Neurocomputing 331 (2019) 366–374

Contents lists available at ScienceDirect

Neurocomputing

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

An effective EM algorithm for mixtures of Gaussian processes via the MCMC sampling and approximation

Di Wu, Jinwen Ma*

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

ARTICLE INFO

Article history:
Received 31 July 2017
Revised 2 October 2018
Accepted 13 November 2018
Available online 23 November 2018

Communicated by zhi yong Liu

Keywords:
Mixture of Gaussian processes
EM algorithm
MCMC sampling
Multimodal data
Classification
Prediction

ABSTRACT

The Mixture of Gaussian Processes (MGP) is a powerful statistical model for characterizing multimodal data, but its conventional Expectation-Maximization (EM) algorithm (Dempster et al., 1977) is computationally intractable because of its time complexity. To solve this problem, some approximation techniques have been proposed in the conventional EM algorithm. However, these approximate EM algorithms are ineffective or limited in some situations. To implement the EM algorithm more effectively, we approximate the EM algorithm with simulated samples of latent variable via the Monte Carlo Markov Chain (MCMC) sampling, and design an MCMC EM algorithm. Experiments on both synthetic and real-world data sets demonstrate that our MCMC EM algorithm is more effective than the state-of-the-art EM algorithms on classification and prediction problems.

1. Introduction

Gaussian Process (GP) is a powerful model in machine learning and computer vision [1,2]. However, it is ineffective for a single Gaussian process to model a multimodal data set along its input region. To overcome this difficulty, Tresp [3] suggested the Mixture of GPs (MGP) that takes the architecture of mixture of experts [4,5]. As such, the MGP model can be considered as a combination of several GPs by a gating network or function. However, it can learn temporal information and predict time series because these GPs take a temporal structure. So, the MGP model has been investigated and developed greatly in recent years beyond the conventional mixture of experts.

As sample points of the MGP model are not independent, its parameter learning is a challenging problem. To date, there are three major approaches for parameter learning, i.e., the Markov Chain Monte Carlo (MCMC) method [6], the Variational Bayesian (VB) inference and the Expectation-Maximization (EM) algorithm. Specifically, the MCMC method often achieves an accurate result but requires more convergence time, while the VB inference is always quickly implemented but ineffective. As a powerful tool, the EM algorithm has been intensively applied for MGPs [7,8]. In fact, there are four EM algorithms, i.e., heuristic EM, variational EM, leave-one-out-cross-validation (LOOCV) EM and hard-cut EM algorithms. Although these EM algorithms work well in some situations, they still have certain limitations.

To get rid of above difficulties, we recently designed an MCMC EM algorithm with merits of MCMC sampling and EM algorithm in [9]. In fact, the MCMC EM algorithm was proposed twenty years ago, but it had not been adopted for the learning of the MGP model. In this paper, we extend the study of the MCMC EM algorithm for MGPs methodologically and theoretically. The research background and related work are systematically analyzed and the MCMC EM algorithm is mathematically derived and carefully designed. Moreover, we conduct systematic experiments on both synthetic and real-world data sets to demonstrate that the MCMC EM algorithm outperforms the state-of-the-art EM algorithms on both classification and prediction.

The paper is organized as follows. Some related works are given in Section 2. For clarity, the GP and MGP models are revisited and analyzed in Section 3. In Section 4, we present the MCMC EM algorithm for MGPs. Experimental results are shown in Section 5. Finally, we conclude the paper in Section 6.

2. Related works

The gating function for MGPs: The gating function of the MGP model has three forms, i.e., the logistic distribution [3], Gaussian distribution [10] and Gaussian mixture distribution or model (GMM) [11]. The logistic distribution gating function inherits from

* Corresponding author.
E-mail addresses: wudi2@snnu.edu.cn (D. Wu), jwma@math.pku.edu.cn, jwma@pku.edu.cn (J. Ma).

https://doi.org/10.1016/j.neucom.2018.11.046
0925-2312/

the conventional mixture of experts directly, but it is not effective for MGPs. The GMM gating function is complex and thus only adopted for certain complex data sets. The Gaussian distribution gating function is effective yet not complex, so it has been used widely.

The MCMC method and VB inference for MGPs: The *MCMC method* approximates the intractable integration or summation by using a large set of simulated samples generated from a posterior probability distribution [10,12–14]. In theory, it can achieve an accurate result with a large number of simulated samples, but the generation of these simulated samples requires more time. The *VB inference* generally approximates the complex posterior probability distribution via the mean field approximation method [15]. As parameters and indicators are assumed to be independent, the VB inference is fast, but it is generally not effective since the approximated distribution is inconsistent with the real one.

