldots,K\}} \frac{\sum_{i=1}^N \mathbb{I}(f(\hat{c}(i)) = c(i))}{N}.$$

Definition 2 (Earth Mover's Distance)**.** Suppose that there are N samples, let $c(i) \in \{1, \ldots, K\}$ be the ground-truth labels and $\hat{c}(i) \in \{1, \ldots, \hat{K}\}$ be the clustering result. Let $n_{k,l} = |\{i : c(i) = l \text{ and } \hat{c}(i) = k\}|$ and consider the following optimization problem:

$$\min_{F_{k,l}} \frac{1}{N} \sum_{k=1}^{\hat{K}} \sum_{l=1}^{K} F_{k,l} D_{k,l}, \tag{10}$$

$$\sum_{k=1}^{\hat{K}} F_{k,l} = \sum_{k=1}^{\hat{K}} n_{k,l} \quad , \quad \sum_{l=1}^{K} F_{k,l} = \sum_{l=1}^{K} n_{k,l} \quad , \quad F_{k,l} \geq 0$$

If $D_{k,l} = 1 - n_{k,l} / \sum_{l=1}^K n_{k,l}$, the solution of Eq. (10) is defined to be EMD-I$(c, \hat{c})$; If $D_{k,l} = 1 - n_{k,l} / \sum_{k=1}^{\hat{K}} n_{k,l}$, the solution of Eq. (10) is defined to be EMD-II$(c, \hat{c})$.

Theorem 1.

1. *GCAR$(c, \hat{c})$ equals to 1 if and only if $\hat{c}(i) \neq \hat{c}(j)$ for any $c(i) \neq c(j)$.*
2. *EMD-I$(c, \hat{c})$ equals to 0 if and only if $\hat{c}(i) \neq \hat{c}(j)$ for any $c(i) \neq c(j)$.*
3. *EMD-II$(c, \hat{c})$ equals to 0 if and only if $\hat{c}(i) = \hat{c}(j)$ for any $c(i) = c(j)$.*

Intuitively, GCAR and EMD-I evaluate how well the algorithm distinguishes elements that belong to different components, while EMD-II evaluates whether similar elements are assigned to the same cluster. GCAR and EMD-I prefer fine structures, while EMD-II prefers coarse structures. As two extreme examples, $\hat{c}(i) = i$ leads to GCAR$(c, \hat{c}) = 1$ and EMD-I$(c, \hat{c}) = 0$, and $\hat{c}(i) = 1$ leads to EMD-II$(c, \hat{c}) = 0$.

4.3. Prediction performances

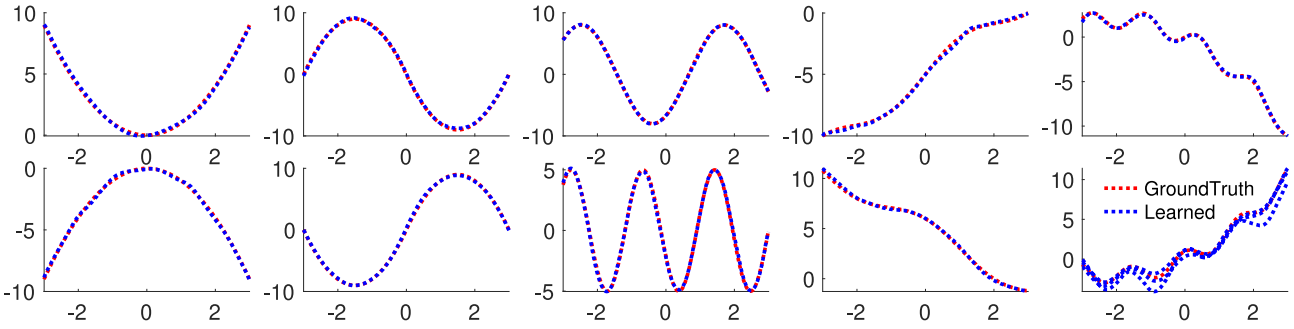
The prediction performances of the proposed and competing methods on synthetic datasets and real-world datasets are shown in Table 3 and Table 4, respectively. All the records are averaged over 10 runs. From these tables, we observe that a single Gaussian process is not flexible enough to model the data. By introducing the B-spline mean function, the RMSE reduces significantly, especially on real-world datasets. Similar observation also holds for mix-GP/mix-GPFR and DPM-GP/DPM-GPFR. This demonstrates the necessity of using functional regression. Besides, mix-GPFR performs much better than a single GPFR, thus it is vital to introduce the mixture structure to model the data. On real-world datasets, although occasionally mix-GPFR outperforms DPM-GPFR marginally, the results of mix-GPFR are very sensitive to the choice $\hat{K}$, which is a major challenge in finite mixture models. The time consumption of mix-GPFR with large $\hat{K}$ is relatively high, and

Table 3
Average RMSEs (Rooted Mean Square Errors), R-Squared (R^2) indices and running times (in seconds) with standard deviations of the proposed and competing methods on synthetic datasets $\mathcal{S}_3$ – $\mathcal{S}_{10}$.

