Article

Automated Model Selection of the Two-Layer Mixtures of Gaussian Process Functional Regressions for Curve Clustering and Prediction

Chengxin Gong and Jinwen Ma *

School of Mathematical Sciences and LMAM, Peking University, Beijing 100871, China; gongchengxin@pku.edu.cn
* Correspondence: jwma@math.pku.edu.cn

Abstract: As a reasonable statistical learning model for curve clustering analysis, the two-layer mixtures of Gaussian process functional regressions (TMGPFR) model has been developed to fit the data of sample curves from a number of independent information sources or stochastic processes. Since the sample curves from a certain stochastic process naturally form a curve cluster, the model selection of TMGPFRs, i.e., the selection of the number of mixtures of Gaussian process functional regressions (MGPFRs) in the upper layer, corresponds to the discovery of the cluster number and structure of the curve data. In fact, this is rather challenging because the conventional model selection criteria, such as BIC and cross-validation, cannot lead to a stable result in practice even with a heavy burden of repetitive computation. In this paper, we improve the original TMGPFR model and propose a Bayesian Ying-Yang (BYY) annealing learning algorithm for the parameter learning of the improved model with automated model selection. The experimental results of both synthetic and realistic datasets demonstrate that our proposed algorithm can make correct model selection automatically during parameter learning of the model.

Keywords: mixture of Gaussian processes; Gaussian process functional regression; model selection; Bayesian Ying-Yang harmony learning; curve clustering and prediction

MSC: 62M20; 62M45; 62F15

Citation: Gong, C.; Ma, J. Automated Model Selection of the Two-Layer Mixtures of Gaussian Process Functional Regressions for Curve Clustering and Prediction. *Mathematics* **2023**, *11*, 2592. https://doi.org/10.3390/math11122592

Academic Editors: Gheorghe Săvoiu and María del Carmen Valls Martínez

Received: 10 April 2023
Revised: 23 May 2023
Accepted: 30 May 2023
Published: 6 June 2023

1. Introduction

As a powerful statistical learning model, the Gaussian process (GP), along with its variants, has been widely used in various scenarios of data analysis and information processing to capture the intrinsic features of time series in a variety of fields such as regression analysis, pattern recognition, image processing, and computer vision [1–4]. However, there exist two major drawbacks which strongly limit the capability of the conventional GP model. Firstly, the training iteration involves the covariance matrix inversion, which is significantly time-consuming for a large-scale dataset [5]. Secondly, for the convenience of parameter learning, the mean function of GP is usually assumed to be zero, so that the learned GP is actually a stationary process. Therefore, it is rather difficult for a GP to model the multi-mode time series, in which there are violent fluctuations [6].

In order to overcome these drawbacks, Tresp [5] adopted the mixture of Gaussian processes (MGP) along the time or input region for multi-mode time series. It employs a number of GPs which are located at different intervals of the time region corresponding to the actual modes. In this case, the heavy computation of a large matrix inversion can be greatly reduced to those of some small matrix inversions in the training iteration because the MGP model limits the parameter learning of each component GP to a small piece of time series. To enhance the learning ability and accommodate different task scenarios, these kinds of MGPs have been improved from the structure and mechanism, such as the mixture of robust GPs [7], domain adaptation MGP [8], sequential local-based MGP [9], etc. The corresponding learning algorithms have been developed with three methodologies: Bayesian inference [10,11], Markov chain Monte Carlo [12,13], and EM algorithm [14,15]. Until now, theoretical research and practical applications of these GP mixtures are still active [16–19].

Nevertheless, the above GP mixtures merely consider the diversity of time series in the input space and try to learn a stochastic process from a given piece of time series, i.e., only one sample curve. In practical applications, it is more important and significant to consider the diversity of sample curves in the output space so that the curves can be clustered and predicted [20]. To this end, we should get rid of the zero-mean assumption. In fact, Shi et al. [6] proposed a model of Gaussian process functional regression (GPFR) by using a linear combination of B-spline functions to learn and approximate the mean function of the output, and then the GPFR mixture along the output space (referred to as mix-GPFR) was established for curve clustering and prediction [21]. However, these GP mixtures limit the curve clusters to be generated by a number of Gaussian processes, not general stochastic processes which can be actual information sources in practice. In order to overcome this limitation, Wu and Ma [22] established a two-layer mixture model of GPFRs (TMGPFR). In the lower layer, there are a number of GPFRs driven by the input variables, whereas in the upper layer, there are actually a number of mixtures of GPFRs (MGPFR) generated by linearly mixing a certain number of GPFRs in the lower layer, which further mix together in the output space and forms the output variables.

In the TMGPFR model, each MGPFR in the upper layer can model a general stochastic process that acts as a general random information source to generate a kind of sample curve as a curve cluster. Thus, the TMGPFR model can fit those curves generated from a number of independent random information sources and further make curve clustering and prediction after its parameters have been learned from the data of those curves. As the number of GPFRs in the lower layer is large enough and fixed, the number of MGPFRs in the upper layer corresponds to the number of actual clusters in the whole curve dataset. Theoretically, it is a model selection problem to determine the number of MGPFRs in the TMGPFR model for a given curve dataset, which has not been deeply investigated in the previous literature. In fact, if this model selection is made correctly during parameter learning, the true number of curve clusters as well as these actual clusters themselves can be found correctly according to the learned TMGPFR model, leading to an effective way of curve clustering and prediction. However, this particular model selection problem is rather challenging because the conventional model selection criteria, such as BIC and cross-validation, cannot yield a stable result in realistic cases. Furthermore, they are even exposed to a heavy load of repetitive computation.

Recently, the Bayesian Ying-Yang (BYY) harmony learning [23], Dirichlet process [24] and reversible-jump Markov chain Monte Carlo (RJMCMC) [25] have been applied to solving the model selection problem of MGP and achieved good effects [15,26–29]. They are referred to as automated model selection algorithms since they make model selection automatically during parameter learning of a given dataset. That is, the parameter estimation and model selection are made synchronously. However, it can be found by the experiments that the Dirichlet process-based algorithms usually select a larger number of components, whereas the RJMCMC-based algorithms are not very stable. In addition, these two kinds of algorithms require an additional EM step to conduct the parameter learning in order to yield a proper prediction, which impairs the automation of model selection. Motivated by the fact that the BYY harmony learning shows an excellent ability of automated model selection of Gaussian mixtures [30], we adopt it to solve this particular model selection problem in the TMGPFR model.

In this paper, we improve the original TMGPFR model to a tight and powerful structure and propose a BYY annealing learning algorithm for the improved TMGPFR model with automated model selection. Specifically, a fully connected structure from the lower layer to the upper layer is integrated into the TMGPFR model, and the BYY harmony function is derived for this kind of TMGPFR model. Moreover, the BYY annealing learning algorithm is designed for the parameter learning and automated model selection of the TMGPFR model of a curve dataset.

The remainder is organized as follows. We introduce the GP, GPFR, and TMGPFR models in Section 2. Then, we design our BYY annealing learning algorithm in Section 3. The decision strategies of our improved TMGPFR model and learning algorithm for curve clustering and prediction are discussed in Section 4. Section 5 presents our experimental results and analyses. We finally make further discussions and conclusions in Section 6.

2. Mathematical Description of the TMGPFR Model

2.1. The GP Model

We begin with a brief introduction of the Gaussian process (GP) model. Suppose that we have input variables $x = \{x_i\}_{i=1}^N$ and output (or response) variables $y = \{y_i\}_{i=1}^N$. Then the GP model can be defined by

$$y \sim \mathcal{N}(0, K(x,x)), \tag{1}$$

where $K(x,x)$ is the kernel matrix. A common form of the kernel matrix, the radial basis function (RBF) kernel, can be expressed by

$$K_\theta(x,x)_{ij} := K_\theta(x_i,x_j) = \theta_1^2 \exp\left(-\theta_2^2 \frac{||x_i - x_j||_2^2}{2}\right) + \theta_3^2 \delta_{ij}, \tag{2}$$

where δ_{ij} is the Kronecker delta function, and $\theta = (\theta_1, \theta_2, \theta_3)$. We adopt this kernel form throughout the paper.

2.2. The GPFR Model

Since the assumption of zero mean in the GP model is too limited, it is possible to use a weighted sum of certain basis functions to learn and approximate the mean function during the parameter learning. In general, a set of B-spline basis functions are adopted. Specifically, we substitute the mean function in the GP model with

$$m(x) = \sum_{d=1}^{D} \phi_d(x) b_d, \tag{3}$$

where D is the number of preset B-spline basis functions such that $\phi(x) = (\phi_1(x), \cdots, \phi_D(x))$ is a set of different B-spline basis functions, and $b = (b_1, \cdots, b_D)$ is a D-dimensional coefficient or weight vector. Obviously, the larger D brings the higher accuracy of the approximation but a heavier computing burden. Furthermore, the overfitting is more likely to emerge as D increases. This is certainly a trade-off problem and D is generally selected by experience.

In such a setting, the GP model is transformed into the so-called GPFR model [6], which is denoted by $y \sim \text{GPFR}(x, \theta, b, \phi)$.

2.3. The TMGPFR Model

The TMGPFR model was developed from the GPFR mixture models to fit the data of curves from different independent random information sources or stochastic processes. In the lower layer, there are a large number of GPFRs driven by the input variables, and in the upper layer, a proper number of MGPFRs is employed to describe these stochastic processes and further mixed together to form the output variables.

2.3.1. The Lower Layer: A Fixed Number of GPFRs for Constructing MGPFRs

We assume that there exist G GPFRs in the lower layer and G is fixed and large enough so that the mixture of these G GPFRs can approximate any stochastic process. For each single sample curve $\{(x_{mn}, y_{mn})\}_{n=1}^{N}$, the latent indicators $\{z_{mn}\}_{n=1}^{N}$ are generated by a multinomial distribution:

$$P(z_{mn} = g) = \eta_g, g = 1, 2, \cdots, G \text{ for } n = 1, 2, \cdots, N. \tag{4}$$

Given the indicator $z_{mn} = g$, the input x_{mn} is subject to a Gaussian distribution:

$$x_{mn}|z_{mn} = g \sim \mathcal{N}(\mu_g, \sigma_g^2) \text{ i.i.d. for } n = 1, 2, \cdots, N. \tag{5}$$

After specifying $\{z_{mn}, x_{mn}\}_{n=1}^{N}$, all the input variables are divided into G sets: $\{x_{1,s}|z_{mn} = 1\}_{s=1}^{S_1}, \cdots, \{x_{G,s}|z_{mn} = G\}_{s=1}^{S_G}$ and $\{x_{mn}\}_{n=1}^{N} = \{\{x_{g,s}|z_{mn} = g\}_{s=1}^{S_g}\}_{g=1}^{G}$. Then, the input variables in $x_g = \{x_{g,s}\}_{s=1}^{S_g}$ and the corresponding output variables of $y_g = \{y_{g,s}\}_{s=1}^{S_g}$ follow the g-th component GPFR model:

$$y_g \sim \text{GPFR}(x_g, \theta_g, b_g, \phi_g). \tag{6}$$

In this case, $\{(x_{mn}, y_{mn})\}_{n=1}^{N} = \{\{(x_{g,s}, y_{g,s})\}_{s=1}^{S_g}\}_{g=1}^{G}$.

Letting $\eta = (\eta_1, \cdots, \eta_G)$ and taking the similar notations for $\theta, b, \phi, \mu, \sigma$, we will use $y \sim \text{MGPFR}(x, \eta, \theta, b, \phi, \mu, \sigma)$ to denote a component MGPFR model in the upper layer. It should be noted that $\eta, \mu, \sigma \in \mathbb{R}^G, \theta \in \mathbb{R}^{3\times G}, b \in \mathbb{R}^{D\times G}, \phi \in \text{B-spline}^{D\times G}$.

2.3.2. The Upper Layer: MGPFRs for Constructing the Output Mix-MGPFR

We assume there exist K MGPFRs in the upper layer. For a dataset $\mathcal{D} = \{(x_m, y_m)\}_{m=1}^{M}$ where $(x_m, y_m) = \{(x_{mn}, y_{mn})\}_{n=1}^{N_m}$ denotes the m-th sample curve, latent indicators $\{\alpha_m\}_{m=1}^{M}$ are also generated by a multinomial distribution:

$$P(\alpha_m = k) = \pi_k, k = 1, 2, \cdots, K \text{ i.i.d. for } m = 1, 2, \cdots, M. \tag{7}$$

Given the indicator $\alpha_m = k$, we have

$$y_m|\alpha_m = k \sim \text{MGPFR}(x_m, \eta_k, \theta, b, \phi, \mu, \sigma). \tag{8}$$

It should be pointed out that the above parameters $\theta, b, \phi, \mu, \sigma$ do not have their own subscripts because all the G GPFRs in the lower layer are fully connected and contributed to each of the MGPFRs in the upper layer just like a fully connected neural network, so they share the same parameters $\theta, b, \phi, \mu, \sigma$ for all the MGPFRs. Certainly, the weight parameters $\eta_k \in \mathbb{R}^G, k = 1, \cdots, K$ are bound to be different because they serve as a unique feature to distinguish these MGPFRs. In fact, η_k reflects how these GPFRs in the lower layer essentially contribute to the k-th MGPFR in the upper layer.