The EM algorithm for MGPs: The *conventional EM algorithm* [8] is a popular iterative method to learn the maximum likelihood estimation for a statistical model with latent variables. In each iteration, there are two major steps, i.e., E-step and M-step. In the E-step, Q-function is obtained by calculating the expectation of the log likelihood with respect to the latent variables. In the M-step, Q-function is further maximized to update parameters. For MGPs, the indicatory variables of samples with respect to GPs are regarded as latent variables. To date, all existing EM algorithms for MGPs are approximations of the conventional EM algorithm since the time complexity of the conventional EM algorithm is of exponential order. Although it is efficient, the *heuristic EM* algorithm is very coarse because the update of parameters is not based on the Q-function [3,8]. The *variational EM algorithm* approximates the distribution of all parameters and indicators by the mean field method, so it shares the same weakness as the VB inference [11,16,17]. The LOOCV EM algorithm approximates the distribution of a Gaussian process with the LOOCV probability decomposition that makes the calculation of the Q-function feasible [18]. Although the LOOCV EM algorithm is more effective, it cannot achieve accurate results due to the similar reason of the variational EM algorithm. The *hard-cut EM algorithm*, also known as the classification EM algorithm [19], approximates the Q-function by the total log likelihood function, whose indicators are calculated in a hard-cut or classification way according to the *maximum a posteriori* (MAP) principle. The hard-cut EM algorithm for MGPs is more accurate since there is no probability decomposition of any Gaussian process [20–22]. However, it is easy to be trapped into a local maximum of the log likelihood function.

The MCMC EM algorithm: In each iteration of the EM algorithm, the MCMC sampling method is used to obtain sufficient simulated samples of latent variables so that a good estimator of the Q-function can be calculated directly [23,24]. In this way, the EM algorithm is implemented efficiently. From the viewpoint of statistics, the MCMC approximation is more precise than the above approximations for the Q-function as long as the number of required simulated samples is sufficiently large. It is clear that the MCMC EM algorithm is a special Monte Carlo EM (MCEM) algorithm. Mathematically, the MCEM algorithm is usually regarded as a stochastic version of the hard-cut EM algorithm, and the hard-cut EM algorithm is regarded as a deterministic version of the MCEM algorithm.

3. The GP and MGP models

We revisit Gaussian process (GP) and mixture of GPs (MGP). For MGPs, the crucial problem for establishing the EM algorithm is analyzed.

3.1. The GP model

The Gaussian process is a widely used stochastic process in which any group of states (or outputs) is subject to a Gaussian distribution [1,2]. For a data set $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$, where x_n and y_n are a pair of input and output (variables) at time n, the general Gaussian process is defined as follows. If the output vector $[y_1, \ldots, y_N]^T$ is subject to an N-dimensional Gaussian distribution[1], denoted by $\mathcal{N}(\mathbf{0}, \mathbf{C})$, then $[y_1, \ldots, y_N]^T$ follow a Gaussian process, where $\mathbf{C} = \left[c\left(x_n, x_{n'}; \boldsymbol{\theta}\right)\right]_{N\times N}$ is an $N \times N$ covariance matrix in which $c\left(x_n, x_{n'}; \boldsymbol{\theta}\right)$ is a covariance function between two inputs.

There are many kinds of covariance functions for the GP model, and the squared exponential covariance function is widely used. Actually, the squared exponential covariance function is stationary, that is, it is a function of $x_n - x_{n'}$. The squared exponential covariance function is a type of the Radial Basis Function (RBF) covariance function. Mathematically, it takes the following form:

$$c\left(x_n, x_{n'}; \boldsymbol{\theta}\right) = \theta_1 \exp\left[-\theta_2 (x_n - x_{n'})^2/2\right] + \theta_3 \delta_{nn'},$$

where $\delta_{nn'}$ is the Kronecker delta function, $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]$ with θ_3 controlling the noise variance globally. If two inputs x_n and $x_{n'}$ are close within the input region, then the corresponding outputs y_n and $y_{n'}$ are more correlated. To date, many algorithms have been established to learn the parameter $\boldsymbol{\theta}$ of the GP model, such as the gradient-ascent maximum likelihood, expectation propagation, Laplace's approximation and variational bounds [25,26]. All of these methods usually achieve good results on parameter estimation, so we adopt the simplest one, i.e., the gradient-ascent maximum likelihood algorithm in our learning paradigm. In the GP model, the dimension of the Gaussian distribution is the number of samples N. As a result, the time complexity of parameter estimation for the GP model is generally $O(N^3)$.

3.2. The MGP model

A single GP cannot characterize a multimodal data set along the input region because the structure of the GP model is rather simple. However, there are many multimodal data sets available in practical applications. To overcome this difficulty, we extend the single GP model to an MGP model in which different components or segments are involved along the input region and each component is subject to a GP model independently [3]. The gating network combines these predictive Gaussian processes together along the input region and we select the gating functions as certain Gaussian distributions. For simplicity, we still denote the data set of the MGP model by $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$, and describe the detail of the MGP model as follows.

First, we introduce the association of the nth sample with respect to K components by an unknown indicator, as denoted by z_{nk}. If the nth sample belongs to the kth component, then $z_{nk} = 1$; otherwise, $z_{nk} = 0$. Let $\mathbf{z}_n = [z_{n1}, \ldots, z_{nK}]^T$ denote a vector of indicators; it is subject to

$$P(\mathbf{z}_n = \mathbf{e}_k) = \pi_k, \tag{1}$$

where $\mathbf{e}_k$ is the kth column of a $K \times K$ unit matrix and $\sum_{k=1}^K \pi_k = 1$.

