Average Mean Functions Based EM Algorithm for Mixtures of Gaussian Processes

Tao Li[1], Xiao Luo[2], and Jinwen Ma[1(✉)]

[1] Department of Information and Computational Sciences, School of Mathematical Sciences and LMAM, Peking University, Beijing 100871, China
jwma@math.pku.edu.cn

[2] Department of Probability and Statistics, School of Mathematical Sciences and LMAM, Peking University, Beijing 100871, China

Abstract. The mixture of Gaussian process functional regressions (mix-GPFR) utilizes a linear combination of certain B-spline functions to fit the mean functions of Gaussian processes. Although mix-GPFR makes the mean functions active in the mixture of Gaussian processes, there are two limitations: (1). This linear combination approximation introduces many parameters and thus a heavy cost of computation is paid for learning these mean functions. (2). It is implicitly assumed that the mean functions of different components share the same degree of smoothness because there is a parameter controlling the smoothness globally. In order to get rid of these limitations, we propose a new kind of EM algorithm for mixtures of Gaussian processes with average mean functions from time-aligned batch trajectory or temporal data. In this way, the mean functions are iteratively updated according to the distributed trajectory data of the estimated Gaussian processes at each iteration of the EM algorithm and the effectiveness of this estimation is also theoretically analyzed. It is demonstrated by the experimental results on both synthetic and real-world datasets that the proposed method performs well.

1 Introduction

Gaussian process (GP) [1] is the dominant non-parametric Bayesian model and has been applied in many research fields [2–4]. In machine learning, a Gaussian process is determined by its mean function and covariance function. Usually, Gaussian processes are used for nonlinear regression or classification, but they can also be used to process batch datasets [5].

The mean function of Gaussian processes is important for processing batch datasets. Usually, the mean function is assumed to be zero. However, the zero mean function fails to model batch datasets with a shared global trend. Shi proposed the Gaussian Process Functional Regression (GPFR) model [5,6], which provides a feasible way to learn nonlinear mean functions. In GPFR, the mean function is assumed to be a linear combination of B-spline basis functions. However, GPFR fails to model heterogeneous/multi-modal data accurately [7]. In real

T. Mantoro et al. (Eds.): ICONIP 2021, CCIS 1516, pp. 549–557, 2021.
https://doi.org/10.1007/978-3-030-92307-5_64

applications, the curves may be generated from different signal sources, thus a single GPFR is not flexible enough to model all the curves. Shi *et al.* suggested introducing mixture structures on GPFRs (mix-GPFR) [7] to further enhance the model flexibility. In mix-GPFR, curves generated by a common signal source are regarded as a GPFR component. This model is naturally suitable for the curve clustering task. Recently, Wu and Ma [8] further extended mix-GPFR to a Two-layer Mixture of Gaussian Process Functional Regressions (TMGPFR) for modeling general stochastic processes from different sources.

The mix-GPFR model suffers from several problems. First, the number of B-spline basis functions D is difficult to set in practice. A large D leads to over-fitting, while a small D results in under-fitting, and the performances are usually frustrating in both circumstances. One can run the mix-GPFR with different D and choose the most proper D by trial and error, but this procedure is tedious and time-consuming. Second, different GPFR components in mix-GPFR share a common D. Therefore, when different components have different smoothness, it would be even more difficult to set a proper D, as illustrated in Sect. 3.1 and Fig. 1. Third, learning coefficients of B-spline basis functions in the M-step of the EM algorithm is both time-consuming and difficult since there are many local maximums and the optimization procedure often gets trapped in a local maximum as indicated [9].

To tackle these problems, we propose the mixture of Gaussian Processes with Average Mean Functions (mix-GPAVM). In the proposed method, the mean functions are estimated non-parametrically by averaging observations. Compared with mix-GPFR, this method is specifically designed for time-aligned datasets and it can adjust the smoothness in each component adaptively. Using average mean functions in the EM algorithm for learning mixtures of Gaussian processes is also more efficient than mix-GPFR. Experimental results on both synthetic and real-world datasets demonstrate the effectiveness of the proposed method.