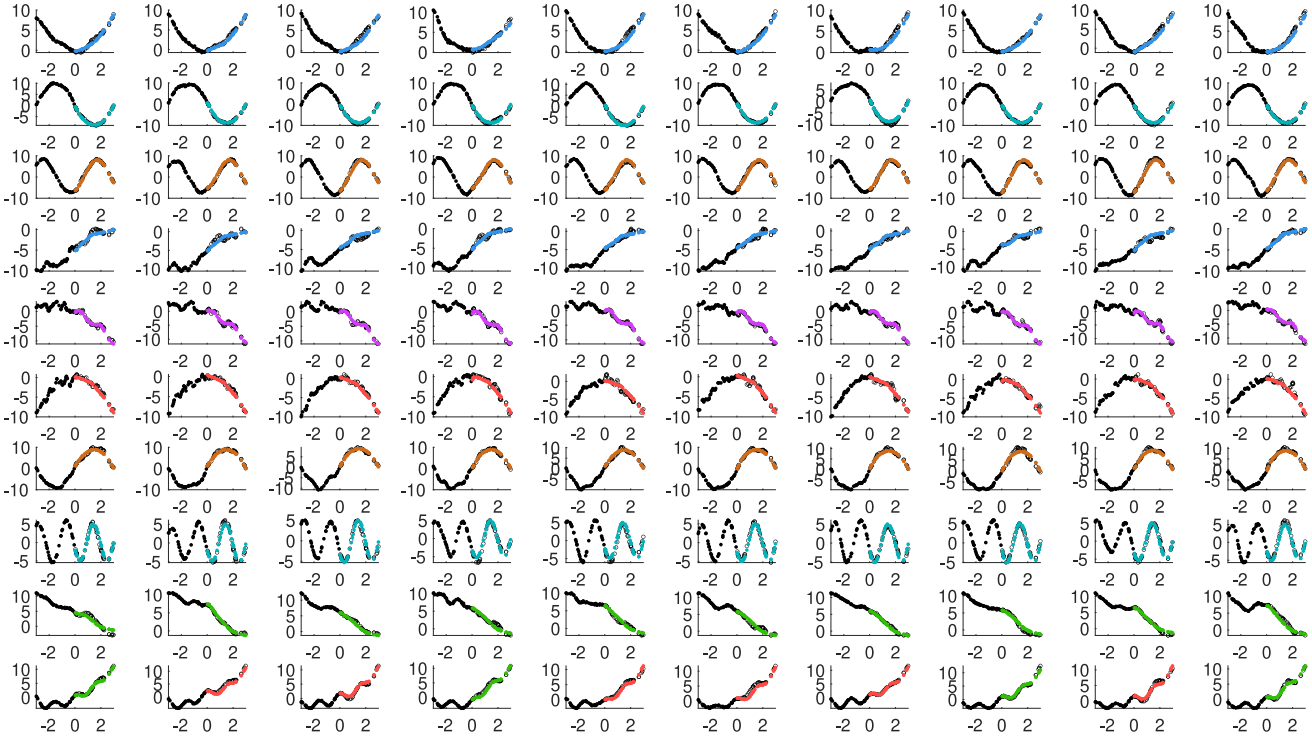
Method	$\mathcal{S}_3$			$\mathcal{S}_4$			$\mathcal{S}_5$		
	RMSE	R^2	Time	RMSE	R^2	Time	RMSE	R^2	Time
GP	4.6923 ± 0.0000	-0.1378 ± 0.0000	0.59 ± 0.15	4.5986 ± 0.0000	-0.1905 ± 0.0000	0.57 ± 0.02	4.5716 ± 0.0000	-0.0951 ± 0.0000	0.79 ± 0.02
mix-GP	4.5823 ± 0.0067	-0.1280 ± 0.0028	3.06 ± 0.33	4.2613 ± 0.0349	-0.1237 ± 0.0117	4.71 ± 0.34	4.1541 ± 0.0286	-0.0153 ± 0.0071	6.03 ± 0.95
GPFR	4.6051 ± 0.0000	0.0782 ± 0.0000	1.01 ± 0.05	4.5634 ± 0.0000	-0.0739 ± 0.0000	0.84 ± 0.05	4.4592 ± 0.0000	-0.1975 ± 0.0000	0.97 ± 0.07
mix-GPFR	1.1378 ± 0.0000	0.9247 ± 0.1008	2.07 ± 0.05	0.7225 ± 0.0000	0.9361 ± 0.0928	3.08 ± 0.06	0.9437 ± 0.1625	0.9044 ± 0.0690	5.67 ± 3.25
DPM-FR	0.5316 ± 0.0280	0.9837 ± 0.0014	56.89 ± 6.42	0.5790 ± 0.0123	0.9757 ± 0.0014	66.44 ± 2.18	0.6160 ± 0.0038	0.9751 ± 0.0016	74.14 ± 5.06
DPM-GP	4.5814 ± 0.0153	-0.1174 ± 0.0125	75.19 ± 48.67	4.2365 ± 0.0257	-0.1201 ± 0.0267	46.22 ± 42.14	4.1543 ± 0.0163	-0.0356 ± 0.0248	43.98 ± 20.59
DPM-GPFR	0.4802 ± 0.0045	0.9870 ± 0.0007	67.67 ± 5.61	0.5324 ± 0.0027	0.9800 ± 0.0002	96.92 ± 9.01	0.5832 ± 0.0024	0.9783 ± 0.0001	135.34 ± 15.92
DPM-GPFR-VT	0.4776 ± 0.0006	0.9874 ± 0.0001	75.59 ± 18.97	0.5322 ± 0.0004	0.9802 ± 0.0003	108.55 ± 11.15	0.5836 ± 0.0009	0.9783 ± 0.0001	165.56 ± 14.42
DPM-GPFR$_{\alpha_0}$	0.4772 ± 0.0007	0.9873 ± 0.0003	75.95 ± 7.99	0.5324 ± 0.0009	0.9800 ± 0.0001	89.39 ± 29.02	0.5822 ± 0.0018	0.9784 ± 0.0002	135.35 ± 11.74
PYP-GPFR	0.4771 ± 0.0015	0.9872 ± 0.0006	69.69 ± 7.63	0.5334 ± 0.0024	0.9800 ± 0.0003	100.67 ± 6.03	0.5826 ± 0.0013	0.9785 ± 0.0004	133.93 ± 16.01
Method		$\mathcal{S}_6$			$\mathcal{S}_7$			$\mathcal{S}_8$	
	RMSE	R^2	Time	RMSE	R^2	Time	RMSE	R^2	Time
GP	4.4533 ± 0.0000	-0.3096 ± 0.0000	0.89 ± 0.03	5.3324 ± 0.0000	-0.5298 ± 0.0000	0.85 ± 0.05	4.8416 ± 0.0000	-0.2657 ± 0.0000	1.18 ± 0.05
mix-GP	4.1588 ± 0.0108	-0.2760 ± 0.0043	8.59 ± 3.03	4.9060 ± 0.0052	-0.3983 ± 0.0081	18.98 ± 6.24	4.5017 ± 0.0210	-0.2132 ± 0.0036	31.23 ± 13.06
GPFR	4.2163 ± 0.0000	-0.3813 ± 0.0000	1.57 ± 0.33	5.2768 ± 0.0000	-0.5153 ± 0.0000	1.25 ± 0.06	4.7652 ± 0.0000	-0.2450 ± 0.0000	1.89 ± 0.05
mix-GPFR	0.9587 ± 0.0715	0.8248 ± 0.1948	9.11 ± 8.73	1.3309 ± 0.3587	0.8848 ± 0.0957	18.07 ± 12.94	0.8627 ± 0.2280	0.8326 ± 0.1174	14.90 ± 7.82
DPM-FR	0.6316 ± 0.0111	0.9646 ± 0.0011	77.39 ± 2.93	0.6834 ± 0.0056	0.9647 ± 0.0013	79.94 ± 5.48	0.6723 ± 0.0114	0.9683 ± 0.0015	89.24 ± 5.21
DPM-GP	4.1657 ± 0.0106	-0.2796 ± 0.0040	69.12 ± 30.45	4.8759 ± 0.0132	-0.3755 ± 0.0101	103.18 ± 35.41	4.4845 ± 0.0065	-0.2101 ± 0.0093	168.50 ± 40.82
DPM-GPFR	0.6049 ± 0.0045	0.9679 ± 0.0004	216.87 ± 54.95	0.6466 ± 0.0028	0.9692 ± 0.0004	263.47 ± 50.85	0.6450 ± 0.0036	0.9717 ± 0.0004	234.73 ± 21.14
DPM-GPFR-VT	0.6003 ± 0.0019	0.9681 ± 0.0003	224.04 ± 20.54	0.6420 ± 0.0010	0.9693 ± 0.0003	275.83 ± 25.93	0.6458 ± 0.0038	0.9714 ± 0.0006	310.71 ± 26.05
DPM-GPFR$_{\alpha_0}$	0.6032 ± 0.0036	0.9679 ± 0.0004	199.23 ± 28.22	0.6462 ± 0.0048	0.9691 ± 0.0005	235.97 ± 30.30	0.6459 ± 0.0046	0.9716 ± 0.0003	257.59 ± 29.62
PYP-GPFR	0.6017 ± 0.0033	0.9675 ± 0.0005	190.65 ± 16.34	0.6452 ± 0.0031	0.9694 ± 0.0003	239.71 ± 42.28	0.6462 ± 0.0040	0.9714 ± 0.0005	247.75 ± 23.84
Method		$\mathcal{S}_9$			$\mathcal{S}_{10}$				
	RMSE	R^2	Time	RMSE	R^2	Time			
GP	4.6994 ± 0.0000	-0.3199 ± 0.0000	1.07 ± 0.01	4.6713 ± 0.0000	-0.1614 ± 0.0000	1.43 ± 0.02			
mix-GP	4.3722 ± 0.0038	-0.2392 ± 0.0023	33.21 ± 13.93	4.4342 ± 0.0048	-0.0729 ± 0.0026	35.52 ± 9.80			
GPFR	4.6265 ± 0.0000	-0.2788 ± 0.0000	2.06 ± 0.06	4.6124 ± 0.0000	-0.1397 ± 0.0000	2.25 ± 0.05			
mix-GPFR	0.9868 ± 0.4047	0.7904 ± 0.0999	31.39 ± 26.08	1.2715 ± 0.5026	0.8675 ± 0.0727	43.21 ± 33.05			
DPM-FR	0.6610 ± 0.0102	0.9634 ± 0.0011	93.92 ± 8.57	0.6723 ± 0.0035	0.9665 ± 0.0044	111.31 ± 7.99			
DPM-GP	4.3644 ± 0.0093	-0.2382 ± 0.0040	129.27 ± 38.33	4.4307 ± 0.0066	-0.0737 ± 0.0022	160.04 ± 32.27			
DPM-GPFR	0.6316 ± 0.0020	0.9667 ± 0.0006	255.42 ± 40.18	0.6391 ± 0.0024	0.9708 ± 0.0004	448.03 ± 120.50			
DPM-GPFR-VT	0.6305 ± 0.0015	0.9666 ± 0.0008	301.48 ± 87.64	0.6388 ± 0.0021	0.9709 ± 0.0001	397.23 ± 99.85			
DPM-GPFR$_{\alpha_0}$	0.6394 ± 0.0116	0.9669 ± 0.0009	329.21 ± 33.96	0.6386 ± 0.0028	0.9708 ± 0.0003	321.88 ± 102.34			
PYP-GPFR	0.6386 ± 0.0122	0.9615 ± 0.0164	307.42 ± 24.27	0.6386 ± 0.0019	0.9657 ± 0.0142	378.59 ± 105.93			

Table 4
Average RMSEs (Rooted Mean Square Errors), R-Squared (R^2) indices and running times (in seconds) with standard deviations of the proposed and competing methods on real-world datasets.

Method	Electricity 2009				Electricity 2010				Drosophila			
	$\hat{K}$	RMSE	R^2	Time	$\hat{K}$	RMSE	R^2	Time	$\hat{K}$	RMSE	R^2	Time
GP	1	0.9231 ± 0.0000	0.0315 ± 0.0000	1.72 ± 0.05	1	0.8838 ± 0.0000	0.0927 ± 0.0000	1.45 ± 0.03	1	0.7465 ± 0.0000	0.3037 ± 0.0000	1.84 ± 0.03
mix-GP	3	0.9381 ± 0.0000	-0.0147 ± 0.0000	7.26 ± 0.27	3	0.8885 ± 0.0000	0.0789 ± 0.0000	8.80 ± 0.07	3	0.6796 ± 0.0019	0.4297 ± 0.0010	30.21 ± 2.78
	5	0.9384 ± 0.0025	-0.0189 ± 0.0055	13.90 ± 1.04	5	0.8889 ± 0.0048	0.0728 ± 0.0162	13.86 ± 1.30	5	0.6742 ± 0.0005	0.4418 ± 0.0017	43.59 ± 4.79
	10	0.9395 ± 0.0013	-0.0165 ± 0.0045	23.30 ± 1.78	10	0.8952 ± 0.0006	0.0493 ± 0.0005	25.72 ± 3.36	10	0.6758 ± 0.0002	0.4400 ± 0.0006	83.69 ± 9.92
	15	0.9391 ± 0.0013	-0.0149 ± 0.0011	32.13 ± 3.62	15	0.8951 ± 0.0005	0.0496 ± 0.0006	36.13 ± 4.55	15	0.6758 ± 0.0004	0.4398 ± 0.0005	111.34 ± 15.08
	20	0.9388 ± 0.0006	-0.0177 ± 0.0025	34.94 ± 3.05	20	0.8955 ± 0.0005	0.0496 ± 0.0003	42.36 ± 4.70	20	0.6757 ± 0.0003	0.4400 ± 0.0005	130.72 ± 20.13
	30	0.9383 ± 0.0007	-0.0153 ± 0.0021	45.80 ± 4.19	30	0.8957 ± 0.0005	0.0494 ± 0.0009	54.21 ± 6.61	30	0.6757 ± 0.0003	0.4399 ± 0.0008	166.17 ± 26.45
	50	0.9386 ± 0.0005	-0.0148 ± 0.0025	67.96 ± 6.93	50	0.8958 ± 0.0005	0.0494 ± 0.0007	75.94 ± 9.79	50	0.6758 ± 0.0003	0.4398 ± 0.0010	220.87 ± 48.31
GPFR	1	0.5651 ± 0.0000	0.6350 ± 0.0000	2.36 ± 0.05	1	0.5338 ± 0.0000	0.6723 ± 0.0000	2.17 ± 0.07	1	0.5396 ± 0.0000	0.5864 ± 0.0000	2.70 ± 0.13
mix-GPFR	3	0.2079 ± 0.0000	0.9478 ± 0.0000	11.08 ± 0.05	3	0.2040 ± 0.0000	0.9490 ± 0.0000	10.82 ± 0.03	3	0.4182 ± 0.0000	0.7608 ± 0.0048	24.24 ± 0.91
	5	0.1628 ± 0.0122	0.9694 ± 0.0035	30.42 ± 5.95	5	0.1788 ± 0.0168	0.9622 ± 0.0052	29.98 ± 3.94	5	0.3409 ± 0.0039	0.8012 ± 0.0362	43.64 ± 12.66
	10	0.1270 ± 0.0175	0.9818 ± 0.0041	59.88 ± 10.65	10	0.1400 ± 0.0192	0.9749 ± 0.0046	46.27 ± 5.30	10	0.3236 ± 0.0105	0.8437 ± 0.0082	163.44 ± 29.12
	15	0.1156 ± 0.0172	0.9813 ± 0.0046	84.20 ± 16.58	15	0.1331 ± 0.0096	0.9778 ± 0.0025	62.35 ± 10.25	15	0.3123 ± 0.0096	0.8523 ± 0.0061	225.27 ± 48.88
	20	0.1145 ± 0.0182	0.9856 ± 0.0028	97.48 ± 20.72	20	0.1320 ± 0.0076	0.9789 ± 0.0027	78.00 ± 12.80	20	0.3026 ± 0.0092	0.8631 ± 0.0080	278.36 ± 45.86
	30	0.1013 ± 0.0044	0.9865 ± 0.0017	99.74 ± 17.71	30	0.1164 ± 0.0037	0.9817 ± 0.0019	115.66 ± 32.79	30	0.3031 ± 0.0083	0.8631 ± 0.0067	453.36 ± 68.26
	50	0.0999 ± 0.0031	0.9871 ± 0.0005	150.72 ± 34.99	50	0.1120 ± 0.0054	0.9821 ± 0.0021	163.69 ± 45.61	50	0.3103 ± 0.0115	0.8636 ± 0.0115	612.39 ± 104.74
DPM-FR	14.10 ± 0.88	0.1148 ± 0.0173	0.9802 ± 0.0052	193.70 ± 6.00	13.10 ± 1.66	0.1181 ± 0.0081	0.9802 ± 0.0038	199.27 ± 13.30	28.90 ± 0.99	0.3245 ± 0.0167	0.8563 ± 0.0062	339.39 ± 83.95
DPM-GP	12.80 ± 1.99	0.9388 ± 0.0015	-0.0212 ± 0.0039	95.48 ± 18.70	12.30 ± 1.77	0.8953 ± 0.0020	0.0514 ± 0.0042	117.88 ± 19.19	26.30 ± 2.54	0.6767 ± 0.0009	0.4384 ± 0.0008	507.16 ± 173.08
DPM-GPFR	13.20 ± 1.75	0.0999 ± 0.0026	0.9852 ± 0.0027	231.21 ± 84.37	12.00 ± 2.00	0.1259 ± 0.0104	0.9804 ± 0.0027	270.55 ± 76.12	27.40 ± 1.17	0.2999 ± 0.0091	0.8685 ± 0.0086	269.91 ± 100.67
DPM-GPFR-VT	14.50 ± 1.51	0.1097 ± 0.0113	0.9856 ± 0.0010	283.83 ± 15.23	11.50 ± 1.51	0.1269 ± 0.0098	0.9797 ± 0.0027	275.52 ± 72.67	22.10 ± 1.79	0.3001 ± 0.0061	0.8671 ± 0.0052	381.06 ± 27.72
DPM-GPFR$_{\alpha_0}$	13.70 ± 1.57	0.1037 ± 0.0117	0.9842 ± 0.0040	243.14 ± 60.13	12.80 ± 2.10	0.1188 ± 0.0108	0.9814 ± 0.0009	205.90 ± 103.49	27.00 ± 0.67	0.3017 ± 0.0066	0.8651 ± 0.0107	297.39 ± 20.51
PYP-GPFR	14.10 ± 1.79	0.1014 ± 0.0044	0.9861 ± 0.0014	242.89 ± 56.78	12.94 ± 1.90	0.1221 ± 0.0119	0.9802 ± 0.0025	270.13 ± 49.39	26.96 ± 1.23	0.2992 ± 0.0079	0.8651 ± 0.0092	305.14 ± 27.36