[1] The Gaussian distribution is a foundation of GP and MGP models. The density function of the d-dimensional Gaussian distribution is mathematically expressed by

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right],$$

where d is the dimension of $\mathbf{x}$. In addition, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are a d-dimensional mean vector and a $d \times d$ covariance matrix, respectively. For simplicity, this Gaussian distribution is denoted by $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Second, let the corresponding input x_n be subject to a Gaussian distribution

$$x_n|(\mathbf{z}_n = \mathbf{e}_k) \sim \mathcal{N}\left(\mu_k, \sigma_k^2\right), \tag{2}$$

where μ_k and σ_k are the mean and standard deviation of the Gaussian distribution, respectively. As the input of each component is subject to a Gaussian distribution, the input of all K components is actually subject to a Gaussian mixture model (GMM) (or mixture of Gaussian distributions). Generally, the input distributions are consistent with the GMM in practical applications. Even if the input distributions are not similar to the GMM, the GMM can still learn them well. Thus, the GMM is very flexible for learning of an input distribution. For example, the input of data set in Section 5.2 is nearly subject to a uniform distribution, but the GMM can still learn it well. On the other hand, the means $\mu_1, \mu_2, \ldots, \mu_K$ should be set differently so that these GP models can be located in different input regions. Two distributions of Eqs. (1) and (2) are combined together to form the gating network of our MGP model.

We further denote the component column vector of y_n belonging to the kth component by $\mathbf{y}_k = [y_n|\mathbf{z}_n = \mathbf{e}_k; n = 1, \ldots, N]$, being subject to a GP model as follows:

$$\mathbf{y}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_k), \tag{3}$$

where $\mathbf{C}_k = \left[c\left(x_n, x_{n'}; \boldsymbol{\theta}_k\right)|\mathbf{z}_n = \mathbf{z}_{n'} = \mathbf{e}_k; n, n' = 1, \cdots, N\right]$ is the covariance matrix of the kth component. With these specific GP components, we get the MGP model via the above gating network.

If prior probabilities are given for certain parameters, then the MGP model is a Bayesian model. But these prior probabilities offer no significant advantage on our learning task. Therefore, we do not use the Bayesian model for MGPs. As a result, the information flow direction of the MGP model is $\mathbf{z}_n \to x_n \to y_n$, which is reasonable for a general regression model. Sketches of four synthetic data sets generated from four typical MGP models are illustrated in Fig. 1, respectively, in which sample points of each component are represented in the same color.

Finally, we have the total log likelihood function of the MGP model on the given data set

$$\mathrm{L}(\Theta, \mathbf{Z}) = \sum_{k=1}^{K}\left\{\sum_{n=1}^{N} z_{nk}[\ln \pi_k + \ln p(x_n|\mu_k, \sigma_k)] + \ln p\left(\mathbf{y}_k|\mathbf{x}_k, \boldsymbol{\theta}_k\right)\right\}, \tag{4}$$

where $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N]$ is a $K \times N$ indicator matrix, $\Theta = \left\{\pi_k, \mu_k, \sigma_k, \boldsymbol{\theta}_k\right\}_{k=1}^{K}$ denotes the whole parameter set, and $\mathbf{x}_k$ is defined in the same way as $\mathbf{y}_k$. Methodologically, we use the EM algorithm to obtain the maximum likelihood estimation iteratively. But in this case, $[\mathbf{y}_k]_1, [\mathbf{y}_k]_2, \ldots, [\mathbf{y}_k]_{\sum_{n=1}^{N} z_{nk}}$ are strongly dependent, which is a challenging problem for the establishment of an effective EM algorithm for MGPs.

4. MCMC EM algorithm

We first derive the $\hat{Q}$-function as the MCMC approximated Q-function for MGPs and establish the method for maximizing the $\hat{Q}$-function. Next, we design the MCMC EM algorithm and its overall MCMC prediction strategy.

4.1. Derivation and maximization of $\hat{Q}$-function

4.1.1. Derivation of $\hat{Q}$-function

We derive the $\hat{Q}$-function in the following two steps. (1). The Q-function of the conventional EM algorithm is derived for MGPs; (2). We further obtain the $\hat{Q}$-function. The indicators of samples with respect to the components of the MGP model are treated as latent variables of conventional EM and MCMC EM algorithms. For the conventional EM algorithm, the Q-function is given by

$$\mathrm{Q}\left(\Theta|\hat{\Theta}\right) = \mathbb{E}_{\mathbf{Z}}\left[\mathrm{L}(\Theta, \mathbf{Z})|\mathcal{D}, \hat{\Theta}\right] = \sum_{\mathbf{Z}} P\left(\mathbf{Z}|\mathcal{D}, \hat{\Theta}\right)\mathrm{L}(\Theta, \mathbf{Z}), \tag{5}$$

where $\mathrm{L}(\Theta, \mathbf{Z})$ is the total log likelihood function given by Eq. (4). Unfortunately, the number of possible values of $\mathbf{Z}$ is K^N and therefore the time complexity of calculating the Q-function value is of exponential order of the number of sample points. As a result, the conventional EM algorithm is of exponential order of the number of sample points, and the time consumption becomes extremely large even if $N \geq 50$.