2 Preliminaries

Suppose we are given a batch dataset $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^N$, each $\mathcal{D}_i = \{(x_{im}, y_{im})\}_{m=1}^{N_i}$ can be regarded as a sampled signal from an underlying function $y_i(x)$. Gaussian Process Functional Regression (GPFR) assumes that these functions $\{y_i(x)\}_{i=1}^N$ share a global mean function $\mu(x)$ and a structured noise $\tau(x)$. More specifically, Shi *et al.* [6] assume that $\mu(x)$ is a linear combination of D B-spline basis functions $\{\phi_j(x)\}_{j=1}^D$, and the noise term $\tau(x)$ is a Gaussian process:

$$y_i(x) = \mu(x) + \tau(x) \quad , \quad \mu(x) = \sum_{j=1}^{D} b_j \phi_j(x) \quad , \quad \tau(x) \sim \mathcal{GP}(0, c(x, x'; \boldsymbol{\theta})).$$

Here, $c(x, x'; \boldsymbol{\theta})$ is the covariance function of the Gaussian process and $\boldsymbol{\theta}$ is the parameter. More compactly, we can write this model as $y_i(x) \sim \mathcal{GPFR}(x; \mathbf{b}, \boldsymbol{\theta})$. Both $\boldsymbol{\theta} = [b_1, \cdots, b_D]^\intercal$ and the coefficients of B-spline basis functions $\mathbf{b}$ need to

be learned. We apply the maximum likelihood method to learn these parameters. Let $\mathbf{y}_i = [y_{i1}, \cdots, y_{iN_i}]^\intercal$, $\mathbf{x}_i = [x_{i1}, \cdots, x_{iN_i}]^\intercal$, $\boldsymbol{\Phi}_i = [\phi_j(x_{im})]_{N_i \times D}$, $\mathbf{C}_i = [c(x_{im}, c_{in}; \boldsymbol{\theta})]_{N_i \times N_i}$, then $\mathbf{y}_i|\mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\Phi}_i\mathbf{b}, \mathbf{C}_i)$. Therefore, the parameter learning boils down to maximizing

$$\sum_{i=1}^{N} \log \mathcal{N}(\mathbf{y}_i; \boldsymbol{\Phi}_i\mathbf{b}, \mathbf{C}_i) = \sum_{i=1}^{N} -\frac{1}{2}\log|\mathbf{C}_i| - \frac{1}{2}(\mathbf{y}_i - \boldsymbol{\Phi}_i\mathbf{b})^\intercal \mathbf{C}_i^{-1}(\mathbf{y}_i - \boldsymbol{\Phi}_i\mathbf{b}) + \texttt{const}.$$

Suppose we are given an $(N+1)$-th curve $\mathcal{D}_{N+1} = \{(x_{N+1,m}, y_{N+1,m})\}_{m=1}^{N_*}$ and need to predict the response at input x_*. According to the conditional property of multivariate Gaussian distribution [1,10], we predict the response as

$$\hat{y} = \boldsymbol{\phi}_*^\intercal \mathbf{b} + \mathbf{c}_* \mathbf{C}_{N+1}^{-1}(\mathbf{y}_{N+1} - \boldsymbol{\Phi}_{N+1}\mathbf{b}),$$
$$\boldsymbol{\phi}_* = [\phi_1(x_*), \cdots, \phi_D(x_*)]^\intercal, \mathbf{c}_* = [c(x_*, x_{N+1,1}; \boldsymbol{\theta}), \cdots, c(x_*, x_{N+1,N_*}; \boldsymbol{\theta})]^\intercal$$

Furthermore, to enhance the model flexibility and deal with the heterogeneity problem, a mixture of GPFR (mix-GFPR) was proposed [7]. In mix-GPFR, we assume that there are K GPFR components $\mathcal{GPFR}(x; \mathbf{b}_k, \boldsymbol{\theta}_k)$ and each $\mathcal{D}_i$ is generated from one of them given the latent indicator variable z_i,

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

The mix-GPFR model is usually learned by the EM algorithm [11]. The final prediction is given by a weighted sum of the results predicted by each component separately.