According to the above description, our assumptions for the TMGPFR model are different from those in Wu and Ma [22]. That is, we adopt Equation (8) instead of $y_m|\alpha_m = k \sim \text{MGPFR}(x_m, \eta_k, \theta_k, b_k, \phi_k, \mu_k, \sigma_k)$. In fact, the original TMGPFR model in Wu and Ma [22] assumes that the k-th MGPFR model in the upper layer possesses its own G_k GPFRs in the lower layer and $\sum_{k=1}^{K} G_k = G$; thus, it is a sparse structure in the lower layer and the fully-connected structure is not actually implemented, which leads to the neglect of possible correlations among different MGPFRs. Therefore, we improve the TMGPFR model to a tight and powerful structure.

The TMGPFR model is actually a generative model which can describe a mixture of multiple information sources or stochastic processes and can be directly applied to the tasks of curve clustering and prediction of a given curve dataset $\mathcal{D}$. The information flow of the improved TMGPFR model is shown in Figure 1, whereas its structure sketch is displayed in Figure 2.

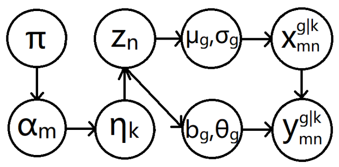


Figure 1. Information flow in the improved TMGPFR model, depicting how the sample curves are generated.

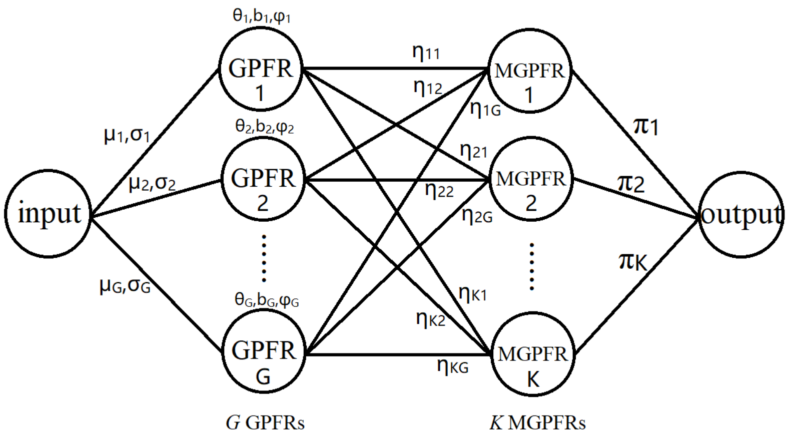


Figure 2. Structure sketch of the improved TMGPFR model, depicting how the input variables are processed and transformed to the output variables.

2.3.3. Notations

For clarity, we summarize all of the necessary notations which will be constantly used to describe our models and algorithms in the following.

To be more convenient, we define a 0–1 indicator variable A_m^k to describe the association between the m-th sample curve and the k-th MGPFR. That is, if the m-th sample curve belongs to the k-th MGPFR, $A_m^k = 1$; otherwise, $A_m^k = 0$. On the other hand, we define an analogous variable $B_{mn}^{g|k}$ to describe the association between the n-th sample point in the m-th sample curve and the g-th GPFR in the k-th MGPFR. That is, if the n-th sample point in the m-th sample curve belongs to the g-th GPFR on the condition that the m-th sample curve belongs to the k-th MGPFR, $B_{mn}^{g|k} = 1$; otherwise, $B_{mn}^{g|k} = 0$. Similarly, we denote $A = \{A_m^k | m = 1, 2, \cdots, M, k = 1, 2, \cdots, K\}$ and $B = \{B_{mn}^{g|k}, n = 1, 2, \cdots, N_m, m = 1, 2, \cdots, M, k = 1, 2, \cdots, K, g = 1, 2, \cdots, G\}$.

The notations of the other parameters are identical to those described in Section 2.3.2, including the dataset $\mathcal{D} = \{\{(x_{mn}, y_{mn})\}_{n=1}^{N_m}\}_{m=1}^{M}$, latent variables, and parameters in the model. We denote the whole parameter set as $\Theta = (D, K, G, \pi, \eta, \theta, b, \phi, \mu, \sigma)$, where $D, K, G \in \mathbb{Z}^+, \pi = (\pi_1, \cdots, \pi_K), \eta = (\eta_1, \cdots, \eta_K)^T = (\eta_{kg})_{K\times G}$. Among all the parameters, D, K, G, ϕ are preset and remain unchangeable, whereas the others are learned by the algorithm and will be updated during iterations.

3. BYY Annealing Learning Algorithm

3.1. BYY Harmony Function

A BYY harmony learning system describes an observation $\boldsymbol{x}$ and its inner representation α through two types of Bayesian decomposition of the joint probability: $p(\boldsymbol{x}, \alpha) = p(\boldsymbol{x})p(\alpha|\boldsymbol{x}) = q(\alpha)q(\boldsymbol{x}|\alpha)$. The former is called the Yang machine, whereas the latter is called the Ying machine. In our improved TMGPFR model, $\boldsymbol{x}$ denotes a sample curve (x, y) and α denotes the index of the corresponding MGPFR in the upper layer. $p(\alpha|\boldsymbol{x})$ is the probability (density) that $\boldsymbol{x}$ belongs to α, whereas $q(\boldsymbol{x}|\alpha)$ is the probability (density) that α generates $\boldsymbol{x}$. The objective of BYY harmony learning is to maximize the following harmony function:

$$H(p||q) = \iint p(\boldsymbol{x})p(\alpha|\boldsymbol{x})\log q(\alpha)q(\boldsymbol{x}|\alpha)d\boldsymbol{x}d\alpha - r_q, \tag{9}$$

where r_q is the regularizer. The theoretical details can be found in [23,31].

As for a specifically given dataset of the TMGPFR model, it is natural to use the empirical density function $\hat{p}(x,y) = \frac{1}{M(N_1+N_2+\cdots N_M)}\sum_{m=1}^{M}\sum_{n=1}^{N_m}\delta_{(x_{mn},y_{mn})}(x,y)$ to estimate the true density function $p(x,y)$. Here, M denotes the total number of curves and $X_m = \{(x_{mn},y_{mn}) : n = 1,2,\cdots,N_m\}$ denotes the m-th sample curve in the given dataset. Supposing that $N_1 = \cdots = N_M = N$ and that N and M are large enough, we have $\hat{p}(x,y) \approx p(x,y)$. For simplicity, the regularizer is omitted. In the same way as the BYY annealing harmony learning for Gaussian mixtures in [30], we consider the annealing harmony function $H_\lambda(\Theta) = H(p||q) + \lambda E(p(\alpha|\boldsymbol{x}))$, where $E(\cdot)$ is the entropy function over K MGPFRs:

$$E(p(\alpha|\boldsymbol{x})) = -\iint p(\alpha|\boldsymbol{x})\log p(\alpha|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}d\alpha, \tag{10}$$

and λ is a tuning parameter dependent on the iteration time t. When λ is relatively large, the maximization of $H_\lambda(\Theta)$ encourages multiple components with a soft classification. When λ tends to be zero, it gradually shifts to a hard-cut classification with a great number of local maximums. Thus, it is reasonable to select a large initial value $\lambda(0)$ and reduce it to zero during the iterations. In this way, the learning parameters can be ergodic to a certain extent and escape from local maximums, thus having a higher probability of reaching the global maximum of $H(p||q)$, which further leads to the automated model selection during the parameter learning [30]. That is, if the initial value K_{init} is set to be greater than the true value K_{true}, the global maximization of $H(p||q)$ can select K_{true} MGPFRs to match the actual components in the curve dataset and force the mixing proportions of the other $(K_{\text{init}} - K_{\text{true}})$ extra MGPFRs to vanish, i.e., $\pi_k = 0$. Therefore, the remaining K^* components through the global maximization of $H(p||q)$ are the actual components in the dataset so that the model selection is actually made automatically. That is, the automated model selection of the TMGPFRs can be effectively implemented by the global maximization of the annealing harmony function.

We now derive the specific expression of the annealing harmony function of our improved TMGPFR model of a given curve dataset. By substituting $p(x,y)$ with $\hat{p}(x,y)$ in Equations (9) and (10), we can obtain the empirical harmony and annealing harmony functions as follows:

$$\hat{H}(p||q) = \frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K} p(\alpha_m = k|(x_m,y_m)) \times \log[\pi_k q((x_m,y_m)|\alpha_m = k)]; \tag{11}$$

$$\hat{H}_\lambda(\Theta) = \hat{H}(p||q) - \lambda \times \frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K} p(\alpha_m = k|(x_m,y_m)) \times \log p(\alpha_m = k|(x_m,y_m)). \tag{12}$$

According to the assumptions in our improved TMGPFR model, $q((x_m,y_m)|\alpha_m = k)$ takes the following specific form:

$$q((x_m,y_m)|\alpha_m = k) = \prod_{g=1}^{G}\left\{\left[\prod_{n=1}^{N_m}\left(\eta_{kg}p_{\mathcal{N}}(x_{mn}|\mu_g,\sigma_g^2)\right)^{B_{mn}^{g|k}}\right]p_{\mathcal{N}}(y_{mn}^{g|k}|b_g,\theta_g)\right\}. \tag{13}$$

Here, $\{x_{mn}^{g|k}\} = \{x_{mn}, n = 1,2,\cdots,N_m|B_{mn}^{g|k} = 1\}$ and $\{y_{mn}^{g|k}\} = \{y_{mn}, n = 1,2,\cdots,N_m|B_{mn}^{g|k} = 1\}$. $\prod_{n=1}^{N_m}\left(\eta_{kg}p_{\mathcal{N}}(x_{mn}|\mu_g,\sigma_g^2)\right)^{B_{mn}^{g|k}}$, $p_{\mathcal{N}}(y_{mn}^{g|k}|b_g,\theta_g)$ are the likelihoods for x_m and y_m conditioned for $\alpha_m = k$, respectively. In the following, we will use the notation $x_{mn}^{g|k}$ or $y_{mn}^{g|k}$ to denote the vector composed of the elements in the set $\{x_{mn}^{g|k}\}$ or $\{y_{mn}^{g|k}\}$ and define $S_{m,k,g} = |\{x_{mn}^{g|k}\}|$.

3.2. Alternative Optimization Algorithm

As the empirical annealing harmony function is clear and specific with Equations (11)–(13), we further design the parameter learning algorithm to maximize it. Actually, the maximization of this annealing harmony function is rather difficult and cannot be solved directly. However, it is lucky that the whole parameters and certain unknown variables can be divided into two parts and we can effectively optimize each of them alternatively. It should be noted that λ is the annealing factor that does not need to be optimized. In this way, we can design an alternative optimization algorithm as follows.

We first divide the whole parameters and unknown variables into 2 sets: $\Theta_1 = \{\{p(\alpha_m = k|(x_m, y_m))\}_{k,m}, B\}$ and $\Theta_2 = \{\pi, \eta, \theta, b, \mu, \sigma\} = \Theta\backslash\{D, K, G, \phi\}$. It is clear that Θ_2 contains the common parameters of the TMGPFR model, whereas Θ_1 contains the conditional probabilities as well as the latent indicators of the curve data. After initializing $D, K, G, \phi, \Theta_1, \Theta_2$, we then iterate Θ_1 and Θ_2 in the following steps until convergence or reaching the maximum iterations T in an alternative optimization manner:

Step 1:Fix Θ_2, update $\Theta_1 = \text{argmax}_{\Theta_1} \hat{H}_\lambda(\Theta_1, \Theta_2)$;
Step 2: Fix Θ_1, update $\Theta_2 = \text{argmax}_{\Theta_2} \hat{H}_\lambda(\Theta_1, \Theta_2)$;
Step 3: Update $\lambda = \lambda^{(t)} < \lambda^{(t-1)}$, where t denotes the iteration time.

In Step 1, it is key to determine the values of the elements of B. In fact, only when all the relations among the sample data points, GPFRs, and MGPFRs are known, we can compute $q((x_m, y_m)|\alpha_m = k)$ according to Equation (13) and then update $p(\alpha_m = k|(x_m, y_m))$. To solve this problem, there are two possible methods: Markov chain Monte Carlo (MCMC) simulation [22] and hard-cut classification [14]. Since the time complexity of the MCMC simulation is rather high [22,32], we choose the following hard-cut rule to obtain the values of the elements of B.