To overcome the time complexity, the variational, LOOCV and hard-cut mechanisms have been proposed to approximate the Q-function. Moreover, calculating these approximated Q-function values are of polynomial order. Thus, the variational EM, LOOCV EM and hard-cut EM algorithms have a complexity of polynomial order. However, all these approximations are heuristic and even quite coarse. In contrast, the Monte Carlo (MC) or MCMC sampling provide a reasonable way to approximate the Q-function. Furthermore, computing the MC or MCMC approximated Q-function value is of polynomial order, and therefore the MCEM or MCMC EM algorithm is more efficient. By replacing every z_{nk} of $\mathbf{Z}$ with a simulated z_{nki}, we define an indicatory matrix $\mathbf{Z}_i$ for $i = 0, 1, 2, \ldots$. Then, $\mathbf{Z}_1, \mathbf{Z}_2, \ldots$ are simulated samples generated by MC or MCMC sampling, with $\mathbf{Z}_0$ being the initial state of the MCMC sampling. Also, the other notation about the ith simulated sample is denoted in the same way as $\mathbf{Z}_i$. The first step of MC approximation is to generate a series of simulated samples, $\mathbf{Z}_1, \mathbf{Z}_2, \ldots$, from the target distribution $P\left(\mathbf{Z}|\mathcal{D}, \hat{\Theta}\right)$ in Eq. (5). However, a lot of MC samplings do not perform well in this case because of the high dimension of $\mathbf{Z}$.

To generate a set of simulated samples, $\mathbf{Z}_1, \mathbf{Z}_2, \ldots$, we utilize the MCMC sampling, which is a special MC sampling in the generation of high-dimensional simulated samples. In the MCMC procedure, we construct an irreducible Markov chain, $\mathbf{Z}_0, \mathbf{Z}_1, \mathbf{Z}_2, \ldots$, whose stationary distribution is our target distribution $P\left(\mathbf{Z}|\mathcal{D}, \hat{\Theta}\right)$. Generally, there exists a reasonable number I_0 such that simulated samples, $\mathbf{Z}_{I_0+1}, \mathbf{Z}_{I_0+2}, \ldots$, are nearly subject to $P\left(\mathbf{Z}|\mathcal{D}, \hat{\Theta}\right)$. In fact, $\mathbf{Z}_1, \ldots, \mathbf{Z}_{I_0}$ may not forget the initial state $\mathbf{Z}_0$. Therefore, we can discard these previous simulated samples during MCMC sampling. By selecting a proper number I, we get the $\hat{Q}$-function, i.e., the MCMC approximated Q-function, by

$$\begin{aligned}\hat{\mathrm{Q}}\left(\Theta|\hat{\Theta}\right) &= \frac{1}{\mathrm{I}}\sum_{\mathrm{i}=\mathrm{I}_0+1}^{\mathrm{I}_0+\mathrm{I}} \mathrm{L}(\Theta, \mathbf{Z}_\mathrm{i}) \\ &= \frac{1}{I}\sum_{i=I_0+1}^{I_0+I}\sum_{k=1}^{K}\left\{\sum_{n=1}^{N} z_{nki}[\ln \pi_k + \ln p(x_n|\mu_k, \sigma_k)] + \ln p\left(\mathbf{y}_{ki}|\mathbf{x}_{ki}, \boldsymbol{\theta}_k\right)\right\}.\end{aligned} \tag{6}$$

4.1.2. Maximization of $\hat{Q}$-function

By setting the derivatives of the $\hat{Q}$-function with respect to π_k, μ_k and σ_k be zero, we obtain the analytical maximums of π_k, μ_k and σ_k directly:

$$\pi_k = \frac{N_k}{\sum_{k=1}^{K} N_k}, \tag{7}$$

$$\mu_k = \frac{1}{N_k}\sum_{i=I_0+1}^{I_0+I}\sum_{n=1}^{N} z_{nki} x_n, \tag{8}$$

$$\sigma_k^2 = \frac{1}{N_k}\sum_{i=I_0+1}^{I_0+I}\sum_{n=1}^{N} z_{nki}(x_n - \mu_k)^2, \tag{9}$$