3 Proposed Approach

3.1 Time Aligned Batch Dataset and Average Mean Functions

In general, these signals may have different lengths, *i.e.*, $N_i \neq N_j$ for $i \neq j$. GPFR enables us to deal with such variable-length signals effectively. When these signals are sampled at the same time-steps, the question can be simplified. We first define the time-aligned batch dataset.

Definition 1 (Time-aligned batch dataset). *A batch dataset $\mathcal{D} = \{\mathcal{D}_i\}_{i=1}^N$ is said to be time-aligned if $N_i = N_j = T$ for all $i, j = 1, \cdots, N$ and $x_{it} = x_{jt} = x_t$ for all $i, j = 1, \cdots, N$, $t = 1 \cdots T$. In this case, each $\mathcal{D}_i$ consists of $\{(x_t, y_{it})\}_{t=1}^T$.*

We write $\mathbf{x} = [x_1, \cdots, x_T]^\intercal, \boldsymbol{\mu} = [\mu(x_1), \cdots, \mu(x_T)]^\intercal, \mathbf{C} = [c(x_m, x_n; \boldsymbol{\theta})]_{T \times T}$, then we have $\mathbf{y}_i|\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$. Since the dataset is time-aligned, different signals share the same mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{C}$. Unlike GPFR, we do not assume $\mu(x)$ is a linear combination of B-spline basis functions here. Instead, we can estimate the global mean function $\mu(x)$ non-parametrically. A naive idea is to estimate $\mu(x_t)$ by $\hat{\mu}(x_t) = \frac{1}{N}\sum_{i=1}^N y_{it}$ and then interpolate on general x. Theoretically, $y_{it} = y_i(x_t) = \mu(x_t) + \tau(x_t)$. Once we fix t and integrate out

y_{is} for $s \neq t$, we immediately obtain $y_{it} = \mu(x_t) + \varepsilon_i$ where ε_i is a Gaussian noise subject to $\mathcal{N}(0, \sigma^2)$ and $\sigma^2 = c(x, x; \boldsymbol{\theta})$. Therefore, $\hat{\mu}(x_t) \sim \mathcal{N}(\mu(x_t), \frac{\sigma^2}{N})$. This derivation justifies that $\hat{\mu}(x_t)$ is a good estimator of $\mu(x_t)$ as long as N is large enough. For general $x \neq x_t, \forall t = 1, \cdots, T$, we can linear interpolate on x to obtain the corresponding $\hat{\mu}(x)$ based on $\{(x_t, \hat{\mu}(x_t))\}_{t=1}^{T}$. We refer to this method as Gaussian Processes with Average Mean Functions (GPAVM).

We can extend GPAVM to mixture models and we refer to it as mix-GPAVM. In mix-GPFR, the number of B-spline basis functions D is the same among all components. Although there are K components, the smoothness of their mean functions $\{\mu_k(x)\}_{k=1}^{K}$ is globally controlled by the hyper-parameter D. Therefore, mix-GPFR cannot effectively model batch datasets with different smoothness in each component, while mix-GPAVM can adjust the smoothness within each component separately and adaptively. See Fig. 1 for a concrete example.

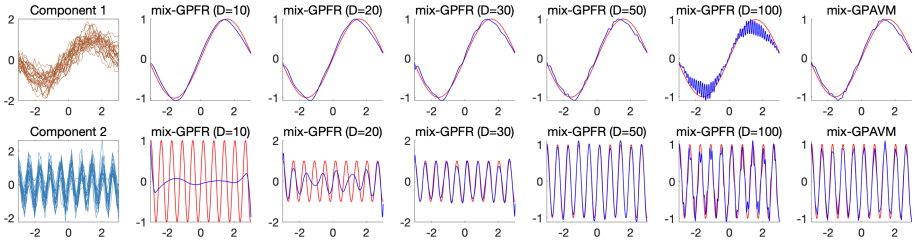


Fig. 1. The fitted results of mix-GPFR with different D on a batch dataset with two components.