According to the Bayes formula, we first have

$$p(B_{mn}^{g|k} = 1|(x_{mn}, y_{mn}), \alpha_m = k) = \frac{\eta_{kg} p([x_{mn}, y_{mn}]|B_{mn}^{g|k} = 1)}{\sum_{g=1}^{G} \eta_{kg} p((x_{mn}, y_{mn})|B_{mn}^{g|k} = 1)}, \tag{14}$$

where

$$p((x_{mn}, y_{mn})|B_{mn}^{g|k} = 1) = p_{\mathcal{N}}(x_{mn}|\mu_g, \sigma_g^2) \times p_{\mathcal{N}}(y_{mm}|\phi(x_{mn}) \cdot b_g, \theta_{g1}^2 + \theta_{g3}^2). \tag{15}$$

By letting each data point belong to the component with the highest probability, we then have the hard-cut rule:

$$B_{mn}^{g|k} = \begin{cases} 1, \text{ if } g = \text{argmax}_g p(B_{mn}^{g|k} = 1|[x_{mn}, y_{mn}], \alpha_m = k); \\ 0, \text{otherwise.} \end{cases} \tag{16}$$

With the above update of B, we can compute $q((x_m, y_m)|\alpha_m = k)$ according to Equation (13). It should be noted that as Θ_1 is updated, there exist certain restrictions, i.e., $\sum_{k=1}^{K} p(\alpha_m = k|(x_m, y_m)) = 1, \forall m$. By using the method of Lagrange multipliers, we can obtain the following update rule:

$$p(\alpha_m = k|(x_m, y_m)) = \frac{[\pi_k q((x_m, y_m)|\alpha_m = k]^{1/\lambda}}{\sum_{k=1}^{K} [\pi_k q((x_m, y_m)|\alpha_m = k)]^{1/\lambda}}. \tag{17}$$

From Equation (17), the role of λ can be further understood. Actually, $p(z_n = k|[x_n, y_n]) \to \frac{1}{K}$ (i.e., soft classification) as $\lambda \to +\infty$, and $p(z_n = k|[x_n, y_n]) \to \delta_{k,\text{argmax}_k p(z_n = k|[x_n, y_n])}$ (i.e., hard classification) as $\lambda \to 0+$. This gives a theoretical explanation of the action of λ in the training process.

In Step 2, there also exist $K+1$ restrictions, i.e., $\sum_{k=1}^{K}\pi_k = 1, \sum_{g=1}^{G}\eta_{kg} = 1, \forall k$. In a similar way, we can derive the update rules of $\pi, \eta, \mu, \sigma, b, \theta$. The details of the derivations are given in Appendix A.

In Step 3, $\lambda(t)$ plays an important role in the convergence behavior with the annealing mechanism. It is clear that if $\lambda(t)$ attenuates faster, the algorithm converges faster but the object function is more likely to be trapped into a local maximum. Therefore, $\lambda(t)$ should be carefully designed to balance the tradeoff between the convergence speed and effect. For this algorithm, we adopt $\lambda(t) = [u(1-e^{-v(t-1)}) + w]^{-1}$, given in [30].

The pseudo-code of our BYY annealing learning algorithm is given in Algorithm 1.

Algorithm 1: The BYY annealing learning algorithm of the TMGPFR model.

```
Input: $\mathcal{D}, D, K, G, \phi, T, T', u, v, w, \epsilon$, learning_rate
Output: $\pi, \eta, \theta, b, \mu, \sigma, p(\alpha_m = k|(x_m, y_m)), B, K^*$
1  Randomly initialize $\pi, \eta, \theta, b, \mu, \sigma, p(\alpha_m = k|(x_m, y_m)), B$;
2  Set $t = 1, H_0(p||q) = 0, \Delta H = -\infty$;
3  while $t \le T$ and $\Delta H \ge \epsilon$ do
4      Set $\lambda = [u(1-e^{-v(t-1)}) + w]^{-1}$;
5      Update $B$ according to Equations (14)–(16);
6      Update $p(\alpha_m = k|(x_m, y_m))$ for all $m, k$ according to Equation (17);
7      Compute $H_t(p||q)$ according to Equation (11);
8      $\Delta H \leftarrow |(H_t(p||q) - H_{t-1}(p||q))/H_{t-1}(p||q)|$;
9      Update $\pi, \eta, \mu, \sigma$ in turn according to Equations (A4)–(A7);
10     for $t' = 1, 2, \cdots, T'$ do
11         for $g = 1, 2, \cdots, G$ do
12             Update $b_g$ according to Equations (8)–(10);
13             Compute the gradient with regard to $\theta_g$ according to Equation (A11),
                 denoted by grad$_g$;
14             $\theta_g \leftarrow \theta_g +$ learning_rate $\times$ grad$_g$;
15         end
16     end
17     $t \leftarrow t + 1$;
18 end
```

4. Decision Strategies for Curve Clustering and Prediction

In this section, we further discuss the decision strategies of our improved TMGPFR model and learning algorithm for the tasks of curve clustering and prediction of a test set $\mathcal{D}' = \{\{(x'_{mn}, y'_{mn})\}_{n=1}^{N'_m}\}_{m=1}^{M'}$ after the BYY annealing algorithm has been implemented in the training dataset $\mathcal{D}$ with the desired parameters.

For curve clustering, the decision strategy or rule for a new curve is clear and simple. First, we determine all the values of the elements in the set $B' = \{B_{mn}^{g|k'}\}$ for any curves in $\mathcal{D}'$ according to Equations (14)–(16). Then, we compute $q((x'_m, y'_m)|\alpha'_m = k)$ and use the Bayes formula to obtain $p(\alpha'_m = k|(x'_m, y'_m))$ for all m, k. Finally, we choose $k = \text{argmax}_k p(\alpha'_m = k|(x'_m y'_m))$ to classify or label (x'_m, y'_m). On the whole, we can use the classification accuracy of $\mathcal{D}'$ to measure the performance of curve clustering by our algorithm.

For the curve prediction, the situation becomes complicated. Given a sample curve (x'_m, y'_m) in $\mathcal{D}'$, we need to predict y'_{mn*} after giving a new input variable x'_{mn*}. Assume that we have already finished the complete learning process required for clustering except labeling the sample curves. Then, for any pair $(k, g) \in \{1, 2, \cdots, K\} \times \{1, 2, \cdots, G\}$, we can compute $p_{kg} = p(B_{mn*}^{g|k'} = 1|x'_{mn*}, \alpha_m = k)$ according to the following Bayes formula:

$$p_{kg} = \frac{\eta_{kg} p_{\mathcal{N}}(x'_{mn*}|\mu_g, \sigma_g^2)}{\sum_{g=1}^{G} \eta_{kg} p_{\mathcal{N}}(x'_{mn*}|\mu_g, \sigma_g^2)}. \quad (18)$$

Then its unique prediction $y'_{kg}(x'_{mn*})$ is computed by

$$y'_{kg}(x'_{mn*}) = \phi_g(x'_{mn*})b_g + C_{kg}\Sigma_{kg}^{-1}(y_{mn}^{g|k'} - \Phi_{kg}b_g), \tag{19}$$

where

$$C_{kg} = K_{\theta_g}(x'_{mn*}, x_{mn}^{g|k'}) \in \mathbb{R}^{S'_{m,k,g}}; \tag{20}$$

$$\Sigma_{kg} = K_{\theta_g}(x_{mn}^{g|k'}, x_{mn}^{g|k'}) \in \mathbb{R}^{S'_{m,k,g}\times S'_{m,k,g}}; \tag{21}$$

$$[\Phi_{kg}]_{ij} = \phi_{gj}(x_{mni}^{g|k'}), \Phi_{kg} \in \mathbb{R}^{S'_{m,k,g}\times D}. \tag{22}$$

For any $k \in \{1, 2, \cdots, K\}$, we can compute its unique prediction $y'_k(x'_{mn*})$ by

$$y'_k(x'_{mn*}) = \sum_{g=1}^{G} p_{kg} y'_{kg}(x'_{mn*}). \tag{23}$$

Finally, we fuse all the component predictions and obtain the ultimate prediction by

$$y'_{mn*} = \sum_{k=1}^{K} p(\alpha_m = k|(x'_m, y'_m))y'_k(x'_{mn*}). \tag{24}$$

Moreover, we can use the root mean square error (RMSE) between this prediction and the true value to measure the prediction accuracy.

5. Experimental Results

5.1. Curve Clustering Analysis

5.1.1. On the Simulation Dataset

We begin with the test of our improved TMGPFR model through the BYY annealing learning algorithm of a simulation dataset. Typically, we generate a set of $M = 50$ unlabeled curves from three MGPFR models with different weights: $\pi_1 = 0.2, \pi_2 = 0.3, \pi_3 = 0.5$. Each curve has 60 points, i.e., $N = N_1 = N_2 = \cdots = N_{50} = 60$. The true values of the parameters of three MGPFR models are given in Table 1. It should be noted that we use some other continuous functions to be the mean functions of the GPFRs rather than the given B-spline functions themselves, which are denoted by $m_i(x)$.

We implement our BYY annealing learning algorithm of this dataset with the initial values of the parameters being set in a simple manner, which are specifically described in Appendix B.1. Our experimental results are shown in Figure 3. In the experiments, we actually set $K_{\text{init}} = 10$. After the automated model selection during the parameter learning, we finally obtain $K^* = 3$. It can be observed from Figure 3 that even if the structures of three curve clusters in the left subfigure are so complex and interlaced, all the sample curves are clustered correctly and the classification accuracy of our algorithm arrives at 100%.

We also implement our algorithm of the datasets with $M = 200, 500, 1000, 5000$, 10,000, and 50,000, i.e., the number of sample curves increases from 200 to 50,000 such that the sample size changes from small to big. In this situation, we set the batch size to 200 and slow down the annealing speed. Our experimental results of these datasets are given in Table 2. It can be found from Table 2 that the classification accuracy of our algorithm fluctuates as M increases, because outliers are more likely to appear due to the stronger randomness. Furthermore, in this case, our algorithm tends to choose a larger K^* with some components only occupying 0.1–5.0% of the whole curves. However, it does not necessarily mean the prediction results will worsen. We will further clarify it in Section 5.2.

In addition, we conduct several parallel experiments to compare our algorithm with other model selection methods. From Figure A1, we can observe that our BYY annealing learning algorithm performs much better. The details of the experiments and comparisons can be found in Appendix B.1.

Table 1. True values of the parameters in three MGPFRs.

Parameters in MGPFR1	**True Values**
η_1, η_2	$0.7, 0.3$
θ_1, θ_2	$(0.32, 0.4, 0.01), (0.4, 0.5, 0.01)$
$m_1(x), m_2(x)$	$\sin(x^3) - 0.3\cos(x^3), \cos(\sqrt{\lvert x\rvert})$
μ_1, μ_2	$-10, 10$
σ_1, σ_2	$5, 4$
Parameters in MGPFR2	**True Values**
η_1, η_2	$0.6, 0.4$
θ_1, θ_2	$(0.3, 0.9, 0.01), (0.2, 0.2, 0.01)$
$m_1(x), m_2(x)$	$\exp(-\lvert x\rvert), 0.01x^2 - 0.1\lvert x\rvert + 1$
μ_1, μ_2	$-8, 6$
σ_1, σ_2	$2, 8$
Parameters in MGPFR3	**True Values**
η_1, η_2	$0.5, 0.5$
θ_1, θ_2	$(0.25, 0.1, 0.01), (0.45, 0.9, 0.01)$
$m_1(x), m_2(x)$	$\sin x + \cos x, \frac{1}{\lvert x\rvert+1} + 0.1\lvert x\rvert - 0.5$
μ_1, μ_2	$-4, 10$
σ_1, σ_2	$3, 5$

Table 2. Average classification accuracies of our BYY annealing leaning algorithm of the simulation datasets with different M.