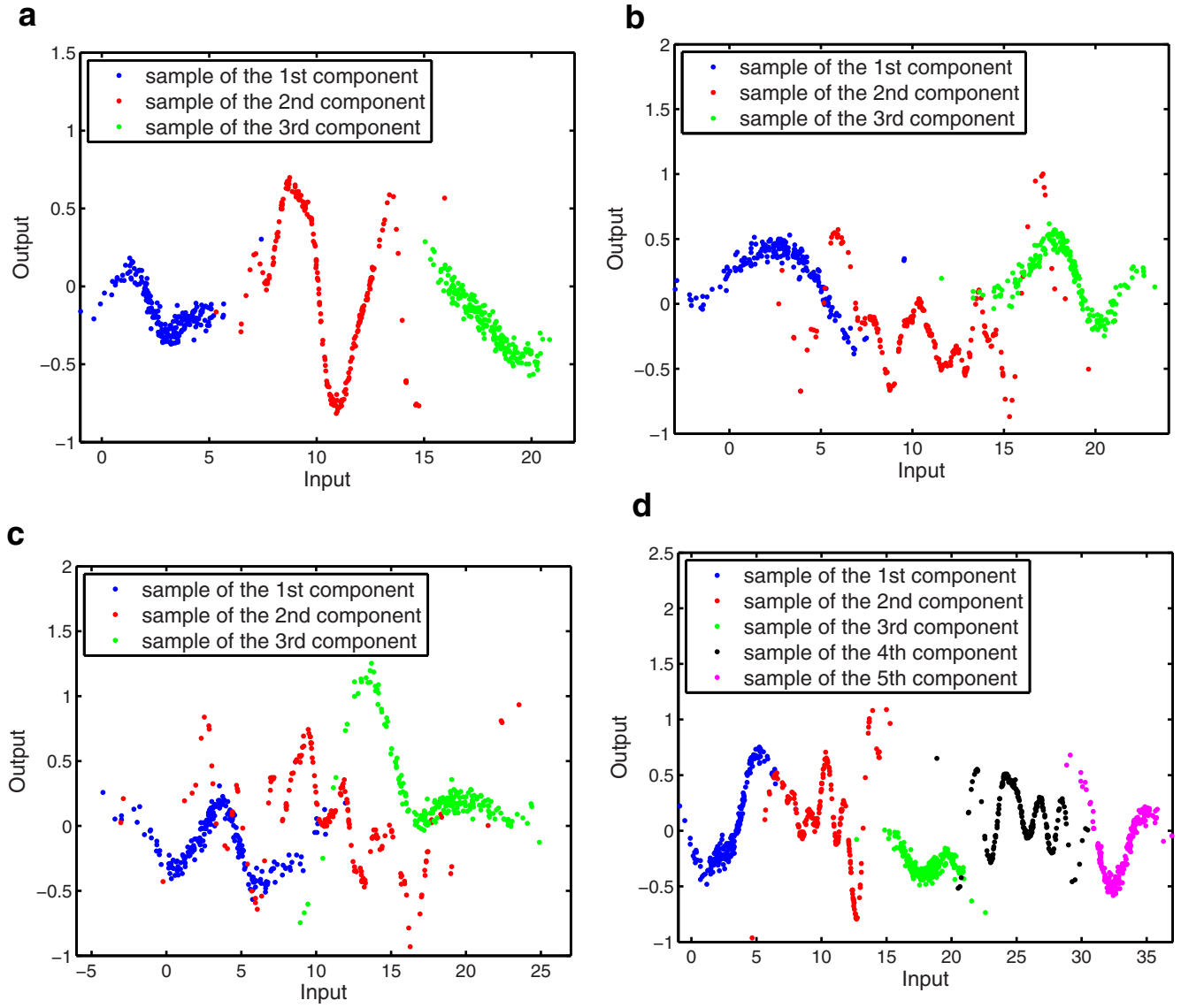


Fig. 1. (a–c), (d) are sketches of data sets $\mathcal{S}_1$,$\mathcal{S}_5$, $\mathcal{S}_9$ with six hundred samples and $\mathcal{S}_{13}$ with one thousand samples, respectively, described in Section 5.1.

where $N_k = \sum_{i=I_0+1}^{I_0+I} \sum_{n=1}^{N} z_{nki}$. But the analytical maximum of parameter $\boldsymbol{\theta}_k$ cannot be solved directly. To solve this problem, we utilize the gradient-ascent method to estimate $\boldsymbol{\theta}_k$ iteratively according to the following partial derivatives:

$$\frac{\partial}{\partial[\boldsymbol{\theta}_k]_j}\hat{\mathrm{Q}}(\Theta|\hat{\Theta}) = \frac{1}{I}\sum_{i=I_0+1}^{I_0+I}\left[-\frac{1}{2}\mathrm{tr}\left(\mathbf{C}_{ki}^{-1}\frac{\partial \mathbf{C}_{ki}}{\partial[\boldsymbol{\theta}_k]_j}\right) + \frac{1}{2}\mathbf{y}_{ki}^{T}\mathbf{C}_{ki}^{-1}\frac{\partial \mathbf{C}_{ki}}{\partial[\boldsymbol{\theta}_k]_j}\mathbf{C}_{ki}^{-1}\mathbf{y}_{ki}\right],$$

where $j = 1, 2, 3$.

4.2. Learning and prediction strategies

4.2.1. MCMC EM learning strategy

By using the $\hat{Q}$-function, we establish the MCMC EM algorithm for MGPs in Algorithm 1.

During each iteration of the MCMC EM algorithm, simulated samples $\mathbf{Z}_1, \ldots, \mathbf{Z}_{I_0+I}$ and the parameter set Θ are updated iteratively. Essentially, the hard-cut EM algorithm can be considered as a special MCMC EM algorithm with only one deterministic sample ($I_0 = 0, I = 1$). So, the complexity of the MCMC EM algorithm is higher than that of the hard-cut EM algorithm. For the conventional EM algorithm, $(Q_r - Q_{r-1})/|Q_{r-1}| < \varepsilon$ is adopted as the convergence criterion, where Q_r is the value of Q-function at the end of the rth iteration. As for the MCMC EM algorithm, $\hat{Q}_r$ may fluctuate during the iterative process because of randomly simulated samples. Thus, we adopt the relatively long-term convergence criterion as

$$\left[\left(\hat{Q}_r + \hat{Q}_{r-1}\right) - \left(\hat{Q}_{r-2} + \hat{Q}_{r-3}\right)\right] / \left|\hat{Q}_{r-2} + \hat{Q}_{r-3}\right| < \varepsilon.$$

It is found by experiments that $\varepsilon = 0.002$ is a good choice. On the other hand, we also adopt $r \le T$ to avoid slow convergence. In our current experiments, the algorithm always converges before the 20th iteration, so we set a reasonable number $T = 24$.

4.2.2. Overall MCMC prediction strategy

To improve the predictive accuracy, we implement an overall MCMC prediction strategy on the learned MGP model. The purpose of the MGP model is to predict the unknown output y_{N+1} at a new known input x_{N+1}. As a general prediction strategy, we divide all training samples into K components and predict the output of each

Algorithm 1 The MCMC EM algorithm for MGPs.