3.2 Average Mean Functions Based EM Algorithm

Let $\mathbf{C}_k = [c(x_m, x_n; \boldsymbol{\theta}_k)]_{T \times T}$, $\hat{\boldsymbol{\mu}}_k$ be the mean function estimated in component k, and $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_k\}_{k=1}^{K} \cup \{\pi_k\}_{k=1}^{K} \cup \{\hat{\boldsymbol{\mu}}_k\}_{k=1}^{K}$ be the parameters to be estimated. The complete data log-likelihood is given by

$$\log p(\mathcal{D}, \{z_i\}_{i=1}^{N} | \boldsymbol{\Theta}) = \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{I}(z_i = k)(\log \pi_k + \log \mathcal{N}(\mathbf{y}_i | \hat{\boldsymbol{\mu}}_k, \mathbf{C}_k)).$$

Note that $\mathbf{C}_k$ depends on $\boldsymbol{\theta}_k$. In the E-step, we need to calculate the posterior distribution of z_i, which is $p(z_i = k | \mathcal{D}, \boldsymbol{\Theta}^{\text{old}}) \propto \pi_k \mathcal{N}(\mathbf{y}_i | \hat{\boldsymbol{\mu}}_k, \mathbf{C}_k)$. In the E-step, we update $\alpha_{ik} = p(z_i = k | \mathcal{D}, \{\boldsymbol{\theta}_k^{\text{old}}\}_{k=1}^{K})$, then the $\mathcal{Q}$-function is

$$\begin{aligned} \mathcal{Q}(\boldsymbol{\Theta} | \boldsymbol{\Theta}^{\text{old}}) &= \mathbb{E}_{p(z_i = k | \mathcal{D}, \{\boldsymbol{\theta}_k^{\text{old}}\}_{k=1}^{K})} [\log p(\mathcal{D}, \{z_i\}_{i=1}^{N} | \{\boldsymbol{\theta}_k\}_{k=1}^{K})] \\ &= \sum_{i=1}^{N} \sum_{k=1}^{K} \alpha_{ik} (\log \pi_k + \log \mathcal{N}(\mathbf{y}_i | \hat{\boldsymbol{\mu}}_k, \mathbf{C}_k)). \end{aligned}$$

In the M-step, we need to maximize $\mathcal{Q}(\boldsymbol{\Theta} | \boldsymbol{\Theta}^{\text{old}})$ with respect to $\boldsymbol{\Theta}$. For π_k and $\hat{\boldsymbol{\mu}}_k$, we can update them explicitly via $\pi_k = \frac{1}{N} \sum_{i=1}^{N} \alpha_{ik}$, $\hat{\boldsymbol{\mu}}_k = \frac{1}{N} \sum_{i=1}^{N} \alpha_{ik} \mathbf{y}_i$.

Table 1. 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

For $\{\boldsymbol{\theta}_k\}_{k=1}^K$, we perform gradient ascent on $\mathcal{Q}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{\text{old}})$. The main difference between mix-GPFR and mix-GPAVM in terms of parameter learning is that we do not need to perform gradient ascent on coefficients $\{\mathbf{b}_k\}_{k=1}^K$. Estimating $\{\hat{\boldsymbol{\mu}}_k\}_{k=1}^K$ by a weighted sum is easy and fast, while optimizing $\{\mathbf{b}_k\}_{k=1}^K$ and $\{\boldsymbol{\theta}_k\}_{k=1}^K$ simultaneously is difficult and time-consuming.