(a) Ground-truth mean functions (in red) and learned mean functions by DPM-GPFR (in blue).



(b) Prediction results of DPM-GPFR on $\mathcal{S}_{10}$. Black solid dots are known samples, colored solid dots are predictions, and hollow dots are ground-truth values.

Fig. 3. Illustrations of learned mean functions and prediction results of DPM-GPFR on $\mathcal{S}_{10}$.

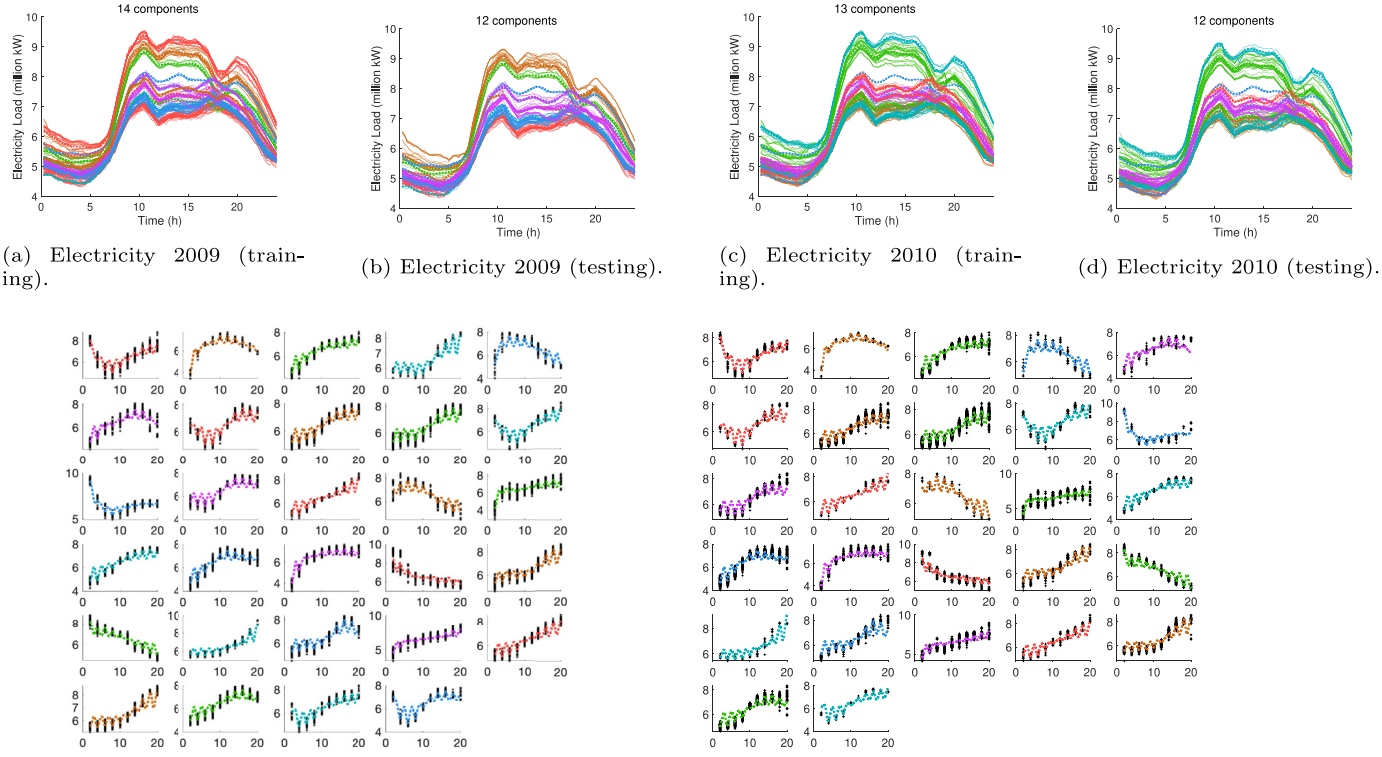
thus running mix-GPFR with different $\hat{K}$ then conducting model selection via cross-validation would be even more time-consuming. However, a single run of DPM-GPFR can successfully infer the appropriate $\hat{K}$ automatically. The performances of mix-GP on real-world datasets are almost the same with varying $\hat{K}$, and the results are similar to a single GP. The reason is that mix-GP fails to reveal fine structures underlying the data and model the evolving trends of curves. On the other hand, DPM-GPFR and its variants successfully cluster the curves into various clusters. Furthermore, DPM-GPFR related models almost always achieve the best results on all datasets, which demonstrates the effectiveness of DPM-GPFR.

We point out that introducing potentially infinite components is not the only reason that DPM-GPFR and its variants outperform competing methods. From Table 3, we observe that the results of mix-GPFR are very unstable. In mix-GPFR, the coefficients of B-spline mean functions are learned by optimization methods, which may lead to severe overfitting and are highly dependent on the initialization. In DPM-GPFR, we view them as random variables and develop a fully Bayesian treatment to infer the uncertainty over these parameters. This explains why the standard deviations of mix-GPFR on synthetic datasets are usually high. Besides, suppose that DPM-GPFR selects $\hat{K}$ components, if we run mix-GPFR with $\hat{K}$, then we may still obtain worse performances. The reason is, in mix-GPFR, the components are equal and symmetric, while in DPM-GPFR, the prior probability of these components is determined by a stick-breaking process and thus not symmetric. From the Chinese restaurant process perspective, the components generated by a Dirichlet process enjoy the self-reinforcing property, *i.e.*, the components are size-biased and a large component tends to be even larger. This property is desirable for real-world applications. Take the electricity load dataset, for example, most of the daily electricity load curves follow a similar trend for normal days, which forms the largest component. The same deduction also applies to the remaining curves, which may correspond to the electricity consumption on holidays and weekends.

In Fig. 3(a), we illustrate the learned mean functions and ground-truth mean functions in $\mathcal{S}_{10}$. We observe that DPM-GPFR can learn the mean functions accurately. For component 10, the mean function is relatively complicated, and the DPM-GPFR further splits this component into 3 components. These 3 learned

Table 5
Average RMSEs (Rooted Mean Square Errors) of DPM-GPFR on synthetic and real-world datasets with different prediction strategies.

Dataset	MAP spline	Spline	MAP	Fused
$\mathcal{S}_3$	0.5190 ± 0.0073	0.5172 ± 0.0042	0.4822 ± 0.0094	0.4798 ± 0.0050
$\mathcal{S}_4$	0.5692 ± 0.0051	0.5644 ± 0.0034	0.5382 ± 0.0054	0.5333 ± 0.0039
$\mathcal{S}_5$	0.6176 ± 0.0042	0.6154 ± 0.0025	0.5860 ± 0.0039	0.5839 ± 0.0022
$\mathcal{S}_6$	0.6141 ± 0.0070	0.6112 ± 0.0042	0.6078 ± 0.0074	0.6048 ± 0.0044
$\mathcal{S}_7$	0.6702 ± 0.0086	0.6684 ± 0.0048	0.6492 ± 0.0093	0.6475 ± 0.0053
$\mathcal{S}_8$	0.6664 ± 0.0054	0.6661 ± 0.0039	0.6460 ± 0.0059	0.6458 ± 0.0043
$\mathcal{S}_9$	0.6602 ± 0.0105	0.6574 ± 0.0102	0.6396 ± 0.0114	0.6369 ± 0.0107
$\mathcal{S}_{10}$	0.6726 ± 0.0033	0.6718 ± 0.0027	0.6405 ± 0.0032	0.6398 ± 0.0028
Electricity 2009	0.1191 ± 0.0114	0.1130 ± 0.0126	0.1119 ± 0.0095	0.1064 ± 0.0107
Electricity 2010	0.1343 ± 0.0092	0.1308 ± 0.0100	0.1247 ± 0.0093	0.1217 ± 0.0104
Drosophila	0.3352 ± 0.0158	0.3088 ± 0.0107	0.3264 ± 0.0148	0.3013 ± 0.0096



(a) Electricity 2009 (training). (b) Electricity 2009 (testing). (c) Electricity 2010 (training). (d) Electricity 2010 (testing).