Input: $\mathcal{D}$, K.
Output: Θ, $\{\mathbf{Z}_i\}_{i=1}^{I_0+I}$, $\{\hat{Q}_r\}$.
1: Initialize the indicatory matrix $\mathbf{Z}$ by k-means clustering on the input data set $\{x_n\}_{n=1}^{N}$. Next, infer $\hat{\Theta}$ by the maximum likelihood estimation. Set the number of current iteration $r = 1$.
2: E-step. From $\hat{\Theta}$, generate a set of simulated samples $\mathbf{Z}_1, \ldots, \mathbf{Z}_{I_0+I}$ by the specialized Gibbs sampling in Appendix A. From simulated samples, $\hat{\mathrm{Q}}(\Theta|\hat{\Theta})$ is derived in Section 4.1.1.
3: M-step. Update Θ by maximizing$\hat{\mathrm{Q}}(\Theta|\hat{\Theta})$, as shown in Section 4.1.2.
4: Denote the value of $\hat{\mathrm{Q}}$-function at the end of the rth iteration as $\hat{Q}_r$ and the largest number of iterations as T.
if $\left[\left(\hat{Q}_r + \hat{Q}_{r-1}\right) - \left(\hat{Q}_{r-2} + \hat{Q}_{r-3}\right)\right] / \left|\hat{Q}_{r-2} + \hat{Q}_{r-3}\right| < \varepsilon$ or $r \geq T$ **then**
stop
else
$r = r + 1$ and return to the Step 2
end if

component independently at the input. Finally,we combine all the outputs together to produce the global predictive output. However, the classification of training samples is sometimes not reasonable and may lead to a certain deviation on prediction. To overcome this problem, we utilize the MCMC sampling to establish an overall MCMC prediction strategy.

The prediction of y_{N+1} with the indicatory matrix $\mathbf{Z}$ is given by

$$\hat{y}_{N+1}(\mathbf{Z}) = \sum_{k=1}^{K} \alpha_{N+1,k} \hat{y}_{N+1,k}(\mathbf{Z}), \tag{10}$$

where

$$\begin{aligned}\alpha_{N+1,k} &= \mathbb{E}\left(\mathbf{z}_{N+1} = \mathbf{e}_k | x_{N+1}, \widehat{\Theta}\right) = p\left(\mathbf{z}_{N+1} = \mathbf{e}_k | x_{N+1}, \widehat{\Theta}\right) \\ &= \hat{\pi}_k p\left(x_{N+1} | \hat{\mu}_k, \hat{\sigma}_k\right) / \sum_{j=1}^{K} \hat{\pi}_j p\left(x_{N+1} | \hat{\mu}_j, \hat{\sigma}_j\right),\end{aligned}$$

and $\hat{y}_{N+1,k}(\mathbf{Z})$ is the prediction of y_{N+1} when the sample (x_{N+1}, y_{N+1}) belongs to the kth component. Therefore, the overall predictive output is obtained by

$$\hat{y}^*_{N+1} = \sum_{\mathbf{Z}} P\left(\mathbf{Z} | \mathcal{D}, \widehat{\Theta}\right) \hat{y}_{N+1}(\mathbf{Z}). \tag{11}$$

It is noted that computation of this overall predictive output is still of exponential order. To solve this problem, we adopt the same MCMC sampling to generate I simulated samples of the indicatory matrix. Here, the number I is selected in a similar way, but it can be much larger for the prediction task because the computation complexity becomes much less than that of the training process in which a series of MCMC samplings must be made. As a result, we compute the overall MCMC predictive output by

$$\hat{y}_{N+1} = \frac{1}{I} \sum_{i=I_0+1}^{I_0+I} \sum_{k=1}^{K} \alpha_{N+1,k} \hat{y}_{N+1,k}(\mathbf{Z}_i), \tag{12}$$

where $\hat{y}_{N+1,k}(\mathbf{Z}_i) = \mathbf{c}_{N+1,i} \mathbf{C}_{ki}^{-1} \mathbf{y}_{ki}$, and $\mathbf{c}_{N+1,i} = \left[C\left(x_{N+1}, x_n; \hat{\boldsymbol{\theta}}_k\right) \middle| \mathbf{z}_{ni} = \mathbf{e}_k; n = 1, \ldots, N\right]$ is a row vector of $C\left(x_{N+1}, x_n; \hat{\boldsymbol{\theta}}_k\right)$.

5. Experimental results

Using synthetic and real-world data sets, we test the MCMC EM algorithm on classification and prediction by comparison with three state-of-the-art EM algorithms.

Table 1
The whole parameter sets for $\mathcal{S}_1$ and $\mathcal{S}_{13}$, respectively.