3.3 Prediction Strategy

Suppose an $(N+1)$-th curve $\mathcal{D}_{N+1}=\{x_t, y_{N+1,t}\}_{t=1}^{T_*}$ is given, where $T_*<T$ is the signal length. To estimate the response y_* of $(N+1)$-th signal at x_*, we first calculate the probability that $(N+1)$-the curve belongs to k-th component, which is given by

$$p(z_{N+1}=k|\mathcal{D},\mathcal{D}_{N+1};\boldsymbol{\Theta})\propto\pi_k\mathcal{N}(\mathbf{y}_{N+1};\hat{\boldsymbol{\mu}}_k(1:T_*),\mathbf{C}_k(1:T_*,1:T_*)).$$

Here, $(1:T_*)$ is the slicing syntax. In k-the component, the prediction is

$$\hat{y}_k=\hat{\mu}(x_*)+\mathbf{c}_*\mathbf{C}_k(1:T_*,1:T_*)^{-1}(\mathbf{y}_{N+1}-\hat{\boldsymbol{\mu}}_k(1:T_*)),$$

where $c_*=[c(x,x_1;\boldsymbol{\theta}_k),\cdots,c(x,x_{T_*};\boldsymbol{\theta}_k)]^\intercal$. The final prediction is the weighted average of these predictions, *i.e.*, $\hat{y}=\sum_{k=1}^K p(z_{N+1}=k|\mathcal{D},\mathcal{D}_{N+1};\boldsymbol{\Theta})\hat{y}_k$.

4 Experimental Results

4.1 On Synthetic Datasets

Dataset Description. We use the mix-GPFR model with K components to generate the time-aligned synthetic datasets, with each component consists of 20 curves for training and 10 curves for testing. We have 100 observed samples in

Table 2. Average RMSEs, CARs and running times (in seconds) with standard deviations of the proposed and competing methods on synthetic datasets.

Method	D	$\mathcal{S}_3$			$\mathcal{S}_5$		
		RMSE	CAR	Time	RMSE	CAR	Time
GP	–	4.6923 ± 0.0000	–	0.39 ± 0.01	4.5716 ± 0.0000	–	0.56 ± 0.01
GPFR	10	4.6147 ± 0.0000	–	0.79 ± 0.01	4.4549 ± 0.0000	–	1.02 ± 0.02
	20	4.6074 ± 0.0000	–	0.77 ± 0.01	4.4560 ± 0.0000	–	1.03 ± 0.07
	30	4.6051 ± 0.0000	–	0.77 ± 0.01	4.4592 ± 0.0000	–	0.96 ± 0.01
	50	4.6150 ± 0.0000	–	0.78 ± 0.02	4.4697 ± 0.0000	–	1.10 ± 0.19
mix-GP	–	4.5834 ± 0.0055	–	3.31 ± 0.54	4.1826 ± 0.0206	–	15.35 ± 8.83
mix-GPAVM	–	0.4814 ± 0.0000	$100.00\% \pm 0.00\%$	2.00 ± 0.68	0.6845 ± 0.2736	$96.84\% \pm 8.79\%$	3.93 ± 0.73
mix-GPFR	10	0.4822 ± 0.0000	$100.00\% \pm 0.00\%$	2.45 ± 0.40	0.8484 ± 0.4201	$93.60\% \pm 10.84\%$	8.15 ± 5.46
	20	0.4882 ± 0.0000	$100.00\% \pm 0.00\%$	2.40 ± 0.30	0.9422 ± 0.4311	$91.17\% \pm 11.86\%$	9.47 ± 5.94
	30	0.5876 ± 0.0000	$100.00\% \pm 0.00\%$	2.43 ± 0.38	0.8003 ± 0.3298	$96.03\% \pm 8.86\%$	7.27 ± 5.87
	50	1.3762 ± 0.0000	$100.00\% \pm 0.00\%$	2.45 ± 0.37	1.1163 ± 0.1790	$96.17\% \pm 8.72\%$	6.95 ± 4.11
Method	D	$\mathcal{S}_7$			$\mathcal{S}_9$		
		RMSE	CAR	Time	RMSE	CAR	Time
GP	–	5.3324 ± 0.0000	–	0.78 ± 0.02	4.6994 ± 0.0000	–	1.02 ± 0.19
GPFR	10	5.2751 ± 0.0000	–	1.33 ± 0.21	4.6234 ± 0.0000	–	1.61 ± 0.24
	20	5.2786 ± 0.0000	–	1.31 ± 0.28	4.6231 ± 0.0000	–	1.62 ± 0.23
	30	5.2768 ± 0.0000	–	2.17 ± 0.94	4.6265 ± 0.0000	–	1.62 ± 0.22
	50	5.2786 ± 0.0000	–	1.28 ± 0.06	4.6266 ± 0.0000	–	1.63 ± 0.22
mix-GP	–	4.9123 ± 0.0138	–	22.47 ± 11.20	4.3753 ± 0.0087	–	27.04 ± 9.97
mix-GPAVM	–	0.7665 ± 0.3102	$97.23\% \pm 6.66\%$	6.59 ± 0.56	0.9209 ± 0.3142	$92.57\% \pm 7.36\%$	10.12 ± 1.70
mix-GPFR	10	0.8128 ± 0.3623	$96.97\% \pm 7.09\%$	14.04 ± 10.41	0.9657 ± 0.3337	$93.42\% \pm 6.56\%$	29.41 ± 16.96
	20	0.8955 ± 0.4237	$94.54\% \pm 8.83\%$	16.85 ± 12.37	0.9256 ± 0.3403	$94.27\% \pm 6.87\%$	26.91 ± 22.48
	30	1.1095 ± 0.3338	$93.29\% \pm 8.58\%$	17.14 ± 15.18	1.1776 ± 0.2280	$92.55\% \pm 7.72\%$	30.28 ± 22.98
	50	1.5697 ± 0.1396	$94.46\% \pm 7.81\%$	14.06 ± 9.13	1.2276 ± 0.2103	$92.09\% \pm 7.51\%$	33.78 ± 30.71