M	**200**	**500**	**1000**	**5000**	**10,000**	**50,000**
Accuracy	98.2%	96.6%	91.5%	92.6%	90.3%	91.9%
K^*	3	4	4	4	5	5

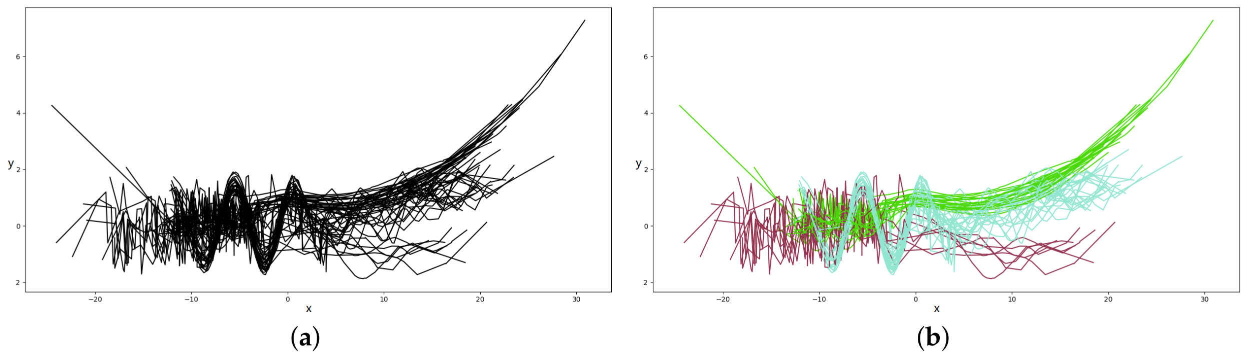
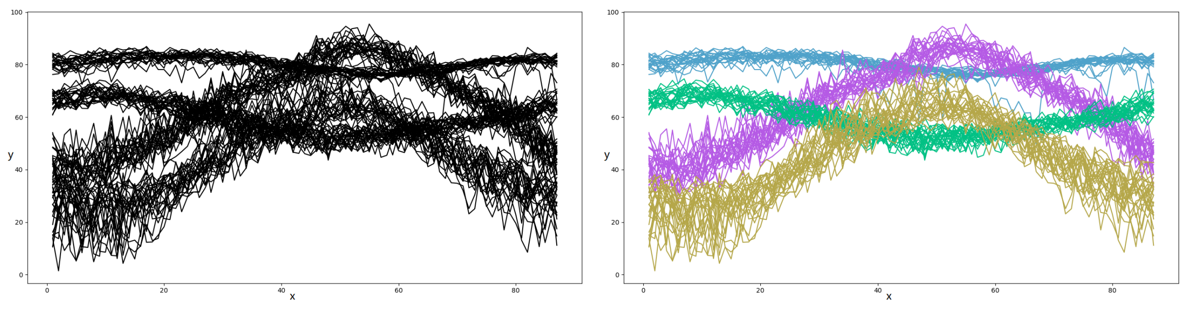


Figure 3. Experimental results of the simulation dataset. The x-axis represents the input variables $\{\{x_{mn}\}_{n=1}^{N}\}_{m=1}^{M}$ and the y-axis represents the output (or response) variables $\{\{y_{mn}\}_{n=1}^{N}\}_{m=1}^{M}$. (**a**) 50 curves of three MGPFRs without their labels (i.e., the algorithm's input). (**b**) 50 curves of three MGPFRs with the predicted labels (i.e., the algorithm's predicted labels; the same color represents the same label).

5.1.2. On the City Temperature Dataset

We further implement our BYY annealing learning algorithm of a real dataset of daily temperatures of different cities. In practice, it is very crucial to determine the type of climate in a certain city according to its daily temperature records. In total, we choose $M = 97$ unlabeled sample curves from 4 cities: Abidjan, Auckland, Shanghai, and Helsinki. Each curve consists of 87 sample points, reflecting the change in temperatures in one year. Specifically, each sample point is the average of city temperatures in a period of 4 adjacent days. Due to the inaccuracy of measurement, some abnormal sample points are discarded. The experimental results are shown in Figure 4 and the details are described in Appendix B.2. Here, $K_{\text{init}} = 10$, and our algorithm finally yields $K^* = 4$ after the automated model selection. It is found that the classification accuracy is 100% for this particular real-world task.



(**a**) (**b**)

Figure 4. Experimental results of the city temperature dataset. The x-axis represents the input variables $\{\{x_{mn}\}_{n=1}^{N}\}_{m=1}^{M}$ and the y-axis represents the output (or response) variables $\{\{y_{mn}\}_{n=1}^{N}\}_{m=1}^{M}$. (**a**) 97 sample curves without the labels (i.e., the algorithm's input). (**b**) 97 sample curves with the predicted labels (i.e., the algorithm's predicted labels, the same color represents the same label).

5.1.3. On the Effective Radiation Dataset

We finally implement our BYY annealing learning algorithm in a real dataset of daily surface effective radiation of different cities in China. In total, we choose $M = 100$ unlabeled sample curves from 20 different cities: Eji'na, Guilin, Chaoyang, Shenyang, Juxian, Tianshui, Hengfeng, Zhongshan, Wulatezhongqi, Changchun, Dalian, Zhenghe, Leshan, Fuzhou, Xi'an, Yichang, Luohe, Emeishan, Yanji, and Shanghai. The meanings and processing methods of this dataset keep the same as those described in Section 5.1.2. However, due to data missing, a sample curve in this dataset has only 75 sample points. The experimental results are shown in Figure 5. Here $K_{\text{init}} = 10$ and our algorithm yields $K^* = 4$ after the automated model selection. The details of the experiments are described in Appendix B.3.

Our classification criterion is the winner-take-all (WTA) principle, i.e., a certain station is definitely classified into the category that its five daily radiation curves mostly belong to. Therefore, despite the fact that these curves are divided into four types by our algorithm, for cities themselves there exist only three types. As shown in Figure 5, a purple curve forms the fourth type, possibly due to certain random noise.

According to our experimental results, all the cities are divided into three clusters: {Eji'na, Wulatezhongqi} (pink), {Chaoyang, Shenyang, Juxian, Tianshui, Changchun, Dalian, Xi'an, Luohe, Yanji} (blue), and {Guilin, Hengfeng, Zhongshan, Zhenghe, Leshan, Fuzhou, Yichang, Emeishan, Shanghai} (orange). In fact, Eji'na and Wulatezhongqi lie in the deserts of the Inner Mongolian Plateau, which explains why these two cities are exposed to the highest solar radiation. The cities in the blue set have a higher latitude than those in the orange set; thus, they are exposed to a longer duration of sunshine in the summer and a lower rainfall during the whole year. These factors make them the second highest radiated. In sum, the clustering results of our algorithm are consistent with the meteorological facts, which demonstrate the effectiveness of our model and algorithm.

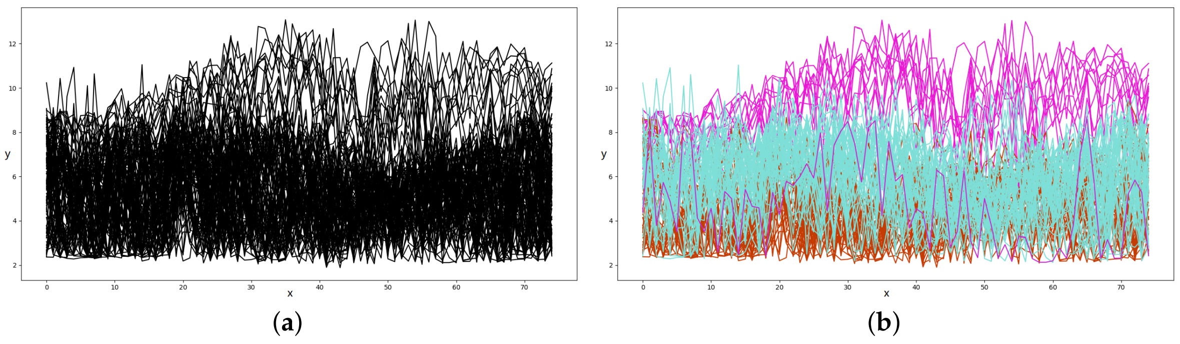


Figure 5. Experimental results of the daily radiation dataset. The x-axis represents the input variables $\{\{x_{mn}\}_{n=1}^{N}\}_{m=1}^{M}$ and the y-axis represents the output (or response) variables $\{\{y_{mn}\}_{n=1}^{N}\}_{m=1}^{M}$. (**a**) 100 sample curves without the labels (i.e., the algorithm's input). (**b**) 100 sample curves with the predicted labels (i.e., the algorithm's predicted labels, the same color represents the same label).

For comparison, we use two typical model selection criteria through the EM algorithm to make parameter learning and model selection of this dataset. It is found by our experiments that the AIC approach yields $K^* = 2$, whereas the BIC approach even yields $K^* = 1$, which are both too rough. These two experimental results demonstrate that the conventional model selection criteria based on the penalty of the number of authentic parameters often fail to tackle the complicated and noisy dataset.

5.2. Curve Prediction

We now turn to test our BYY annealing learning algorithm on curve prediction. In fact, given a piece of time series (as part of a sample curve) from the past to the present, it is very important to predict what will probably appear in the future. A reasonable prediction model can help us to understand the intrinsic logic of the data and make more reasonable decisions. In this subsection, we test our algorithm on an electrical load dataset issued by the Northwest China Grid Company. Actually, there are $M = 100$ training sample curves and 50 test sample curves, each of which records the values of the electrical load over the period of a whole day. The electrical load is detected every 15 min and then there are altogether 96 points in one sample curve. We implement the BYY annealing learning algorithm in the training sample curves to obtain the reasonable TMGPFR model and then utilize it to make the curve prediction, i.e, to predict the future output of the curve with a new given input according to Equation (24). The experimental results of the curve clustering are shown in Figure 6.

It can be found from Figure 6 that the curve clustering results of the electrical load dataset are quite reasonable since these clusters effectively represent different intensities of the electrical load over a day. Moreover, the curve clustering results of the test dataset are consistent with those of the training dataset. In fact, these reasonable clustering results lay a solid foundation for curve prediction.

As for the curve prediction in this dataset, we assume that there are U known sample points in the front and $96 - U$ unknown sample points in the behind. That is, we use the previous U sample points to predict the following $96 - U$ sample points. It is clear that as U becomes smaller, the prediction becomes more difficult. We make curve predictions of the test dataset for different U and the experimental results are listed in Table 3, where the mean and its standard deviation (SD) and the maximum of the RMSEs over 46 prediction trials are given for certain typical values of U. It can be found that the prediction results are rather precise and stable even if U is only 20.

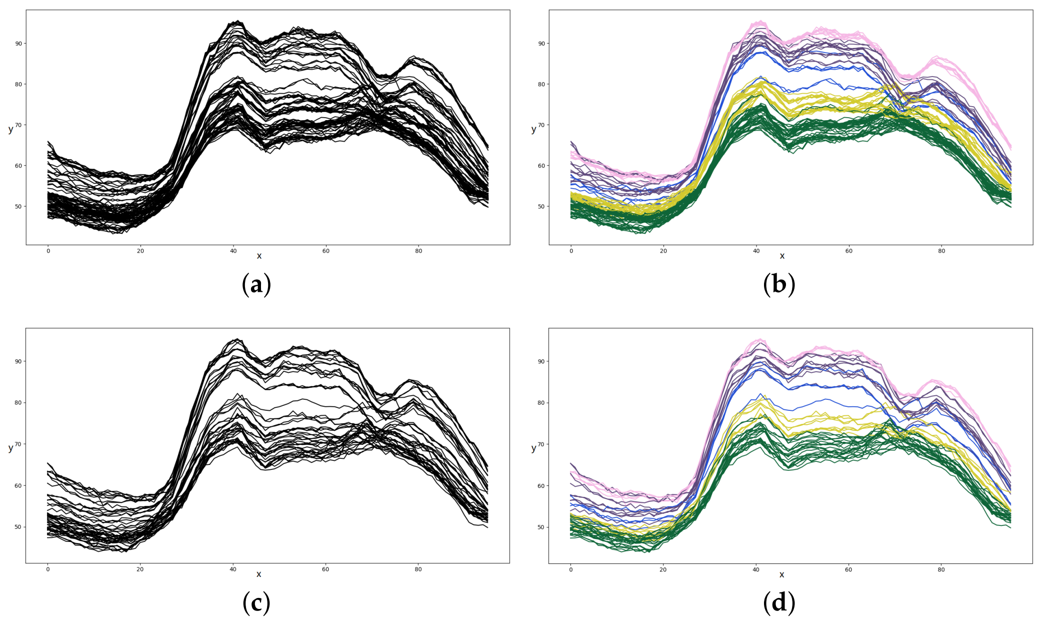


Figure 6. Experimental results of curve clustering on the electrical load dataset. The x-axis represents the input variables $\{\{x_{mn}\}_{n=1}^{N}\}_{m=1}^{M}$ and the y-axis represents the output (or response) variables $\{\{y_{mn}\}_{n=1}^{N}\}_{m=1}^{M}$. (**a**) Training sample curves without labels (i.e., the algorithm's input). (**b**) Training sample curves with the predicted labels of the clustering (i.e., the algorithm's predicted labels, the same color represents the same label or cluster). (**c**) Test sample curves without labels (i.e., the algorithm's input). (**d**) Test sample curves with the predicted labels of the clustering (i.e., the algorithm's predicted labels, the same color represents the same label or cluster).

Table 3. Statistics of the RMSEs of curve predictions of our BYY annealing learning algorithm with different U.

U	20	30	40	50	60	70
mean $\pm$ SD	3.38 ± 0.94	2.56 ± 0.38	2.14 ± 0.29	1.99 ± 0.17	1.93 ± 0.22	1.79 ± 0.14
maximum	4.43	3.28	2.68	2.41	2.51	2.07

We also test our BYY annealing learning algorithm on curve prediction for some larger sizes of this dataset; that is, we let $M = 200, 300, 400, 500, 680$. Here $U \equiv 50$ and the batch size is 100. The larger M is, the slower annealing rate we adopt. The prediction results are listed in Table 4. It can be found that they are more precise, stable, and consistent, except the possible difference of K^*.

Table 4. Statistics of the RMSEs of curve predictions of our BYY annealing learning algorithm with different M and $U \equiv 50$.