		π_k	μ_k	σ_k^2	$\boldsymbol{\theta}_k$
$\mathcal{S}_1$	$k=1$	1/3	3	1.8	[0.1,0.6,0.0025]
	$k=2$	1/3	10.5	4.05	[0.25,4,0.0005]
	$k=3$	1/3	18	1.8	[0.075,0.4,0.0025]
$\mathcal{S}_{13}$	$k=1$	0.2	3	4.5	[0.1,0.6,0.0025]
	$k=2$	0.2	10.5	10.125	[0.25,4,0.0005]
	$k=3$	0.2	18	4.5	[0.075,0.4,0.0025]
	$k=4$	0.2	25.5	10.125	[0.35,2,0.0005]
	$k=5$	0.2	33	4.5	[0.15,0.6,0.0025]

5.1. Synthetic data sets

As the structure of the MGP model is more complex and flexible than those of some mixture models, we select 21 typical synthetic data sets, denoted by $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{21}$, from MGP models for our experiments.

Our basic data set $\mathcal{S}_1$ contains 240 training samples and 660 test samples generated from three GPs, respectively, i.e., $K = 3$. Parameters of this data set are listed in Table 1, and the sketch of six hundred samples of $\mathcal{S}_1$ is already illustrated in Fig. 1(a). Three components are different with each other, so we predict them by three different Gaussian processes. The data sets $\mathcal{S}_2 - \mathcal{S}_{12}$ are based on $\mathcal{S}_1$, but change greatly in different ways. (1). $\mathcal{S}_2$ (a noisy data set): $\left[\boldsymbol{\theta}_1\right]_3 = \left[\boldsymbol{\theta}_3\right]_3 = 0.05$; $\left[\boldsymbol{\theta}_2\right]_3 = 0.01$. (2). $\mathcal{S}_3$ (an unbalanced data set): $\pi_1 = \pi_3 = 0.2$; $\pi_2 = 0.6$. (3). $\mathcal{S}_4$ (a small data set): there are 120 training samples and 330 test samples with the same parameter setting as $\mathcal{S}_1$. (4). $\mathcal{S}_5$ (a medium overlapping data set): $\sigma_1^2 = \sigma_3^2 = 4.5$; $\sigma_2^2 = 10.125$. (5). $\mathcal{S}_6$: this set comes from $\mathcal{S}_5$ in the same changing way as $\mathcal{S}_2$, denoted by $\mathcal{S}_5\&\mathcal{S}_2$. In the following, the notation $\mathcal{S}_j\&\mathcal{S}_l$ is of the similar meaning. (6). $\mathcal{S}_7$: $\mathcal{S}_5\&\mathcal{S}_3$. (7). $\mathcal{S}_8$: $\mathcal{S}_5\&\mathcal{S}_4$. (8). $\mathcal{S}_9$ (a heavily overlapping data set): $\sigma_1^2 = \sigma_3^2 = 9$; $\sigma_2^2 = 20.25$. (9). $\mathcal{S}_{10}$: $\mathcal{S}_9\&\mathcal{S}_2$. (10). $\mathcal{S}_{11}$: $\mathcal{S}_9\&\mathcal{S}_3$. (11). $\mathcal{S}_{12}$: $\mathcal{S}_9\&\mathcal{S}_4$. In summary, data sets $\mathcal{S}_1 - \mathcal{S}_4$ are of small overlap among the components, $\mathcal{S}_5 - \mathcal{S}_8$ are of medium overlap, and $\mathcal{S}_9 - \mathcal{S}_{12}$ are of strong overlap.

As the number of components is important for the MGP model, we generate nine other data sets $\mathcal{S}_{13} - \mathcal{S}_{21}$ with five or ten components, i.e., $K = 5$ or $K = 10$. In $\mathcal{S}_{13}$, there are 400 training samples and 1100 test samples. All parameters for $\mathcal{S}_{13}$ are listed in Table 1. Moreover, the sketches of samples of $\mathcal{S}_5$, $\mathcal{S}_9$ and $\mathcal{S}_{13}$ are illustrated in Fig. 1(b)–(d), respectively. $\mathcal{S}_{14} - \mathcal{S}_{17}$ are based on $\mathcal{S}_{13}$, but change greatly in different ways. (1). $\mathcal{S}_{14}$: $\left[\boldsymbol{\theta}_1\right]_3 = \left[\boldsymbol{\theta}_3\right]_3 = \left[\boldsymbol{\theta}_5\right]_3 = 0.05$; $\left[\boldsymbol{\theta}_2\right]_3 = \left[\boldsymbol{\theta}_4\right]_3 = 0.01$. (2). $\mathcal{S}_{15}$: $\pi_1 = \pi_3 = \pi_5 = 1/9$; $\pi_2 = \pi_4 = 3/9$. (3). $\mathcal{S}_{16}$: there are 200 training samples and 550 test samples with the same parameters as $\mathcal{S}_{13}$. (4). $\mathcal{S}_{17}$: $\mathcal{S}_{14}\&\mathcal{S}_{16}$. (5). $\mathcal{S}_{18}$: there are 48 training samples and 132 test samples with the same parameters as $\mathcal{S}_{13}$. (6). $\mathcal{S}_{19}$: $\mathcal{S}_{14}\&\mathcal{S}_{18}$. (7). We generate a data set $\mathcal{S}_{20}$ with 10 components by 2 steps: translate input of $\mathcal{S}_{17}$ from interval [0,36] to interval [36,72], and get the data set $\mathcal{S}_{17}^{(2)}$, then combine $\mathcal{S}_{17}^{(2)}$ with the data set $\mathcal{S}_{16}$. (8). We generate $\mathcal{S}_{21}$ by similar steps as $\mathcal{S}_{20}$, but adopt $\mathcal{S}_{18}$ and $\mathcal{S}_{19}$ instead of $\mathcal{S}_{16}$ and $\mathcal{S}_{17}$, respectively. Clearly, each of $\mathcal{S}_{17} - \mathcal{S}_{19}$ has a smaller number of samples, while each of $\mathcal{S}_{20} - \mathcal{S}_{21}$ has a larger number of components ($K = 10$).