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 vary the number of components K in $\{3, 5, 7, 9\}$, and these datasets are referred to as $\mathcal{S}_3, \mathcal{S}_5, \mathcal{S}_7, \mathcal{S}_9$, respectively. The mean functions and parameters of Gaussian processes used to generated these datasets are detailed in Table 1, and a dataset with K components involves the first K rows of Table 1.

Comparison Methods and Parameter Settings. We apply GP, GPFR and mix-GPFR methods on these synthetic datasets for comparison. For mix-GPFR and mix-GPAVM, the number of components K is set to be the ground-truth number of components. For GPFR and mix-GPFR, there is an extra parameter: the number of B-spline basis functions D. We vary D in $\{10, 20, 30, 50\}$. We run each method 10 times and report the averaged performance metrics.

Evaluation Metrics. We consider the performances of these methods from two aspects. On the one hand, we concern whether the algorithm can predict the evolving trend of testing curves, and we evaluate the prediction performances via Rooted Mean Square Errors (RMSEs). Suppose there are M testing curves in total, and for the i-th testing curve we need to estimate $y_{i,T/2+1}, y_{i,2}, \cdots, y_{i,T}$,

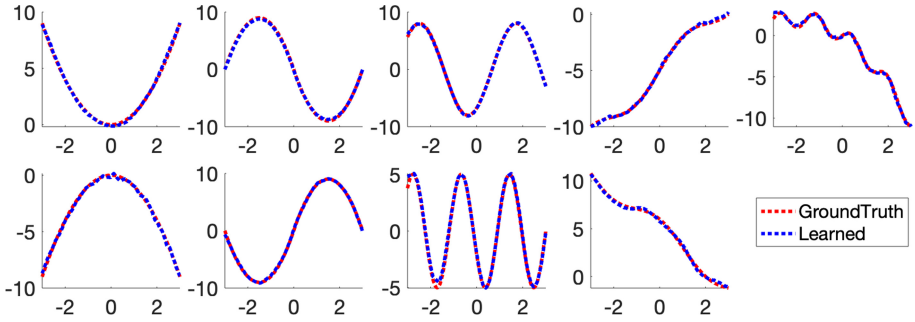
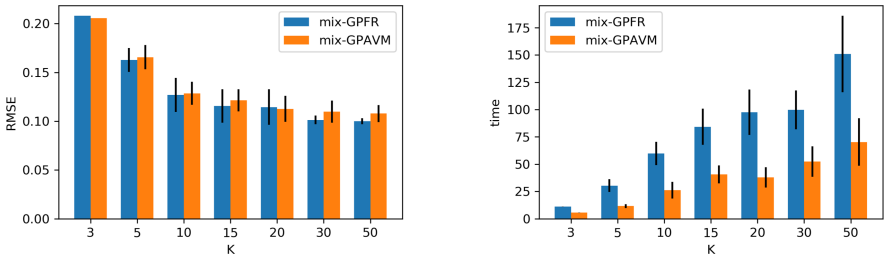