(e) Drosophila (training). (f) Drosophila (testing).

Fig. 4. Clustering results of DPM-GPFR on realworld datasets. Curves of different clusters are in different colors.

mean functions share a similar global trend but have different local variations. For each testing curve in $\mathcal{S}_{10}$, we further show the prediction result in Fig. 3(b). This figure reveals why the DPM-GPFR model outperforms competing methods: DPM-GPFR successfully models evolving patterns in training curves, and in the prediction phase DPM-GPFR can adaptively choose the corresponding pattern based on known samples.

We report the performances of different prediction strategies on real-world datasets in Table 5. We note that fused prediction is consistently better than other strategies. If we ignore the structured noises and only use the spline functions to predict, the performances will drop. This observation confirms that structural noise plays an important role in modeling the data. Besides, the MAP prediction strategy is also worse compared with fused prediction, which shows the necessity of integrating out the uncertainty of latent indicators.

4.4. Clustering analysis

The clustering performances on synthetic datasets are reported in Table 6. Since we have ground-truth labels on synthetic datasets, we calculate the GCAR, EMD-I, EMD-II and ARI between the results and ground-truth labels and report the estimated number of components $\hat{K}$ on both training sets and testing sets. DPM-GPFR and its variants almost always achieve perfect GCAR and EMD-I, which means they can distinguish curves generated by different sources successfully. Usually, the estimated number of components $\hat{K}$ of DPM-GPFR is larger than K, thus it reveals the finer structure of the data. The results of mix-GP and DPM-GP are not good, which demonstrates the importance of using B-spline mean functions to capture the trend. As we increase K, the performances of mix-GPFR drops significantly, and MOHGP even leads to only one component on $\mathcal{S}_8$ and $\mathcal{S}_{10}$, while the performances of DPM-GPFR related models are stable regardless of K.

Table 6
Evaluation of clustering performances of the proposed and competing methods on synthetic datasets $\mathcal{S}_3$, $\mathcal{S}_5$, $\mathcal{S}_8$ and $\mathcal{S}_{10}$.

Method	Phase	$\mathcal{S}_3$					$\mathcal{S}_5$				
		$\hat{K}$	GCAR	EMD-I	EMD-II	ARI	$\hat{K}$	GCAR	EMD-I	EMD-II	ARI
mix-GP	train	3.00 ± 0.00	0.6800 ± 0.0422	0.3721 ± 0.0310	0.3093 ± 0.0963	0.3108 ± 0.0086	5.00 ± 0.00	0.8280 ± 0.0676	0.2018 ± 0.0658	0.1455 ± 0.0242	0.6751 ± 0.0594
mix-GP	test	3.00 ± 0.00	0.6467 ± 0.0422	0.4186 ± 0.0238	0.2893 ± 0.0787	0.2932 ± 0.0267	5.00 ± 0.00	0.6500 ± 0.0483	0.3942 ± 0.0358	0.2876 ± 0.0425	0.4168 ± 0.0428
mix-GPFR	train	3.00 ± 0.00	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	1.0000 ± 0.0000	5.00 ± 0.00	0.9800 ± 0.0632	0.0200 ± 0.0632	0.0019 ± 0.0060	0.9763 ± 0.0749
mix-GPFR	test	3.00 ± 0.00	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	1.0000 ± 0.0000	5.00 ± 0.00	0.9800 ± 0.0632	0.0200 ± 0.0632	0.0000 ± 0.0000	0.9768 ± 0.0734
MOHGP	train	3.10 ± 0.32	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0152 ± 0.0480	0.9877 ± 0.0389	2.20 ± 1.99	0.4200 ± 0.3584	0.5800 ± 0.3584	0.0084 ± 0.0266	0.2719 ± 0.4412
DPM-FR	train	4.70 ± 1.06	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0922 ± 0.0927	0.9247 ± 0.0774	7.60 ± 1.51	1.0000 ± 0.0000	0.0000 ± 0.0000	0.1140 ± 0.0587	0.9213 ± 0.0423
DPM-FR	test	3.70 ± 1.16	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0607 ± 0.0982	0.9479 ± 0.0844	5.90 ± 1.29	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0636 ± 0.0764	0.9538 ± 0.0566
DPM-GP	train	4.80 ± 1.14	0.7183 ± 0.0166	0.3349 ± 0.0162	0.4931 ± 0.0725	0.2899 ± 0.0365	8.40 ± 1.51	0.8260 ± 0.0306	0.2101 ± 0.0373	0.3869 ± 0.0734	0.5348 ± 0.0510
DPM-GP	test	4.10 ± 0.88	0.7000 ± 0.0351	0.3639 ± 0.0481	0.4040 ± 0.0937	0.2929 ± 0.0696	6.70 ± 0.82	0.7160 ± 0.0440	0.3262 ± 0.0534	0.3750 ± 0.0493	0.4522 ± 0.0760
DPM-GPFR	train	3.50 ± 0.85	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0395 ± 0.0756	0.9678 ± 0.0618	6.60 ± 1.07	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0763 ± 0.0503	0.9481 ± 0.0346
DPM-GPFR	test	3.20 ± 0.42	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0167 ± 0.0368	0.9862 ± 0.0306	5.80 ± 0.79	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0396 ± 0.0435	0.9721 ± 0.0310
DPM-GPFR-VT	train	3.70 ± 0.82	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0338 ± 0.0487	0.9730 ± 0.0392	5.60 ± 0.84	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0224 ± 0.0414	0.9848 ± 0.0284
DPM-GPFR-VT	test	3.10 ± 0.32	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0060 ± 0.0190	0.9951 ± 0.0156	5.00 ± 0.00	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	1.0000 ± 0.0000
DPM-GPFR$_{\alpha_0}$	train	4.10 ± 0.88	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0450 ± 0.0567	0.9640 ± 0.0467	6.60 ± 1.07	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0784 ± 0.0529	0.9466 ± 0.0366
DPM-GPFR$_{\alpha_0}$	test	3.20 ± 0.63	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0207 ± 0.0654	0.9822 ± 0.0562	5.70 ± 0.82	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0296 ± 0.0345	0.9793 ± 0.0242
PYP-GPFR	train	4.00 ± 1.15	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0485 ± 0.0649	0.9610 ± 0.0528	6.60 ± 1.43	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0675 ± 0.0608	0.9539 ± 0.0417
PYP-GPFR	test	3.20 ± 0.42	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0267 ± 0.0576	0.9776 ± 0.0486	5.80 ± 0.79	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0352 ± 0.0372	0.9753 ± 0.0264
Method	Phase	$\mathcal{S}_8$					$\mathcal{S}_{10}$				
		$\hat{K}$	GCAR	EMD-I	EMD-II	ARI	$\hat{K}$	GCAR	EMD-I	EMD-II	ARI
mix-GP	train	8.00 ± 0.00	0.5769 ± 0.0221	0.4744 ± 0.0197	0.3134 ± 0.0477	0.4618 ± 0.0203	10.00 ± 0.00	0.6555 ± 0.0400	0.3887 ± 0.0468	0.2909 ± 0.0442	0.5185 ± 0.0315
mix-GP	test	8.00 ± 0.00	0.5562 ± 0.0278	0.5057 ± 0.0347	0.3622 ± 0.0536	0.3702 ± 0.0256	10.00 ± 0.00	0.5220 ± 0.0319	0.5361 ± 0.0224	0.4438 ± 0.0365	0.3234 ± 0.0214
mix-GPFR	train	8.00 ± 0.00	0.9625 ± 0.0604	0.0375 ± 0.0604	0.0046 ± 0.0080	0.9570 ± 0.0693	10.00 ± 0.00	0.9300 ± 0.0675	0.0700 ± 0.0675	0.0186 ± 0.0198	0.9164 ± 0.0780
mix-GPFR	test	8.00 ± 0.00	0.9625 ± 0.0604	0.0375 ± 0.0604	0.0000 ± 0.0000	0.9581 ± 0.0675	10.00 ± 0.00	0.9300 ± 0.0675	0.0700 ± 0.0675	0.0084 ± 0.0158	0.9189 ± 0.0761
MOHGP	train	1.00 ± 0.00	0.1250 ± 0.0000	0.8750 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	1.00 ± 0.00	0.1000 ± 0.0000	0.9000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000
DPM-FR	train	11.00 ± 2.31	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0932 ± 0.0723	0.9403 ± 0.0476	13.50 ± 1.78	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0878 ± 0.0567	0.9458 ± 0.0372
DPM-FR	test	9.90 ± 1.73	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0787 ± 0.0714	0.9471 ± 0.0493	12.10 ± 1.45	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0750 ± 0.0538	0.9515 ± 0.0369
DPM-GP	train	10.80 ± 2.20	0.6288 ± 0.0434	0.4077 ± 0.0330	0.4311 ± 0.0741	0.4298 ± 0.0391	13.30 ± 2.45	0.7220 ± 0.0266	0.3145 ± 0.0271	0.3341 ± 0.0792	0.5329 ± 0.0388
DPM-GP	test	9.50 ± 1.90	0.5850 ± 0.0275	0.4603 ± 0.0263	0.4442 ± 0.0680	0.3503 ± 0.0373	10.80 ± 1.81	0.5440 ± 0.0151	0.5105 ± 0.0180	0.4474 ± 0.0681	0.3450 ± 0.0364
DPM-GPFR	train	9.30 ± 1.16	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0498 ± 0.0535	0.9688 ± 0.0338	12.00 ± 1.25	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0446 ± 0.0253	0.9733 ± 0.0155
DPM-GPFR	test	9.10 ± 1.29	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0410 ± 0.0504	0.9730 ± 0.0334	10.60 ± 0.70	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0206 ± 0.0248	0.9871 ± 0.0156
DPM-GPFR-VT	train	9.20 ± 1.03	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0357 ± 0.0358	0.9780 ± 0.0222	11.50 ± 1.35	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0264 ± 0.0285	0.9843 ± 0.0172
DPM-GPFR-VT	test	8.90 ± 0.88	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0318 ± 0.0343	0.9794 ± 0.0224	10.40 ± 0.70	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0124 ± 0.0222	0.9922 ± 0.0140
DPM-GPFR$_{\alpha_0}$	train	9.60 ± 1.43	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0582 ± 0.0565	0.9634 ± 0.0362	11.80 ± 1.23	0.9900 ± 0.0316	0.0100 ± 0.0316	0.0434 ± 0.0222	0.9635 ± 0.0325
DPM-GPFR$_{\alpha_0}$	test	9.40 ± 1.43	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0590 ± 0.0653	0.9606 ± 0.0447	10.90 ± 0.88	0.9900 ± 0.0316	0.0100 ± 0.0316	0.0364 ± 0.0299	0.9660 ± 0.0361
PYP-GPFR	train	9.80 ± 1.32	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0572 ± 0.0393	0.9645 ± 0.0248	12.10 ± 1.29	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0467 ± 0.0333	0.9719 ± 0.0206
PYP-GPFR	test	9.10 ± 1.29	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0348 ± 0.0402	0.9774 ± 0.0265	11.30 ± 0.95	1.0000 ± 0.0000	0.0000 ± 0.0000	0.0358 ± 0.0312	0.9774 ± 0.0199