M	200	300	400	500	680
mean $\pm$ SD	1.93 ± 0.18	1.64 ± 0.19	1.93 ± 0.22	1.84 ± 0.15	1.92 ± 0.35
maximum	2.24	1.96	2.38	2.07	2.44
K^*	4	5	5	4	4

For comparison, we first implement three conventional model selection methods: AIC, BIC, and k-fold cross-validation (CV), with the EM algorithm of this electrical load dataset. The hyperparameters in the TMGPFR model keep constant and are selected by experience. The curve clustering results are shown in Figure A3 and the curve prediction results are listed in Table 5. Actually, the AIC and BIC approaches both yield $K^* = 3$, the 5-fold

CV approach yields $K^* = 4$ and the 10-fold CV approach yields $K^* = 5$. By comparing Tables 3 and 5, we can find out that our BYY annealing learning algorithm outperforms these conventional model selection methods of this dataset. It is clear that the prediction results of the AIC, BIC, and 5-fold CV approaches are worse than our BYY annealing learning algorithm. Although the prediction error of the 10-fold CV approach is relatively smaller than ours, the difference between them is quite small. On the other hand, our algorithm really saves plenty of time and GPU resource owing to the synchronization of parameter learning and model selection.

Table 5. Statistics of the RMSEs of curve predictions of three conventional model selection approaches with different U.

Method	U	20	30	40	50	60	70
AIC & BIC	mean ± SD	3.30 ± 1.08	3.40 ± 0.69	3.18 ± 0.45	3.09 ± 0.43	2.74 ± 0.25	2.58 ± 0.42
	maximum	4.61	4.38	3.92	4.02	3.26	3.12
5-fold CV	mean ± SD	3.42 ± 0.96	2.78 ± 0.34	2.15 ± 0.33	2.23 ± 0.25	2.22 ± 0.28	2.04 ± 0.28
	maximum	4.76	3.17	2.60	2.62	2.61	2.50
10-fold CV	mean ± SD	3.42 ± 1.02	2.45 ± 0.27	1.91 ± 0.18	1.90 ± 0.16	1.96 ± 0.16	1.85 ± 0.17
	maximum	5.03	2.98	2.34	2.36	2.32	2.09

We further compare our improved TMGPFR model with some state-of-the-art statistical and machine-learning models of curve prediction. The first kinds of comparative models are several previous statistical models, such as GPFR, mix-GPFR, MGPFR, and the original TMGPFR model proposed in Wu and Ma [22]. The EM algorithm is employed for training the GPFR and MGPFR models, whereas the BYY annealing learning algorithm is employed for training the mix-GPFR and the original TMGPFR models. The second kinds of comparative models are state-of-the-art machine learning models, such as long short-term memory (LSTM) [33], fully visible belief network (FVBN) [34,35], and generative adversarial nets (GAN) [36,37]). The experiments are conducted under the same conditions for $U = 50$ and the curve prediction results are listed in Tables 6 and 7. It can be seen that our improved TMGPFR model achieves the best performance for the aspects of both prediction precision and stability. Furthermore, it can be found from the experiments that the previous statistical models are weak (see Table 6) mainly due to their simpler assumptions, whereas LSTM and FVBN fail (see Table 7) mainly due to the accumulative errors in the rolling or successive prediction, and GAN fails (see Table 7) mainly due to the mode collapsing.

More analyses and supplementary experiments of this dataset are stated in Appendix B.4.

Table 6. Statistics of the RMSEs of curve predictions of the GPFR, mix-GPFR, MGPFR, original TMGPFR, and improved TMGPFR models with different $\{D, G)\}$ when $U = 50$.

Model	G	D	10	15	20	25	30
GPFR	-	RMSE	7.03 ± 1.83	7.04 ± 1.82	7.03 ± 1.82	7.03 ± 1.83	7.02 ± 1.83
mix-GPFR ($K_{\text{init}} \equiv 10$)	-	RMSE	3.02 ± 0.32	2.92 ± 0.31	2.50 ± 0.21	2.49 ± 0.21	2.50 ± 0.17
		K^*	2	2	3	3	3
MGPFR	4	RMSE	7.08 ± 1.74	7.33 ± 2.27	7.54 ± 2.00	7.10 ± 1.74	7.02 ± 1.67
	6	RMSE	7.23 ± 1.82	7.11 ± 1.78	7.00 ± 1.75	7.12 ± 1.73	7.27 ± 1.73
	8	RMSE	7.18 ± 2.06	7.11 ± 1.76	7.05 ± 1.77	7.18 ± 1.76	7.04 ± 1.83
	10	RMSE	7.16 ± 1.80	7.10 ± 1.77	7.03 ± 1.82	7.03 ± 1.82	7.03 ± 1.80
	12	RMSE	7.07 ± 1.79	7.13 ± 1.73	7.04 ± 1.82	7.13 ± 1.79	7.08 ± 1.78

Table 6. *Cont.*

Model	G	D	10	15	20	25	30
The original TMGPFR [22] ($G_k \equiv 2$, $K_{\text{init}} \equiv G/G_k$)	4	RMSE	3.16 ± 0.38	3.17 ± 0.35	3.06 ± 0.28	2.92 ± 0.32	2.92 ± 0.23
		K^*	2	2	2	2	2
	6	RMSE	2.37 ± 0.33	2.83 ± 0.57	2.41 ± 0.24	2.50 ± 0.19	2.42 ± 0.22
		K^*	3	3	3	3	3
	8	RMSE	2.96 ± 0.24	2.20 ± 0.38	2.50 ± 0.20	2.54 ± 0.24	2.62 ± 0.30
		K^*	2	3	3	3	3
	10	RMSE	2.43 ± 0.20	2.51 ± 0.37	2.49 ± 0.17	2.63 ± 0.20	2.00 ± 0.16
		K^*	3	3	3	3	4
	12	RMSE	2.87 ± 0.33	2.64 ± 0.33	2.31 ± 0.22	2.11 ± 0.20	2.42 ± 0.27
		K^*	2	3	4	4	4
improved TMGPFR	4	RMSE	2.79 ± 0.20	2.89 ± 0.35	2.46 ± 0.28	2.45 ± 0.32	2.30 ± 0.26
		K^*	2	3	3	3	3
	6	RMSE	2.41 ± 0.23	2.49 ± 0.34	2.48 ± 0.22	2.24 ± 0.31	2.27 ± 0.35
		K^*	3	3	3	3	3
	8	RMSE	2.11 ± 0.17	2.23 ± 0.20	1.94 ± 0.18	2.16 ± 0.30	2.12 ± 0.19
		K^*	4	4	6	5	5
	10	RMSE	2.32 ± 0.42	2.14 ± 0.24	2.10 ± 0.32	2.00 ± 0.20	1.99 ± 0.17
		K^*	4	5	5	5	5
	12	RMSE	2.20 ± 0.29	2.08 ± 0.21	1.98 ± 0.26	1.89 ± 0.16	2.04 ± 0.16
		K^*	4	5	6	7	7

Table 7. Statistics of the RMSEs of curve predictions of LSTM, FVBN, and GAN when $U = 50$.

Model	**LSTM**			**FVBN**		
params	layers = 15 hid dims = 16	layers = 5 hid dims = 32	layers = 32 hid dims = 3	linear lags = 2	linear lags = 10	linear lags = 50
RMSE	11.73 ± 2.97	11.73 ± 2.96	11.75 ± 2.98	4.23 ± 3.52	7.07 ± 2.49	7.17 ± 3.76
model	**FVBN**			**GAN (·/· : generator/discriminator params)**		
params	LeakyReLU NN layers = 5 hid dims = 256	Tanh NN layers = 4 hid dims = 1024	Sigmoid NN layers = 6 hid dims = 128	vanilla GAN layers = 20/3 hid dims = 512/2	WGAN layers = 25/2 hid dims = 128/2	WGAN layers = 40/2 hid dims = 512/2
RMSE	4.51 ± 1.97	4.50 ± 1.55	8.15 ± 2.38	6.59 ± 2.61	5.68 ± 2.19	5.17 ± 2.16

6. Discussion and Conclusions

6.1. Relation to the EM Algorithms

Interestingly, the updating rules of our BYY annealing learning algorithm take similar forms as those of the EM algorithm and the deterministic annealing EM algorithm [38]. However, our idea is quite different from those of the EM algorithms. For the EM algorithm, the annealing parameter λ should be fixed to 1, whereas in the deterministic annealing EM algorithm, λ should gradually approach to 1. Their common goal is to reach the global maximum of the likelihood function. However, in our BYY annealing algorithm, λ attenuates to 0, which drives the algorithm to reach the global maximum of the harmony function. That is exactly why our BYY annealing algorithm possesses the ability of automated model selection but the EM algorithms usually do not. Actually, the harmony function provides severe penalization for large K, whereas the likelihood function is bound to increase with K. Unlike AIC, BIC, and CV, we integrate this kind of penalization into the parameter learning so that the model selection can be made automatically. Therefore, we do not need to train multiple candidate models for model selection.

6.2. The Annealing Mechanism in the BYY Harmony Learning System

When training the TMGPFR model with the hard-cut EM algorithm, we find out that a small perturbation of the initial values of the parameters might strongly impact the results, especially in model selection. The reason is that there are a large number of local maximums of the BYY harmony function with the hard-cut form so that the hard-cut EM algorithm is easily trapped. On the contrary, as we apply the annealing mechanism to the hard-cut EM algorithm, most of the components are able to attract certain samples due to a higher posterior probability in the early iterations (see Equation (17)). Moreover, the change in the parameters is more smooth because the annealing mechanism can prevent the abrupt change in the sample number of each component. In this way, the BYY harmony learning system gradually turns into a hard-cut classification form to arrive at the optimal model. Hence, we must emphasize that without the annealing mechanism, it is rather difficult for the hard-cut EM algorithm to reach the global maximum of the harmony function, and therefore the automated model selection cannot be guaranteed.

6.3. Conclusions and Remarks

Conclusions. We have improved the TMGPFR model and proposed a BYY annealing learning algorithm for its parameter learning with the automated model selection of a given dataset. Specifically, we fixed the number of GPFRs in the lower layer and let them be fully connected to the MGPFRs in the upper layer such that the structure of this improved TMGPFR model becomes tight and powerful. Under the BYY harmony learning framework, the BYY harmony function of the improved TMGPFR model is derived and the BYY annealing learning algorithm is established with the ability to automatically find out the number of actual clusters in a given curve dataset and further make curve clustering and prediction. The experimental results of both simulation and real datasets have shown the effectiveness and superiority of the improved TMGPFR model with the BYY annealing learning algorithm.

Remarks. First, the differences between the blind source separation (BSS) techniques and the TMGPFR modeling. In the setting of the BSS models, the original independent sources are blind and cannot be detected individually. However, there are some sensors that can obtain linearly mixed signals from these original sources. When the number of mixed signals is equal to that of the sources, the independent component analysis (ICA) algorithms can be applied to decompose these signals of the original independent sources, whereas in the setting of our TMGPFR model, the original independent sources are also blind. As their sample curves are accumulated together without any labels in the dataset, our goal of the TMGPFR modeling is to discover the structure of these sources as a number of general stochastic processes throughout the BYY harmony annealing learning of the TMGPFR model in a set of unlabeled sample curves. Moreover, we can even find out the true number of the original sources in the dataset with the automated model selection mechanism. Second, the hard-cut mechanism in our BYY annealing harmony algorithm. Although we adopt it to determine the set B to speed up the training process, it impairs the convergence of the algorithm to a certain extent, especially when these GPFRs in the lower layer are strongly overlapped [22]. Fortunately, the annealing mechanism can mitigate these negative effects. The MCMC method (e.g., Gibbs sampler) can play the same role with the expense of a slower convergence rate. Third, the influences of $\boldsymbol{D}, \boldsymbol{G}$ in our TMGPFR model. We only care about how to select K in the TMGPFR model, regardless of D, G. When D or G becomes larger (smaller), overfitting (underfitting) is more likely to emerge. It should be pointed out that our core goal is to find out the actual number of curve clusters in a dataset under the assumption that D, G are large enough so that the TMGPFR model can describe a mixture of general stochastic processes. However, it is still a problem to choose the optimal D, G for a given dataset. In our experiments, we apply certain heuristic methods to select D, G, which will be described in Appendix C.

Author Contributions: Conceptualization, C.G. and J.M.; Methodology, C.G. and J.M.; Software, C.G.; Validation, C.G.; Formal analysis, C.G.; Investigation, C.G.; Resources, J.M.; Data curation, J.M.; Writing—original draft, C.G.; Writing—review & editing, J.M.; Visualization, C.G.; Supervision, J.M.; Project administration, J.M.; Funding acquisition, J.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China under grant 2018AAA0100205.

Data Availability Statement: Restrictions apply to the availability of these data. The electrical load data were obtained from the State Grid of China and are available from the corresponding authors with the permission of the State Grid of China. Other data can be directly acquired from the corresponding authors.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of the data; in the writing of the manuscript; nor in the decision to publish the results.