Fig. 2. Ground-truth mean functions (in red) and learned mean functions by mix-GPAVM (in blue). (Color figure online)

the RMSE is defined as RMSE $= \frac{2}{MT}\sum_{i=1}^{M}\sum_{t=T/2+1}^{T}(\hat{y}_{i,t} - y_{i,t})^2$. Here, $\{\hat{y}_{i,t}\}$ are predictions obtained by the algorithms. On the other hand, we concern whether the algorithm successfully partition the training time-series into meaningful groups and reveal the underlying cluster structure of the dataset. We use the Classification Accuracy Rate (CAR) to evaluate the performance of component identification, which is defined as CAR $= \max_{\xi\in\Pi_K}\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(z_i = \xi(\hat{z}_i))$. Here, $\{z_i\}_{i=1}^{N}$ are ground-truth labels as in Eq. 1 and $\{\hat{z}_i\}_{i=1}^{N}$ are cluster labels assigned by the algorithms. Π_K denotes the set of K-permutations, and the permutation ξ is employed to account for the label switching problem.

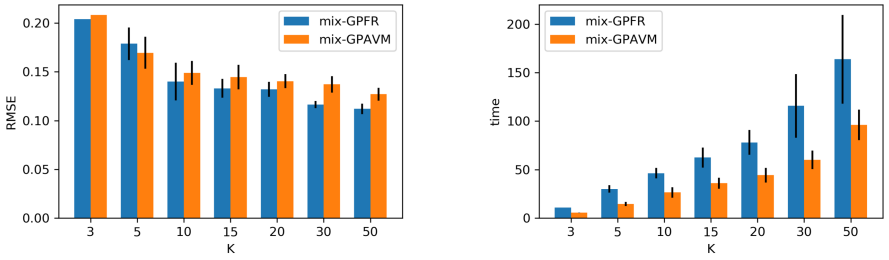
The results are reported in Table 2. We find that a single GP or a single GPFR is not flexible enough to model these datasets, since in each dataset the curves are generated by various signal sources. Besides, mix-GP fails to cluster synthetic datasets into meaningful groups since the global mean function is ignored. Usually, mix-GPAVM performs better than mix-GPFR in terms of RMSE and CAR, which demonstrates that mix-GPAVM is effective in curve prediction and clustering. Furthermore, mix-GPAVM is significantly faster than mix-GPFR, and it is even faster than mix-GP. The reason is introducing global mean functions in the model accelerates the convergence of the EM iteration, and the computational cost of estimating mean functions in mix-GPAVM is marginal. We also observe that the setting of D influences the results severely. On $\mathcal{S}_3$ and $\mathcal{S}_7$, mix-GPFR with $D = 10$ attains the best results, and increasing D has a negative effect on the results. However, on $\mathcal{S}_5$, the optimal D equals to 30, while $D = 20$ leads to the best performances on $\mathcal{S}_9$. Therefore, setting a proper D is very challenging in mix-GPFR, while mix-GPAVM can adjust the smoothness of mean functions adaptively.

We illustrate the mean functions estimated by mix-GPAVM on $\mathcal{S}_9$ in Fig. 2. We observe that mix-GPAVM successfully learns the mean functions without obvious over-fitting or under-fitting.



(a) Electricity 2009/RMSE.