Table 7
Average RMSEs (Rooted Mean Square Errors) of DPM-GPFR on real-world datasets with $\texttt{max\_K} \in \{10, 20, 30, 50\}$, $\alpha_0 \in \{0.01, 0.10, 0.50, 1.00, 3.00, 5.00, 10.00\}$ and $D = 30$.

max_K	α_0	Electricity 2009			Electricity 2010			Drosophila		
		$\hat{K}$	RMSE	Time	$\hat{K}$	RMSE	Time	$\hat{K}$	RMSE	Time
10	0.01	9.4 ± 0.7	0.1151 ± 0.0135	50.3 ± 14.39	8.9 ± 0.88	0.1426 ± 0.005	59.17 ± 9.49	10.00 ± 0.00	0.3210 ± 0.0143	67.28 ± 30.66
	0.10	9.1 ± 0.74	0.1282 ± 0.0168	59.84 ± 5.45	8.7 ± 0.67	0.1418 ± 0.0082	65.99 ± 6.09	10.00 ± 0.00	0.3254 ± 0.0113	81.34 ± 37.20
	0.50	8.9 ± 0.88	0.1333 ± 0.0219	62.15 ± 4.99	9.2 ± 0.79	0.1425 ± 0.0035	63.73 ± 8.56	10.00 ± 0.00	0.3187 ± 0.0123	69.63 ± 18.40
	1.00	9.0 ± 1.05	0.1336 ± 0.0121	59.12 ± 8.62	8.6 ± 1.07	0.1445 ± 0.0179	63.81 ± 6.54	10.00 ± 0.00	0.3216 ± 0.0111	64.73 ± 29.25
	3.00	8.5 ± 0.53	0.1287 ± 0.0156	61.68 ± 6.55	8.9 ± 0.57	0.14 ± 0.0089	63.12 ± 6.42	10.00 ± 0.00	0.3223 ± 0.0094	69.87 ± 26.21
	5.00	9.5 ± 0.71	0.1291 ± 0.0172	56.78 ± 5.38	8.6 ± 0.7	0.1423 ± 0.0175	61.63 ± 14.21	10.00 ± 0.00	0.3211 ± 0.0122	72.87 ± 23.62
	10.00	9.0 ± 0.67	0.1196 ± 0.0155	63.14 ± 4.9	8.8 ± 0.79	0.1363 ± 0.0098	61.9 ± 8.97	10.00 ± 0.00	0.3223 ± 0.0107	78.07 ± 53.07
20	0.01	12.5 ± 1.51	0.1067 ± 0.0104	136.75 ± 36.91	11.2 ± 1.32	0.132 ± 0.009	153.95 ± 14.83	20.00 ± 0.00	0.3008 ± 0.0063	163.61 ± 69.87
	0.10	12.3 ± 2.21	0.1126 ± 0.0161	149.51 ± 12.13	11.6 ± 1.35	0.126 ± 0.0087	127.15 ± 44.72	20.00 ± 0.00	0.3027 ± 0.0104	168.45 ± 57.52
	0.50	12.2 ± 1.14	0.1088 ± 0.0138	149.17 ± 6.74	12.3 ± 1.16	0.1228 ± 0.0098	140.67 ± 31.46	19.80 ± 0.42	0.3018 ± 0.0054	149.23 ± 63.47
	1.00	12.3 ± 1.42	0.1085 ± 0.0151	151.95 ± 8.67	10.9 ± 1.66	0.1324 ± 0.0114	142.47 ± 35.05	19.80 ± 0.42	0.3008 ± 0.0075	142.27 ± 54.14
	3.00	11.6 ± 0.97	0.109 ± 0.0108	152.18 ± 8.93	12.0 ± 1.7	0.1245 ± 0.0114	109.63 ± 47.35	19.90 ± 0.32	0.2986 ± 0.0067	141.70 ± 54.61
	5.00	11.8 ± 1.69	0.1121 ± 0.0134	132.52 ± 43.52	12.1 ± 1.2	0.1207 ± 0.0062	134.29 ± 45.8	20.00 ± 0.00	0.3054 ± 0.0077	204.90 ± 71.42
	10.00	12.4 ± 1.35	0.1127 ± 0.0161	137.57 ± 35.45	12.2 ± 1.4	0.1294 ± 0.0114	128.27 ± 47.54	19.90 ± 0.32	0.3004 ± 0.0051	145.23 ± 60.24
30	0.01	14.2 ± 1.75	0.1032 ± 0.012	244.18 ± 69.59	12.1 ± 1.79	0.1232 ± 0.0108	310.13 ± 12.12	26.90 ± 1.20	0.3034 ± 0.0085	261.85 ± 85.62
	0.10	13.6 ± 1.78	0.1038 ± 0.0126	270.46 ± 13.0	12.6 ± 2.22	0.1205 ± 0.0082	317.46 ± 39.56	26.90 ± 1.45	0.2988 ± 0.0074	238.30 ± 69.73
	0.50	13.3 ± 1.64	0.103 ± 0.0035	221.36 ± 92.12	12.3 ± 2.06	0.1209 ± 0.0132	264.8 ± 73.87	26.60 ± 1.35	0.2995 ± 0.0077	242.02 ± 98.26
	1.00	13.2 ± 1.75	0.0999 ± 0.0026	231.21 ± 84.37	12.0 ± 2.0	0.1259 ± 0.0104	270.55 ± 76.12	27.40 ± 1.17	0.2999 ± 0.0091	269.91 ± 100.67
	3.00	14.1 ± 1.6	0.1001 ± 0.0033	260.32 ± 10.75	12.1 ± 1.6	0.1219 ± 0.009	270.34 ± 16.39	26.70 ± 1.42	0.3047 ± 0.0109	278.03 ± 119.32
	5.00	14.0 ± 1.89	0.1 ± 0.0031	256.18 ± 13.28	12.8 ± 1.87	0.1212 ± 0.0105	238.6 ± 67.13	27.30 ± 1.34	0.2980 ± 0.0081	273.82 ± 86.37
	10.00	13.6 ± 1.17	0.1037 ± 0.0113	220.61 ± 80.51	13.6 ± 1.26	0.1193 ± 0.0117	257.82 ± 14.36	26.80 ± 0.92	0.2993 ± 0.0083	260.36 ± 79.86
50	0.01	13.8 ± 1.4	0.1016 ± 0.0052	370.66 ± 188.26	14.2 ± 2.04	0.1136 ± 0.0065	362.06 ± 134.7	36.40 ± 1.35	0.3001 ± 0.0083	446.10 ± 139.67
	0.10	14.7 ± 1.7	0.1032 ± 0.0066	470.84 ± 34.38	14.4 ± 1.58	0.1123 ± 0.0055	403.04 ± 158.12	34.40 ± 2.27	0.3022 ± 0.0085	411.15 ± 146.06
	0.50	15.2 ± 1.32	0.1015 ± 0.0043	487.85 ± 12.75	14.4 ± 1.78	0.115 ± 0.0085	430.96 ± 127.77	34.70 ± 2.36	0.3026 ± 0.0107	509.99 ± 188.90
	1.00	14.4 ± 2.01	0.1003 ± 0.0038	404.57 ± 158.41	14.0 ± 2.31	0.1153 ± 0.0067	421.28 ± 107.3	34.80 ± 1.93	0.3037 ± 0.0083	637.26 ± 137.92
	3.00	14.8 ± 1.03	0.1014 ± 0.0019	485.51 ± 3.37	13.6 ± 1.78	0.1228 ± 0.0132	397.1 ± 151.77	34.80 ± 1.14	0.3034 ± 0.0091	569.81 ± 227.54
	5.00	15.2 ± 2.25	0.1032 ± 0.0049	401.85 ± 159.71	14.2 ± 1.4	0.1172 ± 0.0083	361.43 ± 178.15	34.40 ± 2.84	0.3009 ± 0.0083	506.12 ± 196.20
	10.00	15.0 ± 1.63	0.1002 ± 0.0054	448.33 ± 121.63	15.0 ± 1.33	0.1119 ± 0.0053	416.27 ± 117.66	35.90 ± 1.52	0.3058 ± 0.0077	440.73 ± 138.04