Appendix A. BYY Annealing Learning Algorithm

In Step 2 of our alternative optimization algorithm, we aim to obtain the updating rules of $\Theta_2 = \{\pi, \eta, \mu, \sigma, b, \theta\}$ with $\Theta_1 = \{\{p(\alpha_m = k|(x_m, y_m)\}_{k,m}, B\}$ fixed. Since the empirical entropy function $\hat{E}(p(\alpha|\boldsymbol{x})) := -\sum_{m=1}^{M}\sum_{k=1}^{K} p(\alpha_m = k|(x_m, y_m)) \times \log p(\alpha_m = k|(x_m, y_m))$ is now a constant in this situation, we can consider the following optimization problem:

$$\begin{aligned}\pi.\eta, \mu, \sigma, b, \theta &= \mathrm{argmax}_{\Theta_2} \hat{H}_\lambda(\Theta_1, \Theta_2)\\ &= \mathrm{argmax}_{\pi,\eta,\mu,\sigma,b,\theta} \frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K} p(\alpha_m = k|(x_m, y_m)) \log(\pi_k q((x_m, y_m)|\alpha_m = k))\end{aligned} \tag{A1}$$

Note that there exist $K+1$ restrictions, i.e.,

$$\sum_{k=1}^{K} \pi_k = 1, \sum_{g=1}^{G} \eta_{kg} = 1, \forall k = 1, 2, \cdots, K \tag{A2}$$

Thus, we introduce $K+1$ Lagrange multipliers and obtain the Lagrangian function:

$$\mathscr{L}(\Theta_2, \gamma) = \hat{H}(p||q) + \gamma_0(\sum_{k=1}^{K} \pi_k - 1) + \sum_{k=1}^{K} \gamma_k(\sum_{g=1}^{G} \eta_{kg} - 1) \tag{A3}$$

By setting $\frac{\partial \mathscr{L}}{\partial \pi_k} = \frac{\partial \mathscr{L}}{\partial \eta_{kg}} = \frac{\partial \mathscr{L}}{\partial \mu_g} = \frac{\partial \mathscr{L}}{\partial \sigma_g^2} = \frac{\partial \mathscr{L}}{\partial \gamma_i} = 0$ for all $k, g \in \{1, 2, \cdots, K\} \times \{1, 2, \cdots, G\}$ and $i = 0, 1, \cdots, K$, we obtain

$$\pi_k = \frac{1}{M}\sum_{m=1}^{M} p(\alpha_m = k|(x_m, y_m)) \tag{A4}$$

$$\eta_{kg} = \frac{\sum_{m=1}^{M}\left(p(\alpha_m = k|(x_m, y_m))\sum_{n=1}^{N_m} B_{mn}^{g|k}\right)}{\sum_{m=1}^{M} p(\alpha_m = k|(x_m, y_m))N_m} \tag{A5}$$

$$\mu_g = \frac{\sum_{m=1}^{M}\sum_{k=1}^{K}\left(p(\alpha_m = k|(x_m, y_m))\sum_{n=1}^{N_m} B_{mn}^{g|k} x_{mn}\right)}{\sum_{m=1}^{M}\sum_{k=1}^{K}\left(p(\alpha_m = k|(x_m, y_m))\sum_{n=1}^{N_m} B_{mn}^{g|k}\right)} \tag{A6}$$

$$\sigma_g^2 = \frac{\sum_{m=1}^{M}\sum_{k=1}^{K}\left(p(\alpha_m = k|(x_m, y_m))\sum_{n=1}^{N_m} B_{mn}^{g|k}(x_{mn} - \mu_g)^2\right)}{\sum_{m=1}^{M}\sum_{k=1}^{K}\left(p(\alpha_m = k|(x_m, y_m))\sum_{n=1}^{N_m} B_{mn}^{g|k}\right)} \tag{A7}$$

However, the parameters b, θ do not have closed-form solutions when their gradients are equal to zero. Therefore, we have to implement a subalternative optimization algorithm. For every m, k, g, we define $S_{m,k,g} = |\{x_{mn}^{g|k}\}|$ and $\{x_{mn}^{g|k}\} = \{x_{mni}^{g|k}\}_{i=1}^{S_{m,k,g}}$. Fixing θ_g, we have a closed-form solution of b_g to maximize $H(p||q)$:

$$b_g = \left(\sum_{m=1}^{M} \sum_{k=1}^{K} p(\alpha_m = k|(x_m, y_m)\Phi_{m,k,g}^T \Sigma_{m,k,g}^{-1} \Phi_{m,k,g} \right)^{-1} \times \left(\sum_{m=1}^{M} \sum_{k=1}^{K} p(\alpha_m = k|(x_m, y_m))\Phi_{m,k,g}^T \Sigma_{m,k,g}^{-1} y_{mn}^{g|k} \right) \tag{A8}$$

where

$$[\Phi_{m,k,g}]_{ij} = \phi_{gj}(x_{mni}^{g|k}), \Phi_{m,k,g} \in \mathbb{R}^{S_{m,k,g} \times D} \tag{A9}$$

$$\Sigma_{m,k,g} = K_{\theta_g}(x_{mn}^{g|k}, x_{mn}^{g|k}) \in \mathbb{R}^{S_{m,k,g} \times S_{m,k,g}} \tag{A10}$$

Fixing b_g, we can update θ_g based on its gradient:

$$\begin{aligned} \frac{\partial H(p||q)}{\partial \theta_g} &= \frac{1}{2M} \sum_{m=1}^{M} \sum_{k=1}^{K} p(\alpha_m = k|(x_m, y_m)) \\ &\times \left((y_{mn}^{g|k} - \Phi_{m,k,g} b_g)^T \Sigma_{m,k,g}^{-1} \frac{\partial \Sigma_{m,k,g}}{\partial \theta_g} \Sigma_{m,k,g}^{-1} (y_{mn}^{g|k} - \Phi_{m,k,g} b_g) - \mathrm{tr}(\Sigma_{m,k,g}^{-1} \frac{\partial \Sigma_{m,k,g}}{\partial \theta_g}) \right) \end{aligned} \tag{A11}$$

Appendix B. Experimental Details

A specific description of how we choose the hyperparameters and the initialization methods for the common parameters is presented in Appendix C. In this section, we mainly focus on the experimental details. The codes of our BYY annealing learning algorithm can be downloaded at https://github.com/WQgcx/Automated-model-selection-of-TMGPFR (accessed on 1 June 2023). Our experiments are conducted with AMD Ryzen 5 4600U with Radeon Graphics, 2.10 GHz.

Appendix B.1. On the Simulation Dataset

The parameters of the generator with three different MGPFRs briefly mentioned in Section 5.1.1 are listed in Table 1. Recall that there are $M = 50$ unlabeled sample curves in total and in each curve there are 60 sample points.

Then, the initial values of the parameters in our BYY annealing learning algorithm are set in the following way: $K_{\text{init}} = 10, \pi_1 = \cdots = \pi_{K_{\text{init}}} = \frac{1}{K_{\text{init}}}, G = 10, \eta_k \sim \text{Dirichlet}(1, \cdots, 1)$ i.i.d. for $k = 1, \cdots, K_{\text{init}}, \theta_g \equiv (0.5, 1.0, 0.1), \mu_g \sim \text{Uniform}(\{-20, -19, \cdots, 20\})$ i.i.d. for $g = 1, 2, \cdots, G, \sigma_g \equiv 2, D = 20, b_g \sim \text{Uniform}(0, \frac{2}{D})$ i.i.d. for $g = 1, 2, \cdots, G$. When selecting the B-spline functions, for simplicity, we assume $\phi_1 = \phi_2 = \cdots = \phi_G$ and the interpolation points $\{(\bar{x}_d, \bar{y}_d)\}_{d=1}^{D} = \{\{(\bar{x}_{di}, \bar{y}_{di})\}_{i=1}^{n}\}_{d=1}^{D}$ are determined by the following steps:

Step 1: Set $n = N + 2$ and select a sufficiently small W_1 and a sufficiently large W_2 such that the interval $[W_1, W_2]$ can cover all the inputs x;
Step 2: Randomly select D different sample curves from the training dataset, being denoted by $\{(\hat{x}_d, \hat{y}_d)\}_{d=1}^{D} = \{\{(\hat{x}_{di}, \hat{y}_{di})\}_{i=1}^{N}\}_{d=1}^{D}$;
Step 3: Set $(\bar{x}_{d1}, \bar{y}_{d1}) = (W_1, 0), (\bar{x}_{d2}, \bar{y}_{d2}) = (\hat{x}_{d1}, \hat{y}_{d1}), (\bar{x}_{d3}, \bar{y}_{d3}) = (\hat{x}_{d2}, \hat{y}_{d2}), \cdots, (\bar{x}_{d(n-1)}, \bar{y}_{d(n-1)}) = (\hat{x}_{dN}, \hat{y}_{dN}), (\bar{x}_{dn}, \bar{y}_{dn}) = (W_2, 0)$.

In general, this mechanism is reasonable and suitable for datasets where sample curves are not too closely interlaced. It is also data-driven and self-adaptive. However, for the extremely noisy datasets, we have to use another sampling mechanism for interpolation points, which will be described in Appendix B.3. In this simulation dataset, $n = 62$ and we set $W_1 = -35, W_2 = 35$. The learning rate with respect to Equation (A11) is 0.001 and we repeat Equations (A8)–(A11) for $T' = 10$ times to update b, θ in each iteration. The annealing parameter $\lambda = \lambda(t) = (10 \times (1 - e^{-0.1(t-1)}) + 0.1)^{-1}$. The convergence or stopping criterion for iteration is set by $\Delta H < 1\text{e}^{-6}$. We find out that our algorithm has converged after about 20 iterations, so we set $T = 30$ for simplicity. The total training time is 1422.6 s.

We change the random seed and hyperparameters to conduct multiple parallel experiments to validate the robustness of our algorithm. Since the data suffer very violent fluctuations, our algorithm sometimes yields $K^* = 2$ or 4. In these cases, we find out that the harmony function is trapped in a local maximum. However, by adjusting the learning rate and slowing down the annealing speed, we can still drive the harmony function to reach the global maximum such that $K^* = 3$.

As a supplement, aiming to demonstrate how the classification accuracy of our algorithm changes as these curves from different clusters become "similar", we conduct some extra experiments for this simulation dataset. We adopt the J-divergence as the criterion of measuring the similarity of two clusters (i.e., the corresponding probability distributions), which is given by

$$D_J(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx + \int q(x) \log \frac{q(x)}{p(x)} dx. \tag{A12}$$

We regulate the J-divergence between two clusters by changing the values of the parameters of their MGPFR models (refer to Table 1). The experimental results are shown in Figure A1. When the J-divergence is relatively small, all of the three methods unsurprisingly lead to the same conclusion that there exists only one cluster. However, as it becomes larger, our BYY annealing learning algorithm gradually acquires the capability of automated model selection at a faster speed and obtains higher classification accuracy. On the other hand, when it is large enough, all three methods can distinguish different clusters, with the classification accuracy approaching 100%.

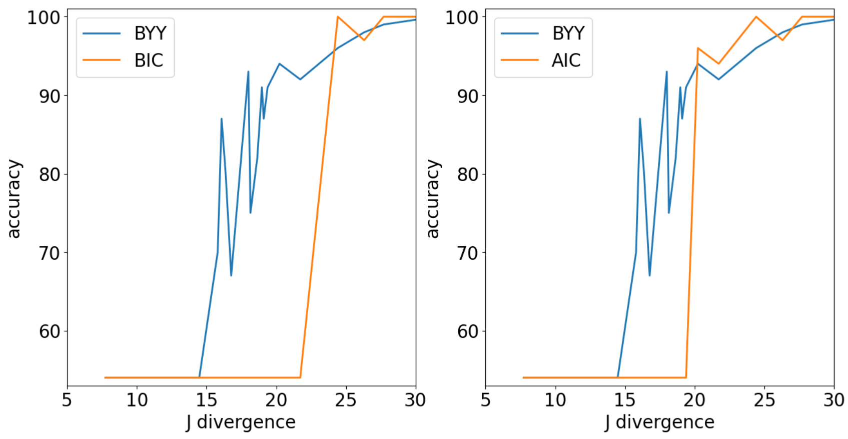


Figure A1. Comparative results of the classification accuracies of the BYY annealing learning, and BIC and AIC methods with the change in the J-divergence between the two clusters.

Appendix B.2. On the City Temperature Dataset

One can download this dataset from the University of Dayton—Environmental Protection Agency Average Daily Temperature Archive: http://academic.udayton.edu/kissock/http/Weather/default.htm (accessed on 1 June 2023). It records the daily average temperature in 324 cities from 1995 to 2020. For simplicity, we selected four cities from them for our experiments.