(b) Electricity 2009/time.



(c) Electricity 2010/RMSE.

(d) Electricity 2010/time.

Fig. 3. RMSEs and running times of mix-GPFR ($D = 30$) and mix-GPAVM on Electricity 2009 and Electricity 2010.

4.2 On Real-World Datasets

We use the electricity load dataset issued by the Northwest China Grid Company in this experiment. The electricity dataset records electricity loads in 2009 and 2010 every 15 min. Therefore, daily electricity loads can be regarded as a time-series of length 96, and this dataset is naturally time-aligned since the record times keep the same everyday. We further split the dataset according to the year, and refer to them as Electricity 2009/2010, respectively. Within each sub-datasets, we randomly select 200 curves for training and the rest 165 curves for testing. Since the number of components K is unknown on real-world datasets, we vary K in $\{3, 5, 10, 15, 20, 30, 50\}$. The RMSEs and running times of mix-GPFR ($D = 30$) and mix-GPAVM are shown in Fig. 3. On these two real-world datasets, the performance of mix-GPFR ($D = 30$) is slightly better than mix-GPAVM, but mix-GPFR is $2\times$ to $3\times$ slower than mix-GPAVM. Therefore, mix-GPAVM is able to achieve comparable results as mix-GPFR with a significantly shorter running time.

5 Conclusion

In this paper, we utilize average mean functions in the EM algorithm and propose the mixtures of Gaussian processes with average mean functions for time-aligned batch data. In mix-GPAVM, the mean functions are estimated by sample

average non-parametrically, and the effectiveness of this estimation is justified theoretically. Compared with mix-GPFR, mix-GPAVM can adaptively adjust the smoothness of mean functions. Furthermore, the proposed method is faster than mix-GPFR since estimating mean functions by sample average is computationally efficient. Experimental results validate the effectiveness and efficiency of the proposed method.

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

References

1. Rasmussen, C.E.: Gaussian processes in machine learning. In: Bousquet, O., von Luxburg, U., Rätsch, G. (eds.) ML -2003. LNCS (LNAI), vol. 3176, pp. 63–71. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28650-9_4
2. Zhao, J., Sun, S., Wang, H., Cao, Z.: Promoting active learning with mixtures of Gaussian processes. Knowl.-Based Syst. **188**, 105044 (2020)
3. Xu, M., Ding, W., Zhu, J., Liu, Z., Chen, B., Zhao, D.: Task-agnostic online reinforcement learning with an infinite mixture of Gaussian processes. arXiv preprint arXiv:2006.11441 (2020)
4. Velásquez, R.M.A., Lara, J.V.M.: Forecast and evaluation of COVID-19 spreading in USA with reduced-space Gaussian process regression. Chaos Solitons Fractals **136**, 109924 (2020)
5. Shi, J.Q., Choi, T.: Gaussian Process Regression Analysis for Functional Data. Chapman and Hall/CRC press (2011). https://doi.org/10.1201/b11038
6. Shi, J.Q., Wang, B., Murray-Smith, R., Titterington, D.M.: Gaussian process functional regression modeling for batch data. Biometrics **63**(3), 714–723 (2007)
7. Shi, J.Q., Wang, B.: Curve prediction and clustering with mixtures of Gaussian process functional regression models. Stat. Comput. **18**(3), 267–283 (2008)
8. Di, W., Ma, J.: A two-layer mixture model of Gaussian process functional regressions and its MCMC EM algorithm. IEEE Trans. Neural Netw. Learn. Syst. **99**, 1–11 (2018)
9. Wu, D., Ma, J.: A DAEM algorithm for mixtures of Gaussian process functional regressions. In: Huang, D.-S., Han, K., Hussain, A. (eds.) ICIC 2016. LNCS (LNAI), vol. 9773, pp. 294–303. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42297-8_28
10. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006). 738 pages. https://link.springer.com/book/9780387310732. ISBN: 978-0-387-31073-2
11. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. Ser. B (Methodol.) **39**(1), 1–22 (1977)