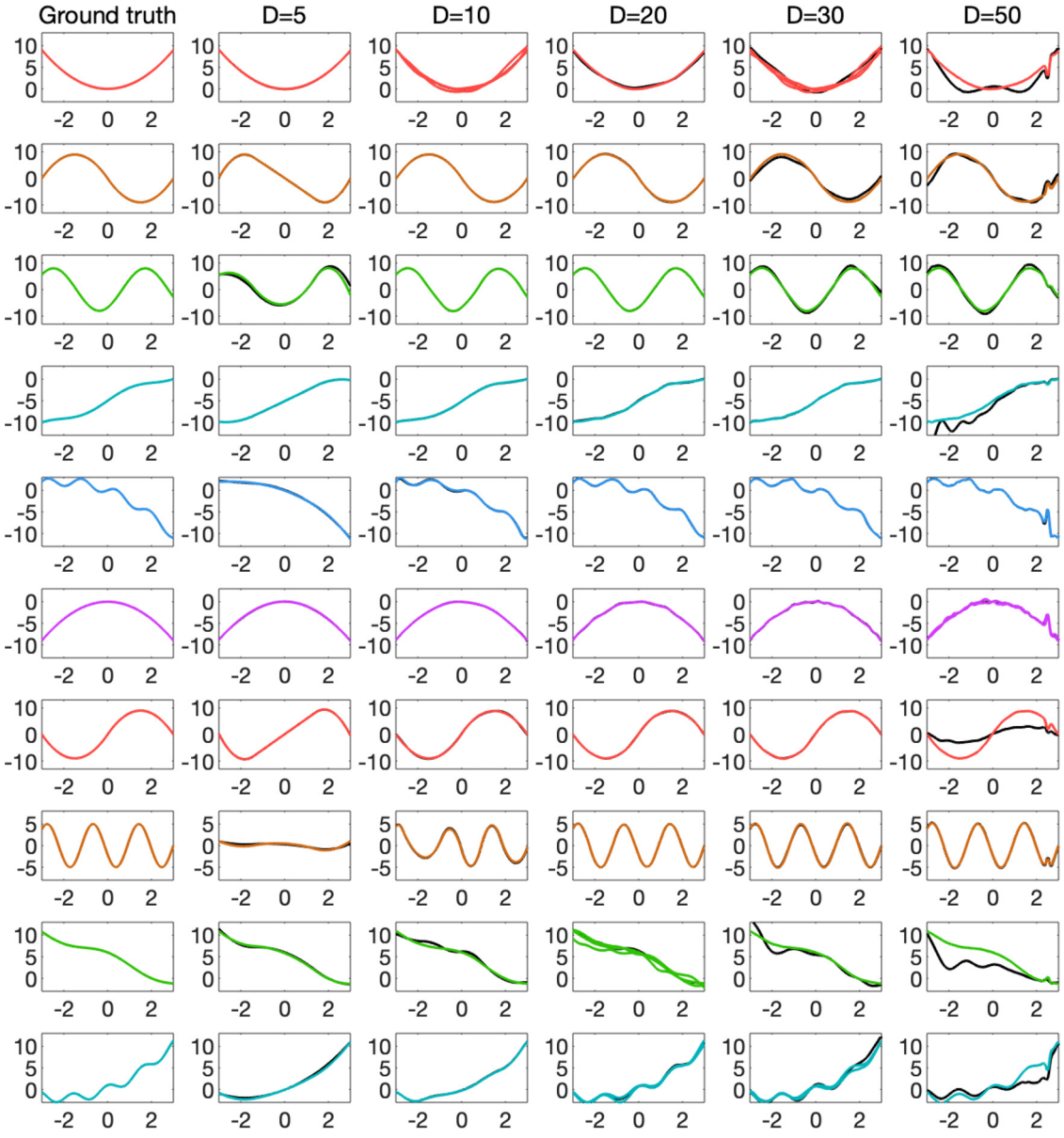


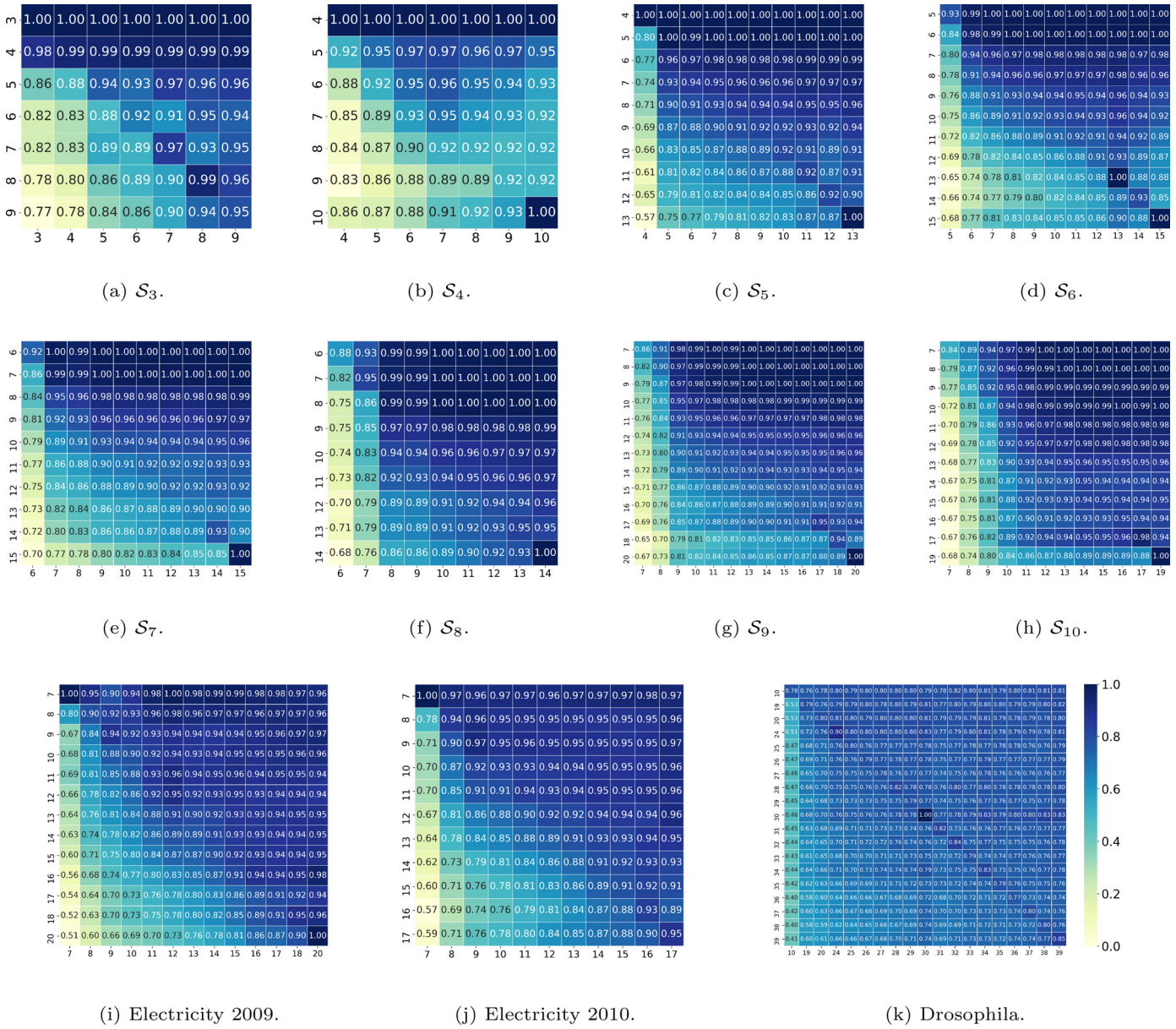
Fig. 5. Learned mean functions of DPM-GPFR and mix-GPFR on $\mathcal{S}_{10}$ under different D. The results of mix-GPFR are in black, while the results of DPM-GPFR are in color.

Since we do not have access to ground-truth clustering labels on real-world datasets, we visualize the clustering results in Fig. 4. We plot the training curves and testing curves together with learned mean functions. Curves belonging to different components are in different colors. This figure further reflects the self-reinforcing property, *i.e.*, the sizes of clusters are highly imbalanced. The learned structures are very fine and accurately capture the evolving law.

4.5. Discussions on variants of the model

From Tables 3, 4 and 6, we observe that the improvements brought by introducing the Dirichlet process prior to the mixture of GPs over a single Gaussian process are minor. Although DPM-GP has a mixture structure and divides the curves into clusters, the clustering and prediction results are not good. The reason is DPM-GP fails to model the data, especially capture the common evolving trends without B-spline mean functions. On the other hand, DPM-FR is much better than competing methods and achieves comparable performances with DPM-GPFR. If only "FR" or "GP" is used, each component is not as flexible as "GPFR", thus the model tends to use more components to describe the data. We conclude that both the functional regression part and the Gaussian process part are necessary to model the data, and using functional regression is more important. DPM-GPFR-VT has fewer components than DPM-GPFR and higher ARI, thus the variational tempering technique improves the clustering performance. Other variants have similar results compared with DPM-GPFR, and none of them consistently outperforms others.

From Tables 3 and 4, we find that although the number of parameters of DPM-GPFR and its variants are close, the time costs are significantly different. The main reason is that the prior of stick-breaking variables influences the distribution of mixing proportions significantly, as discussed in Section 3.3. Since the parameters α_0 and δ are automatically learned in DPM-GPFR$_{\alpha_0}$ and PYP-GPFR, the number of remaining components during the learning process may be different from DPM-GPFR, which influences the computational cost significantly. Unfortunately, since the learning process is data-dependent, we have no idea how many iterations are needed



(a) $\mathcal{S}_3$. (b) $\mathcal{S}_4$. (c) $\mathcal{S}_5$. (d) $\mathcal{S}_6$.

(e) $\mathcal{S}_7$. (f) $\mathcal{S}_8$. (g) $\mathcal{S}_9$. (h) $\mathcal{S}_{10}$.

(i) Electricity 2009. (j) Electricity 2010. (k) Drosophila.

Fig. 6. Visualization of AGCARs (Average Generalized Classification Accuracy Rates) of clustering results with different $\hat{K}$. Large values are dark, and small values are light.

for convergence, nor do we know how many components are remained. Therefore, we can not easily conclude which method is the fastest, and their time costs highly depend on the specific problem.

4.6. Sensitivity analysis of hyper-parameters

We fix $D = 30$, and vary α_0 in $\{0.1, 0.25, 0.5, 1, 3, 5, 10\}$, `max_K` in $\{10, 20, 30, 50\}$. The results are shown in Table 7. We find different α_0 leads to similar RMSEs, thus the performances are not sensitive to α_0. In particular, given `max_K`, the number of remaining components almost keeps invariant under different α_0 in all datasets. Intuitively, α_0 influences the number of components, and a larger α_0 may lead to more components. The observation here is contradictory to intuition. The reason is α_0 indeed influences the number of components *priorly*, but the final $\hat{K}$ is determined by both prior and *data*. Once we have abundant data, the role of prior becomes less important, thus we obtain similar results even α_0 varies severely. However, α_0 stills affects the time cost as discussed in Section 3.3, because the number of remaining components during the learning process is affected by α_0, especially at the first few EM iterations when prior plays a more important role than data.