In this particular dataset, there are $M = 97$ unlabeled curves in total and there are 87 sample points in each curve, i.e., $N = N_1 = \cdots = N_{97} = 87$. The initial values of the parameters are set in the following way: $K_{\text{init}} = 10, \pi_1 = \pi_2 = \cdots = \pi_{K_{\text{init}}} = \frac{1}{K_{\text{init}}}, G = 10, \eta_k \sim \text{Dirichlet}(1, 1, \cdots, 1)$ i.i.d. for $k = 1, 2, \cdots, K_{\text{init}}, \theta_g \equiv (1, 3, 3), \mu_g \sim \text{Uniform}(\{20, 21, \cdots, 68\})$ i.i.d. for $g = 1, 2, \cdots, G, \sigma_g \equiv 10, D = 10, b_g \sim \text{Uniform}(0, \frac{2}{D})$ i.i.d. for $g = 1, 2, \cdots, G$. When selecting the B-spline functions, we use the same method described in Appendix B.1. However, we delete two interpolation points, $(\bar{x}_{d1}, \bar{y}_{d1}), (\bar{x}_{dn}, \bar{y}_{dn})$, because all the sample curves have the identical inputs. The learning rate with respect to Equation (A11) is 0.01 and we repeat Equations (A8)–(A11) for $T' = 10$ times to update b, θ in each iteration. The annealing parameter $\lambda = \lambda(t) = (20 \times (1 - e^{-0.1(t-1)}) + 0.1)^{-1}$. The stopping criterion for iteration is $\Delta H < 1\text{e}^{-6}$. It was found by our experiments that the algorithm converged after about 25 iterations; thus, we set $T = 30$ for simplicity. Actually, the total training time of our algorithm is 2013.9 s.

Likewise, we also change the random seed and hyperparameters and conduct multiple parallel experiments to validate the robustness of our algorithm. Since the noise in this dataset is relatively small, the classification accuracy of our algorithm is generally greater than 96.90% and the average value is 98.97%. Figure 4 shows the experimental results in which the harmony function converges to a greater value than those in the other parallel experiments.

Appendix B.3. On the Effective Radiation Dataset

This dataset was downloaded from the Digital Journal of Global Change Data Repository, https://doi.org/10.3974/geodb.2018.03.02.V1 (accessed on 1 June 2023). It records the daily average surface effective radiation at 130 radiation stations from 1971 to 2014. We selected 20 stations from them located in different cities in China for our experiments.

There are $M = 100$ unlabeled curves in total and in each curve, there are 75 sample points, i.e., $N = N_1 = \cdots = N_{100} = 75$. The initial values of the parameters in our algorithm are set in the following way: $K_{\text{init}} = 10, \pi_1 = \pi_2 = \cdots = \pi_{K_{\text{init}}} = \frac{1}{K_{\text{init}}}, G = 10, \eta_k \sim \text{Dirichlet}(1, 1, \cdots, 1)$ i.i.d. for $k = 1, 2, \cdots, K_{\text{init}}, \theta_g \equiv (1, 5, 2), \mu_g \sim \text{Uniform}(\{20, 21, \cdots, 55\})$ i.i.d. for $g = 1, 2, \cdots, G, \sigma_g \equiv 10, D = 15, b_g \sim \text{Uniform}(0, \frac{2}{D})$ i.i.d. for $g = 1, 2, \cdots, G$. For selecting the B-spline functions, for simplicity, we assume $\phi_1 = \phi_2 = \cdots = \phi_G$. Since this dataset is very noisy, we use an alternative mechanism to select the interpolation points $\{(\bar{x}_d, \bar{y}_d)\}_{d=1}^{D} = \{\{(\bar{x}_{di}, \bar{y}_{di})\}_{i=1}^{n}\}_{d=1}^{D}$:

Step 1: Select a sufficiently small W_1 and a sufficiently large W_2 such that the interval $[W_1, W_2]$ can cover all inputs x;
Step 2: Compute $\bar{y} = \frac{1}{M \sum_{m=1}^{M} N_m} \sum_{m=1}^{M} \sum_{n=1}^{N_m} y_{mn}$;
Step 3: Divide the interval $[W_1, W_2]$ into $R - 1$ subintervals with equal lengths and for $i = 1, \cdots, R$ and $d = 1, \cdots, D$, set $\bar{x}_{di} = W_1 + (W_2 - W_1)(i - 1)/(R - 1)$;
Step 4: Sample $\bar{y}_{di} \sim \text{Uniform}(0, 2\bar{y})$ i.i.d. for $i = 1, \cdots, R$ and $d = 1, \cdots, D$.

For this dataset, $W_1 = 0$, $W_2 = 74$, and $R = 75$. The learning rate with respect to Equation (A11) is 0.01 and we repeat Equations (A8)–(A11) for $T' = 10$ times to update b, θ in each iteration. The annealing parameter $\lambda = \lambda(t) = (10 \times (1 - e^{-0.1(t-1)}) + 0.1)^{-1}$. The stopping criterion for iteration is $\Delta H < 1\text{e}^{-6}$. We find out that our algorithm has converged after about 25 iterations and set $T = 30$ for simplicity. Actually, the total training time of our algorithm is 2104.4 s.

The random seed and hyperparameters are changed several times to conduct parallel experiments for validating the robustness of our algorithm. It is found by the experimental results that the middle-latitude cities Xi'an and Luohe and the mountain city Hengfeng sometimes have different labels. The other cities keep the same classification results. This phenomenon is probably because the middle-latitude cities have hybrid features of high-latitude cities and low-latitude cities, and the mountain cities lie in a high-altitude area, which exposes them to more severe solar radiation. As these training curves are rather difficult to distinguish from our eyes, we can conclude that our algorithm is very powerful and robust in curve clustering.

Appendix B.4. On the Electrical Load Dataset

This dataset was issued by the Northwest China Grid Company. It records daily variation in electrical load in 2009 and 2010. We randomly select 100 curves for training and 50 curves for test. There are no labels for these curves. In each curve there are 96 sample points.

The initial values of the parameters in our BYY annealing learning algorithm are set in the following simple way: $K_{\text{init}} = 10, \pi_1 = \pi_2 = \cdots = \pi_{K_{\text{init}}} = \frac{1}{K_{\text{init}}}, G = 10, \eta_k \sim \text{Dirichlet}(1,1,\cdots,1)$ i.i.d. for $k = 1,2,\cdots,K_{\text{init}}, \theta_g \equiv (1,0.1,1), \mu_g \sim \text{Uniform}(\{1,2,\cdots,94\})$ i.i.d. for $g = 1,2,\cdots,G, \sigma_g \equiv 10, D = 30, b_g \sim \text{Uniform}(0,\frac{2}{D})$ i.i.d. for $g = 1,2,\cdots,G$. For selecting the B-spline functions, we use the same method described in Appendix B.2. The learning rate with respect to Equation (A11) is 0.003 and we repeat Equations (A8)–(A11) for $T' = 10$ times to update b, θ in each iteration. The annealing parameter $\lambda = \lambda(t) = (20 \times (1 - e^{-0.1(t-1)}) + 0.1)^{-1}$. The stopping criterion for iteration is $\Delta H < 1\text{e}^{-6}$. As we find out that our algorithm has converged after about 20 iterations, we simply set $T = 30$. We spend 2808.4s on training the model. The clustering results are shown in Figure 6.

As for curve prediction, our strategy is rolling or successive prediction, i.e., when our algorithm has already predicted y^*_{mn} for a new input x^*_{mn} and we need to predict y^{**}_{mn} for an another new input x^{**}_{mn}, we regard (x^*_{mn}, y^*_{mn}) as known and use its values along with the previous data together to predict the next step output y^{**}_{mn} and so on. The overall prediction results are listed in Table 3. We can find out that the RMSE decreases at a much slower rate when $U \geq 40$. According to the bias–variance decomposition of MSE, $\mathbb{E}[(y - \hat{f}(x))^2|x] = \sigma^2 + \text{bias}^2 + \text{variance}$, most of the error can be interpreted as the intrinsic noise in this dataset.

The total times of the prediction over 50 test curves with different U are listed in Table A1.

Table A1. Total times of the prediction of $96 - U$ sample points over 50 test curves with different U.

U	20	30	40	50	60	70
T	338.46 s	298.47 s	262.24 s	217.99 s	175.34 s	127.48 s

In particular, we compute the RMSEs of the first point prediction for different U, which are listed in Table A2. Clearly, the first point is the unique one that the rolling prediction method is not actually applied to. We can find out that the RMSE of the first point prediction is remarkably less than the corresponding mean error of the rolling prediction for all U. Specifically, when $U \in [40, 70]$, the RMSE of the first point prediction is relatively high, mainly because the electrical load suffers the greater noise at the peak of power consumption and smaller noise at the trough.

Table A2. RMSEs of the first point prediction with different U.

U	20	30	40	50	60	70
RMSE	0.9212	1.0749	1.5601	1.9240	1.7066	1.7500

We further plot nine prediction curves with the corresponding 95% confidence intervals when $U = 50$, which are shown in Figure A2. Black curves are the real sample curves, red curves are the prediction curves, and blue domains represent the confidence intervals. Although there exists certain divergence, our algorithm still yields excellent prediction results, which further demonstrates that our algorithm is effective for curve prediction.

To investigate the sensitivity of the tuning parameter $\lambda(t)$ on the performance of curve clustering and prediction, we conduct several experiments of our BYY annealing learning algorithm with the hyperparameters u and v in $\lambda(t)$ taking different values. The experimental results are listed in Tables A3 and A4. It can be found that although the clustering results might change slightly, the prediction results are relatively stable and robust.

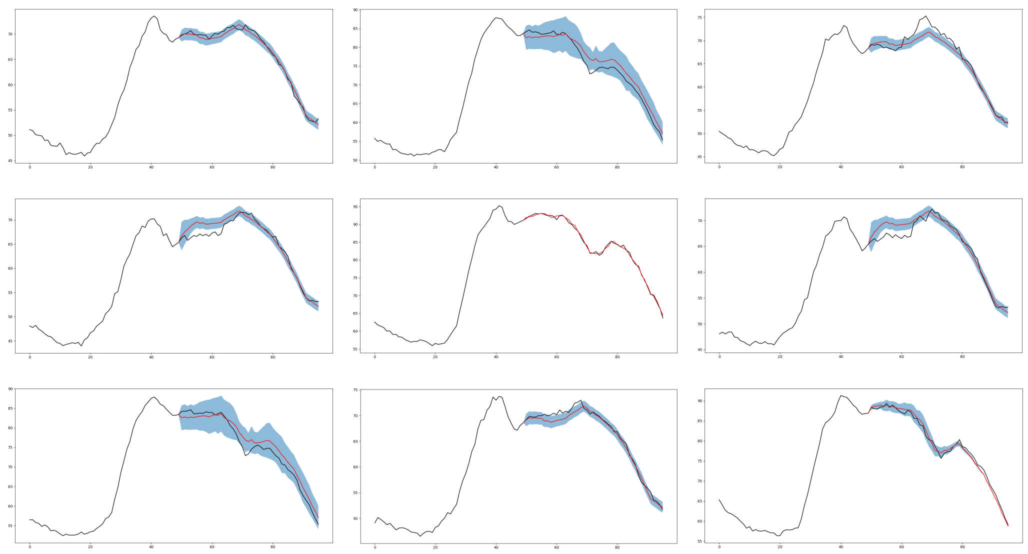


Figure A2. Nine prediction curves and their corresponding 95% confidence intervals of the electrical load dataset when $U = 50$.

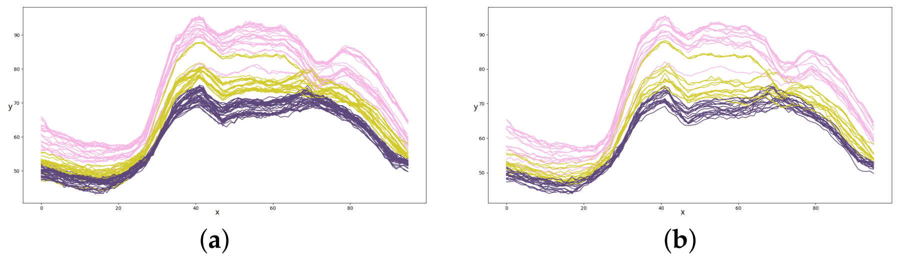
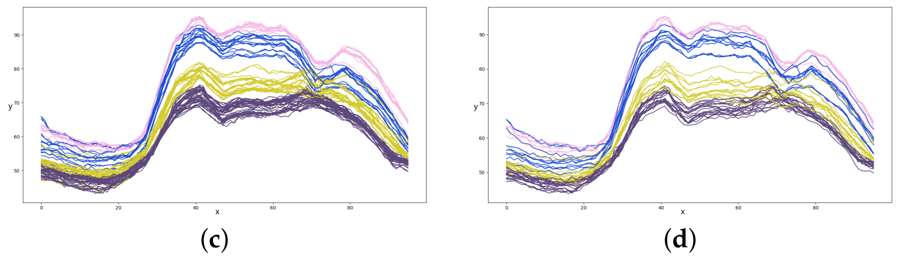
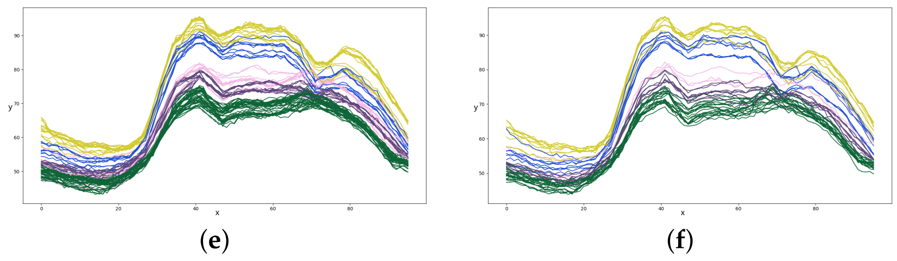




Figure A3. Clustering results of AIC, BIC, and *k*-fold CV for the electrical load dataset. (**a**) Clustering result of AIC and BIC on the training set. (**b**) Clustering result of AIC and BIC on the test set. (**c**) Clustering result of 5-fold CV on the training and validation sets. (**d**) Clustering result of 5-fold CV on the test set. (**e**) Clustering result of 10-fold CV on the training and validation sets. (**f**) Clustering result of 10-fold CV on the test set.