On the other hand, the results are sensitive to `max_K`. Given α_0, as we increase `max_K`, there tend to be more components and the running times tend to be longer. The effect of α_0 on time cost is influenced by `max_K`, as indicated in Section 3.3. When `max_K` is too small, there are not enough components for mixing proportions to decay to 0, thus the running times under different α_0 are similar. When `max_K` is large enough, the effect of α_0 is more significant. Note that the original Dirichlet process assumes there are potentially infinite many components and `max_K` is only introduced to make the variational inference tractable. Therefore, a larger `max_K` may better retain the properties of the Dirichlet process, while a small `max_K` makes it more like a finite mixture model.

Finally, we fix $\alpha_0 = 1$, `max_K` $= 50$ and vary D in $\{5, 10, 20, 30, 50\}$. To further compare DPM-GPFR and mix-GPFR, we report the prediction performances of both methods under different D in Table 8, and show the learned mean functions on $\mathcal{S}_{10}$

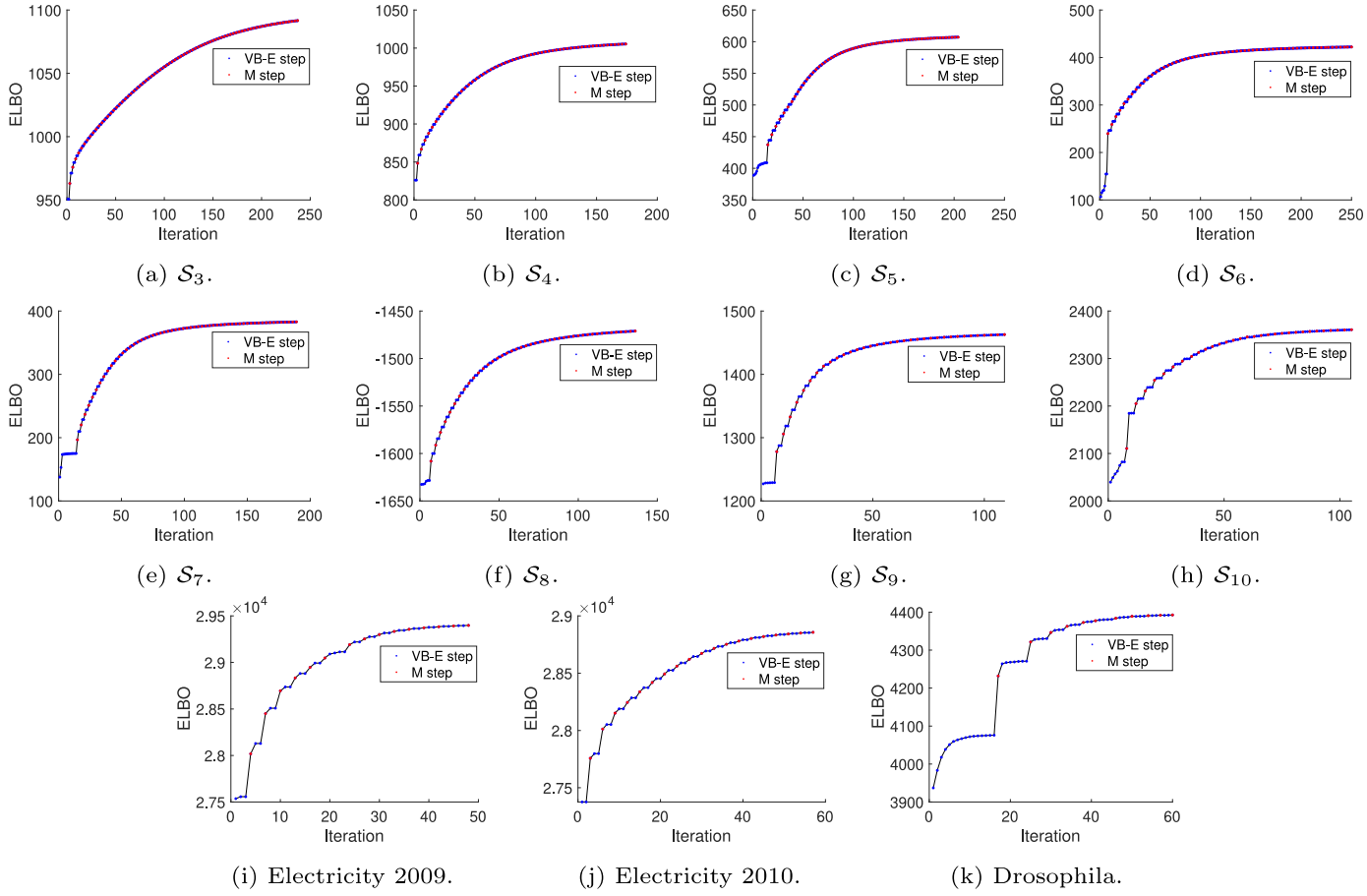


Fig. 7. Evidence lower bound (ELBO) v.s. iterations. ELBOs after an M-step are in red, and ELBOs after a VB iteration inside an E-step are in blue.

Table 8
Average RMSEs (Rooted Mean Square Errors) of DPM-GPFR on synthetic and real-world datasets with $\texttt{max\_K} = 30, \alpha_0 = 1.00$ and $D \in \{5, 10, 20, 30, 50\}$.

Dataset	Method\D	5	10	20	30	50
$\mathcal{S}_3$	mix-GPFR	1.4907 ± 0.0000	0.4877 ± 0.0000	0.4763 ± 0.0000	1.1378 ± 0.0000	1.3591 ± 0.0000
	DPM-GPFR	1.5363 ± 0.0011	0.4812 ± 0.0015	0.5314 ± 0.0024	0.5358 ± 0.0025	0.5378 ± 0.0010
$\mathcal{S}_5$	mix-GPFR	1.5637 ± 0.0610	0.6792 ± 0.2523	0.9544 ± 0.3057	0.9437 ± 0.1582	1.3133 ± 0.0967
	DPM-GPFR	1.2154 ± 0.0020	0.5982 ± 0.0022	0.5824 ± 0.0018	0.5865 ± 0.0034	0.5850 ± 0.0013
$\mathcal{S}_6$	mix-GPFR	1.3951 ± 0.2372	1.0330 ± 0.8764	0.8493 ± 0.2441	0.9587 ± 0.0696	0.9509 ± 0.1312
	DPM-GPFR	1.2202 ± 0.1101	0.6044 ± 0.0012	0.6032 ± 0.0037	0.6075 ± 0.0038	0.6068 ± 0.0108
$\mathcal{S}_7$	mix-GPFR	1.7462 ± 0.0695	0.7491 ± 0.2853	1.3249 ± 0.8260	1.3309 ± 0.3491	1.3574 ± 0.1303
	DPM-GPFR	1.2316 ± 0.0008	0.6575 ± 0.0026	0.6440 ± 0.0049	0.6512 ± 0.0040	0.6511 ± 0.0050
$\mathcal{S}_8$	mix-GPFR	2.0916 ± 0.3237	0.8365 ± 0.4668	0.8810 ± 0.4671	0.8627 ± 0.2220	1.1940 ± 0.5667
	DPM-GPFR	1.8314 ± 0.0143	0.6837 ± 0.0027	0.6478 ± 0.0045	0.6414 ± 0.0023	0.6413 ± 0.0022
$\mathcal{S}_9$	mix-GPFR	2.0088 ± 0.2849	1.1011 ± 0.4741	1.0828 ± 0.4047	0.9868 ± 0.3939	1.3796 ± 0.3269
	DPM-GPFR	1.6815 ± 0.2662	0.6937 ± 0.1379	0.6375 ± 0.0119	0.6757 ± 0.1411	0.6794 ± 0.1411
$\mathcal{S}_{10}$	mix-GPFR	1.7408 ± 0.3282	1.0823 ± 0.4284	1.2112 ± 0.4584	1.2715 ± 0.4892	1.3357 ± 0.3496
	DPM-GPFR	1.5298 ± 0.1298	0.6801 ± 0.0022	0.6765 ± 0.1176	0.6378 ± 0.0032	0.6875 ± 0.1476
Drosophilia	mix-GPFR	0.3651 ± 0.0363	0.3614 ± 0.0436	0.3630 ± 0.0447	0.3615 ± 0.0437	0.3593 ± 0.0433
	DPM-GPFR	0.3212 ± 0.0057	0.3016 ± 0.0059	0.2998 ± 0.0082	0.3065 ± 0.0095	0.3030 ± 0.0077
Electricity 2009	mix-GPFR	0.2749 ± 0.0240	0.2125 ± 0.0345	0.1795 ± 0.0429	0.1652 ± 0.0421	0.1677 ± 0.0391
	DPM-GPFR	0.2562 ± 0.0020	0.1777 ± 0.0090	0.1256 ± 0.0185	0.1014 ± 0.0023	0.1086 ± 0.0084
Electricity 2010	mix-GPFR	0.2838 ± 0.0227	0.2134 ± 0.0328	0.1832 ± 0.0428	0.1735 ± 0.0366	0.1752 ± 0.0370
	DPM-GPFR	0.2610 ± 0.0035	0.1686 ± 0.0047	0.1176 ± 0.0126	0.1194 ± 0.0095	0.1191 ± 0.0058

in Fig. 5. We can see that if D is too small, the B-spline functions are not flexible enough to fit the trend of curves thus leading to large prediction errors. As we increase D, the RMSE tends to decrease, and the running time tends to be longer. For DPM-GPFR, the improvements of the performances are marginal as long as D is large enough. Although we develop a fully Bayesian treatment of DPM-GPFR, in the prediction phase we use the MAP estimations of $\mathbf{b}_k$ instead of integrating out the uncertainty, thus the model may still suffer from over-fitting and the performance may drop slightly if D is too large. On the other hand, for mix-GPFR, the choice of D is very tricky and significantly influences the final results. From Fig. 5, we observe that DPM-GPFR is less likely to overfit than mix-GPFR, which confirms that treating B-spline coefficients as latent indicators and placing a multivariate Gaussian prior helps to enhance the robustness of the model. We set $D = 20$ on synthetic datasets and $D = 30$ on real-world datasets in our experiments,

which balances the trade-off between flexibility, running time, and the risk of over-fitting.