Table A3. Curve clustering results of our BYY annealing learning algorithm with different values of the hyperparameters in $\lambda(t) = \frac{1}{u(1-e^{-v(t-1)})+0.1}$ when $D = 30, G = 10$.

K^* \ u / v	**5**	**10**	**15**	**20**	**25**
0.05	6	6	6	6	6
0.1	6	6	5	5	5
0.5	5	5	5	5	5
1.0	5	5	5	5	5
5.0	5	5	5	5	5

Table A4. Curve prediction results of our BYY annealing learning algorithm with different values of the hyperparameters in $\lambda(t) = \frac{1}{u(1-e^{-v(t-1)})+0.1}$ when $D = 30, G = 10$. When v is too small, the RMSE becomes relatively large mainly due to a lower convergence rate.

RMSE \ u / v	**5**	**10**	**15**	**20**	**25**
0.05	2.01 ± 0.25	2.20 ± 0.23	2.02 ± 0.21	1.87 ± 0.17	1.99 ± 0.18
0.1	1.97 ± 0.18	1.96 ± 0.17	2.01 ± 0.18	1.99 ± 0.17	1.80 ± 0.19
0.5	1.95 ± 0.18	1.79 ± 0.18	1.81 ± 0.20	1.80 ± 0.19	1.79 ± 0.18
1.0	1.81 ± 0.20	1.81 ± 0.20	1.80 ± 0.19	1.78 ± 0.16	1.79 ± 0.16
5.0	1.81 ± 0.19	1.80 ± 0.19	2.18 ± 0.25	1.79 ± 0.16	1.79 ± 0.16

Appendix C. Further Discussion of the Parameters

In this final part, we discuss the selection of the hyperparameters and the initialization of the common parameters. As is well-known, they both play an important role in searching for the global maximum of the BYY harmony function.

Appendix C.1. The Hyperparameters

In our model and algorithm, the hyperparameters are K, G, D, ϕ and u, v, w. First, K should be set to be larger than the true number of actual curve clusters in the dataset so that the automated model selection can be made. As for G, it should be adaptive to the number of sample points in the training curves. According to our further experiments, G can be selected from the interval $[\frac{\bar{N}}{10}, \frac{\bar{N}}{6}]$ where $\bar{N} := \frac{1}{M}\sum_{m=1}^{M} N_m$. Furthermore, a larger G results in a more careful and precise classifier, but at the same time it tends to split the input space into more subspaces, which is likely to reduce the correlation among different subspaces and worsen the ability of prediction in time series. As for D, in order to distinguish these GPFRs in the lower layer, it is necessary to set $D \geq G$. In addition, we suggest $D \leq \min(3G, \frac{M}{3})$ to avoid the overfitting. For ϕ, we have already proposed two methods as described in Appendices B.1 and B.3. The former is feasible for the normal datasets and the latter suits the noisy ones.

The hyperparameters u, v, w control the speed of annealing. We have already discussed how they effect the performance of curve clustering and prediction in Sections 3.2 and 6.2.

Appendix C.2. The Other Learnable Parameters

The other parameters are, respectively, $\pi, \eta, \theta, b, \mu, \sigma$, which are learned by the BYY annealing learning algorithm from the training data. Actually, the initialization of these parameters is also important for the good convergence of the algorithm. In fact, we have already described their initialization methods. Here, we further discuss them regarding certain aspects.

As for π, η, setting $\pi_{k,\text{init}} = \frac{1}{K}$ for all $k = 1, 2, \cdots, K$ and $\eta_{k,\text{init}} \sim \text{Dirichlet}(1, 1, \cdots, 1)$ i.i.d. for $k = 1, 2, \cdots, K$ is very natural. As for θ, θ_2 describes the strength of association among the inputs, and θ_1, θ_3 describes the strength of the noise. For those sample curves that have a strong correlation between the past and the future, the smaller $\theta_{2,\text{init}}$ is recommended. For the noisier datasets, θ_1, θ_3 should be initialized with the larger values. As for b, we set $b_{\text{init}} \sim \text{Uniform}(0, \frac{2}{D})$, with the goal of letting $\mathbb{E}\left(\sum_{d=1}^{D} \phi_d(x) b_d\right)$ approximate the spatial scale of the given data. Furthermore, for μ, σ, their initial values can be determined by the range of the input variables of the sample curves. μ_{init} is randomly sampled from the integers in this range and selecting a larger σ_{init} can make the algorithm more robust to outliers.

References

1. Ketu, S.; Mishra, P.K. Enhanced Gaussian process regression-based forecasting model for COVID-19 outbreak and significance of IoT for its detection. *Appl. Intell.* **2021**, *51*, 1492–1512. [CrossRef] [PubMed]
2. Zheng, Y.; Yang, Y.; Che, T.; Hou, S.; Huang, W.; Gao, Y.; Tan, P. Image Matting with Deep Gaussian Process. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–15. [CrossRef] [PubMed]
3. Fradi, A.; Samir, C.; Yao, A.F. Manifold-based inference for a supervised Gaussian process classifier. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 4239–4243.
4. Hebbal, A.; Brevault, L.; Balesdent, M.; Talbi, E.G.; Melab, N. Bayesian optimization using deep Gaussian processes with applications to aerospace system design. *Optim. Eng.* **2021**, *22*, 321–361. [CrossRef]
5. Tresp, V. Mixtures of Gaussian processes. *Adv. Neural Inf. Process. Syst.* **2000**, *13*.
6. Shi, J.Q.; Wang, B.; Murray-Smith, R.; Titterington, D. Gaussian process functional regression modeling for batch data. *Biometrics* **2007**, *63*, 714–723. [CrossRef]
7. Li, T.; Wu, D.; Ma, J. Mixture of robust Gaussian processes and its hard-cut EM algorithm with variational bounding approximation. *Neurocomputing* **2021**, *452*, 224–238. [CrossRef]
8. Zhang, X.; Song, C.; Zhao, J.; Deng, X. Domain Adaptation Mixture of Gaussian Processes for Online Soft Sensor Modeling of Multimode Processes When Sensor Degradation Occurs. *IEEE Trans. Ind. Inform.* **2022**, *18*, 4654–4664. [CrossRef]
9. Liu, J.; Liu, T.; Chen, J. Sequential local-based Gaussian mixture model for monitoring multiphase batch processes. *Chem. Eng. Sci.* **2018**, *181*, 101–113. [CrossRef]
10. Chen, P.; Zabaras, N.; Bilionis, I. Uncertainty propagation using infinite mixture of Gaussian processes and variational Bayesian inference. *J. Comput. Phys.* **2015**, *284*, 291–333. [CrossRef]
11. Schwaighofer, A.; Tresp, V.; Yu, K. Learning Gaussian process kernels via hierarchical Bayes. *Adv. Neural Inf. Process. Syst.* **2004**, *17*.
12. Tayal, A.; Poupart, P.; Li, Y. Hierarchical Double Dirichlet Process Mixture of Gaussian Processes. *Proc. Aaai Conf. Artif. Intell.* **2012**, *26*, 1126–1133. [CrossRef]
13. Pandita, P.; Tsilifis, P.; Ghosh, S.; Wang, L. Scalable Fully Bayesian Gaussian Process Modeling and Calibration with Adaptive Sequential Monte Carlo for Industrial Applications. *J. Mech. Des.* **2021**, *143*, 074502. [CrossRef]
14. Chen, Z.; Ma, J.; Zhou, Y. A Precise Hard-Cut EM Algorithm for Mixtures of Gaussian Processes. In Proceedings of the International Conference on Intelligent Computing, Taiyuan, China, 3–6 August 2014.
15. Li, T.; Ma, J. Dirichlet process mixture of Gaussian process functional regressions and its variational EM algorithm. *Pattern Recognit.* **2023**, *134*, 109–129. [CrossRef]
16. Seitz, S. Mixtures of Gaussian Processes for regression under multiple prior distributions. *arXiv* **2021**, arXiv:2104.09185.
17. Luo, W.; Sycara, K. Adaptive sampling and online learning in multi-robot sensor coverage with mixture of gaussian processes. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 6359–6364.
18. Teckentrup, A.L. Convergence of Gaussian Process Regression with Estimated Hyper-parameters and Applications in Bayesian Inverse Problems. *SIAM/ASA J. Uncertain. Quantif.* **2019**, *8*, 1310–1337. [CrossRef]
19. Pang, Y.; Zhou, X.; He, W.; Zhong, J.; Hui, O. Uniform Design–Based Gaussian Process Regression for Data-Driven Rapid Fragility Assessment of Bridges. *J. Struct. Eng.* **2021**, *147*, 04021008. [CrossRef]
20. Shi, J.Q.; Murray-Smith, R.; Titterington, D.M. Hierarchical Gaussian process mixtures for regression. *Stat. Comput.* **2005**, *15*, 31–41. [CrossRef]
21. Shi, J.Q.; Wang, B. Curve prediction and clustering with mixtures of Gaussian process functional regression models. *Stat. Comput.* **2008**, *18*, 267–283. [CrossRef]
22. Wu, D.; Ma, J. A two-layer mixture model of Gaussian process functional regressions and its MCMC EM algorithm. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 4894–4904. [CrossRef]
23. Xu, L. BYY harmony learning, structural RPCL, and topological self-organizing on mixture models. *Neural Netw.* **2002**, *15*, 1125–1151. [CrossRef] [PubMed]

24. Ferguson, T.S. A Bayesian analysis of some nonparametric problems. *Ann. Stat.* **1973**, *1*, 209–230. [CrossRef]
25. Green, P.J. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* **1995**, *82*, 711–732. [CrossRef]
26. Görür, D.; Edward Rasmussen, C. Dirichlet process gaussian mixture models: Choice of the base distribution. *J. Comput. Sci. Technol.* **2010**, *25*, 653–664. [CrossRef]
27. Zhang, Z.; Chan, K.L.; Wu, Y.; Chen, C. Learning a multivariate Gaussian mixture model with the reversible jump MCMC algorithm. *Stat. Comput.* **2004**, *14*, 343–355. [CrossRef]
28. Xu, L. Bayesian ying yang learning. *Scholarpedia* **2007**, *2*, 1809. [CrossRef]
29. Zhao, L.; Ma, J. A dynamic model selection algorithm for mixtures of Gaussian processes. In Proceedings of the 2016 IEEE 13th International Conference on Signal Processing (ICSP), Chengdu, China, 6–10 November 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1095–1099.
30. Ma, J.; Liu, J. The BYY annealing learning algorithm for Gaussian mixture with automated model selection. *Pattern Recognit.* **2007**, *40*, 2029–2037. [CrossRef]
31. Xu, L. Best harmony, unified RPCL and automated model selection for unsupervised and supervised learning on Gaussian mixtures, three-layer nets and ME-RBF-SVM models. *Int. J. Neural Syst.* **2001**, *11*, 43–69. [CrossRef]
32. Nguyen, T.; Bonilla, E. Fast allocation of Gaussian process experts. In Proceedings of the International Conference on Machine Learning, Beijing, China, 22–24 June 2014; pp. 145–153.
33. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
34. van den Oord, A.; Kalchbrenner, N.; Kavukcuoglu, K. Pixel Recurrent Neural Networks. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016.
35. van den Oord, A.; Kalchbrenner, N.; Espeholt, L.; Kavukcuoglu, K.; Vinyals, O.; Graves, A. Conditional Image Generation with PixelCNN Decoders. *arXiv* **2016**, arXiv:1606.05328.
36. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.C.; Bengio, Y. Generative Adversarial Nets. In Proceedings of the NIPS, Montreal, QC, Canada, 8–13 December 2014.
37. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein GAN. *arXiv* **2017**, arXiv:1701.07875.
38. Ueda, N.; Nakano, R. Deterministic annealing EM algorithm. *Neural Netw.* **1998**, *11*, 271–282. [CrossRef] [PubMed]