Due to randomness of initialization of latent variables and optimization procedures, multiple runs of DPM-GPFR on a given dataset may lead to different $\hat{K}$. Besides, under various parameter settings, the resulting $\hat{K}$ may also be different. Even two clustering results have the same $\hat{K}$, they may still be different. One natural question is, what is the relationship between these clustering results? Suppose that we have n_1 clustering results $c_1, \ldots, c_{n_1}$ with $\hat{K}_1$ and n_2 clustering results $c'_1, \ldots, c'_{n_2}$ with $\hat{K}_2$, we calculate

$$\text{AGCAR}(\hat{K}_1, \hat{K}_2) = \frac{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \text{GCAR}(c_i, c'_j)}{n_1 n_2}$$

for each possible combination of $\hat{K}_1, \hat{K}_2$. The results in Fig. 6 shows that when $\hat{K}_2 \geq \hat{K}_1$, $\text{AGCAR}(\hat{K}_1, \hat{K}_2)$ is close to 1, which means c'_j is nearly a refinement of c_i for each c_i, c'_j.

4.7. Effectiveness of the variational EM algorithm

Although there are convergences guarantees for the conventional EM algorithm [3,33,34], these theoretical results cannot be directly applied to the variational EM algorithm since the Q-function is calculated by an approximate posterior distribution. To see the convergence property of the variational EM algorithm, we plot the ELBO v.s. iterations in Fig. 7. We can see the ELBO increases with iterations and tends to be stable soon after several iterations, which empirically demonstrates that the proposed learning algorithm is effective for the DPM-GPFR model.

5. Conclusion and discussion

In this paper, we have proposed an infinite mixture of Gaussian process functional regression model (DPM-GPFR) based on the Dirichlet process, which can accurately capture trends information and characterize structured noises in functional datasets. The proposed method successfully solved the model selection problem in the mixture of Gaussian process functional regressions (mix-GPFR) model. Furthermore, the variational EM-algorithm for learning is derived in detail, and several variants are discussed. Extensive experimental results on both synthetic and real-world datasets demonstrate the effectiveness of the proposed model compared with competing methods. We also analyzed the clustering results and investigated the sensitivity of hyper-parameters of DPM-GPFR.

We point out several possible research directions. In DPM-GPFR, each component is a GPFR, and the mixture is developed with output space. As indicated in [9], introducing a mixture structure in the input space may further improve the performance. Besides, using MCMC sampling to approximate the Q-function in the EM algorithm [18,35] may improve the approximation accuracy compared with variational methods at the cost of longer running time. Furthermore, it is interesting to extend DPM-GPFR to multivariate cases and consider exogenous covariates.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under grant 2018AAA0100205.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.patcog.2022.109129

References

[1] C.K. Williams, C.E. Rasmussen, Gaussian Processes for Machine Learning, 2, MIT Press Cambridge, MA, 2006.
[2] R.C. Grande, T.J. Walsh, G. Chowdhary, S. Ferguson, J.P. How, Online regression for data with changepoints using gaussian processes and reusable models, IEEE Trans. Neural Netw. Learn. Syst. 28 (9) (2017) 2115–2128.
[3] H. Asheri, R. Hosseini, B.N. Araabi, A new em algorithm for flexibly tied GMMS with large number of components, Pattern Recognit. 114 (2021) 107836.
[4] J. Shi, B. Wang, R. Murray-Smith, D. Titterington, Gaussian process functional regression modeling for batch data, Biometrics 63 (3) (2007) 714–723.
[5] J. Shi, B. Wang, Curve prediction and clustering with mixtures of gaussian process functional regression models, Stat. Comput. 18 (3) (2008) 267–283.
[6] J.Q. Shi, T. Choi, Gaussian Process Regression Analysis for Functional Data, Chapman and Hall/CRC, 2011.
[7] D. Wu, J. Ma, A daem algorithm for mixtures of gaussian process functional regressions, in: Proceedings of the International Conference on Intelligent Computing, Springer, 2016, pp. 294–303.
[8] Z. Qiang, J. Luo, J. Ma, Curve clustering via the split learning of mixtures of gaussian processes, in: Proceedings of the IEEE 13th International Conference on Signal Processing, IEEE, 2016, pp. 1089–1094.
[9] D. Wu, J. Ma, A two-layer mixture model of gaussian process functional regressions and its mcmc em algorithm, IEEE Trans. Neural Netw. Learn. Syst. (99) (2018) 1–11.
[10] J.-L. Wang, J.-M. Chiou, H.-G. Müller, Functional data analysis, Annu. Rev. Stat. Appl. 3 (2016) 257–295.
[11] P. Kokoszka, M. Reimherr, Introduction to Functional Data Analysis, Chapman and Hall/CRC, 2017.
[12] M.I. Jordan, R.A. Jacobs, Hierarchical mixtures of experts and the em algorithm, Neural Comput. 6 (2) (1994) 181–214.
[13] Y.W. Teh, Dirichlet process, in: Encyclopedia of Machine Learning, Springer, 2010, pp. 280–287.
[14] S. Sun, X. Xu, Variational inference for infinite mixtures of Gaussian processes with applications to traffic flow prediction, IEEE Trans. Intell. Transp. Syst. 12 (2) (2011) 466–475.
[15] W. Zhang, L. Du, L. Li, X. Zhang, H. Liu, Infinite Bayesian one-class support vector machine based on dirichlet process mixture clustering, Pattern Recognit. 78 (2018) 56–78.
[16] D.M. Blei, A. Kucukelbir, J.D. McAuliffe, Variational inference: a review for statisticians, J. Am. Stat. Assoc. 112 (518) (2017) 859–877.
[17] G. McLachlan, T. Krishnan, The EM Algorithm and Extensions, volume 382, John Wiley & Sons, 2007.
[18] D. Wu, J. Ma, An effective em algorithm for mixtures of Gaussian processes via the MCMC sampling and approximation, Neurocomputing 331 (2019) 366–374.
[19] C. Liu, H.-C. Li, K. Fu, F. Zhang, M. Datcu, W.J. Emery, Bayesian estimation of generalized gamma mixture model based on variational em algorithm, Pattern Recognit. 87 (2019) 269–284.
[20] H. Robbins, An Empirical Bayes Approach to Statistics, University of California Press, 2020.
[21] J. Hensman, M. Rattray, N.D. Lawrence, Fast nonparametric clustering of structured time-series, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2) (2015) 383–393.
[22] A. Damianou, N. Lawrence, Deep gaussian processes, in: Proceedings of the Artificial Intelligence and Statistics, 2013, pp. 207–215.
[23] D.M. Blei, M.I. Jordan, et al., Variational inference for dirichlet process mixtures, Bayesian Anal. 1 (1) (2006) 121–143.
[24] M.J. Wainwright, M.I. Jordan, et al., Graphical models, exponential families, and variational inference, Found. Trends Mach. Learn. 1 (1–2) (2008) 1–305.
[25] J. Hensman, M. Rattray, N.D. Lawrence, Fast variational inference in the conjugate exponential family, in: Advances in Neural Information Processing Systems, 2012, pp. 2888–2896.
[26] H. Liu, Y.-S. Ong, X. Shen, J. Cai, When gaussian process meets big data: a review of scalable GPS, IEEE Trans. Neural Netw. Learn. Syst. 31 (11) (2020) 4405–4423.
[27] H. Liu, Y.-S. Ong, X. Jiang, X. Wang, Modulating scalable gaussian processes for expressive statistical learning, Pattern Recognit. 120 (2021) 108–121.
[28] S.P. Chatzis, Y. Demiris, Nonparametric mixtures of gaussian processes with power-law behavior, IEEE Trans. Neural Netw. Learn. Syst. 23 (12) (2012) 1862–1871.
[29] S. Mandt, J. McInerney, F. Abrol, R. Ranganath, D. Blei, Variational tempering, in: Artificial Intelligence and Statistics, 2016, pp. 704–712.
[30] A.T. Kalinka, K.M. Varga, D.T. Gerrard, S. Preibisch, D.L. Corcoran, J. Jarrells, U. Ohler, C.M. Bergman, P. Tomancak, Gene expression divergence recapitulates the developmental hourglass model, Nature 468 (7325) (2010) 811–814.
[31] Z. Chen, J. Ma, Y. Zhou, A precise hard-cut em algorithm for mixtures of gaussian processes, in: Proceedings of the International Conference on Intelligent Computing, Springer, 2014, pp. 68–75.
[32] J.M. Santos, M. Embrechts, On the use of the adjusted rand index as a metric for evaluating supervised classification, in: Proceedings of the International Conference on Artificial Neural Networks, Springer, 2009, pp. 175–184.

[33] J. Ma, S. Fu, On the correct convergence of the em algorithm for gaussian mixtures, Pattern Recognit. 38 (12) (2005) 2602–2611.
[34] J. Yu, C. Chaomurilige, M.-S. Yang, On convergence and parameter selection of the em and da-em algorithms for gaussian mixtures, Pattern Recognit. 77 (2018) 188–203.
[35] R.M. Neal, Markov chain sampling methods for dirichlet process mixture models, J. Comput. Graph. Stat. 9 (2) (2000) 249–265.

Tao Li received the B.S. degree in mathematics from University of Science and Technology of China, Hefei, China, in 2017. Currently he is pursuing the Ph.D. degree in applied mathematics from Peking University, Beijing, China. His research interests include machine learning, Bayesian inference, and neural networks.

Jinwen Ma received the M.S. degree in applied mathematics from Xi'an Jiaotong University in 1988 and the Ph.D. degree in probability theory and statistics from Nankai University in 1992. From July 1992 to November 1999, he was a lecturer or associate professor at the Department of Mathematics, Shantou University. From December 1999, he became a full professor at the Institute of Mathematics, Shantou University. From September 2001, he has joined the Department of Information Science at the School of Mathematical Sciences, Peking University, where he is currently a full professor and Ph.D. tutor. During 1995 and 2003, he also visited several times at the Department of Computer Science and Engineering, the Chinese University of Hong Kong as a Research Associate or Fellow. He also worked as Research Scientist at Amari Research Unit, RIKEN Brain Science Institute, Japan from September 2005 to August 2006. He has published over 100 academic papers on neural networks, pattern recognition, bioinformatics, and information theory.