For the MCMC EM algorithm, we achieve a higher accuracy on parameter estimation with a larger number I, but the time of MCMC procedure scales linearly with I. In fact, as I increases from 1 to 15, the prediction accuracy of the MCMC algorithm improves remarkably. As it increases from 15 to 100, the MCMC EM algorithm slightly outperforms on prediction. For example, on the data set $\mathcal{S}_8$, the average predicted RMSEs of the MCMC EM algorithm with $I = 5, 15, 100$ are 0.1859, 0.1856, 0.1855, respectively. Actually, we cannot adopt $I = 100$ because of the prohibitive time complexity. We set $I = 25$, which is reasonably larger than 15 to

Table 2
The average CARs of the MCMC EM algorithm and comparative algorithms over thirty trials on synthetic test data sets. In $\mathcal{S}_{20}$ and $\mathcal{S}_{21}$ column, the average F1 values are listed for three algorithms, respectively.

···%	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	$\mathcal{S}_7$	$\mathcal{S}_8$
LOOCV EM	98.71	98.61	95.97	98.56	93.96	94.00	82.06	93.13
Hard-cut EM	98.84	98.80	98.47	98.48	90.96	91.51	85.41	91.43
MCMC EM	**99.13**	**99.11**	**99.34**	**98.93**	**94.72**	**95.09**	**96.53**	**94.30**
···%	$\mathcal{S}_9$	$\mathcal{S}_{10}$	$\mathcal{S}_{11}$	$\mathcal{S}_{12}$	$\mathcal{S}_{13}$	$\mathcal{S}_{14}$	$\mathcal{S}_{15}$	$\mathcal{S}_{16}$
LOOCV EM	**91.83**	**92.12**	76.87	**90.19**	93.32	92.85	81.63	92.60
Hard-cut EM	86.13	84.99	76.90	85.96	89.81	89.71	85.66	89.85
MCMC EM	89.64	91.46	**90.80**	86.11	**94.93**	**94.39**	**94.80**	**93.65**
···%	$\mathcal{S}_{17}$	$\mathcal{S}_{18}$	$\mathcal{S}_{19}$	$\mathcal{S}_{20}$	$\mathcal{S}_{21}$			
LOOCV EM	92.84	**85.60**	**86.08**	82.57	–			
Hard-cut EM	90.49	84.31	85.25	80.22	**71.62**			
MCMC EM	**92.54**	77.93	80.57	**84.01**	69.79			

Table 3
The average predicted RMSEs of the MCMC EM algorithm and comparative algorithms over thirty trials on synthetic test data sets.

$\cdots \times 10^{-2}$	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$	$\mathcal{S}_6$	$\mathcal{S}_7$	$\mathcal{S}_8$
FNN	10.98	24.85	13.21	14.83	20.07	30.10	20.01	26.64
SVM	9.108	22.28	9.521	10.88	19.01	27.58	17.67	21.23
GP	12.05	23.85	9.863	15.10	18.89	28.83	19.25	22.52
Variational EM	56.21	39.43	89.98	64.90	46.40	42.07	70.08	46.34
LOOCV EM	15.84	25.79	17.87	18.26	20.65	29.58	23.10	23.33
Hard-cut EM	11.40	22.74	13.23	14.01	19.07	28.12	18.03	22.76
MCMC EM	**8.847**	**21.48**	**8.248**	**10.73**	**17.68**	**26.64**	**16.75**	**19.91**
$\cdots \times 10^{-2}$	$\mathcal{S}_9$	$\mathcal{S}_{10}$	$\mathcal{S}_{11}$	$\mathcal{S}_{12}$	$\mathcal{S}_{13}$	$\mathcal{S}_{14}$	$\mathcal{S}_{15}$	$\mathcal{S}_{16}$
FNN	26.36	35.14	29.96	30.49	23.77	30.71	25.52	25.74
SVM	24.57	32.42	26.54	27.54	22.65	28.76	22.45	22.77
GP	24.63	35.16	26.35	28.95	23.73	28.44	22.39	24.30
Variational EM	54.59	46.17	57.65	48.79	57.80	59.67	64.87	48.47
LOOCV EM	25.43	34.09	29.77	28.17	23.74	30.07	25.86	24.60
Hard-cut EM	25.16	32.69	27.03	28.32	23.53	29.28	23.27	23.51
MCMC EM	**23.16**	**31.56**	**24.58**	**26.02**	**21.24**	**27.78**	**21.09**	**21.17**
$\cdots \times 10^{-2}$	$\mathcal{S}_{17}$	$\mathcal{S}_{18}$	$\mathcal{S}_{19}$	$\mathcal{S}_{20}$	$\mathcal{S}_{21}$			
FNN	35.10	41.35	47.42	30.67	43.11			
SVM	30.80	32.58	40.45	28.04	36.74			
GP	32.27	34.94	41.90	30.98	39.81			
Variational EM	47.92.	44.26	45.37	46.70	44.51			
LOOCV EM	31.80	33.19	41.19	28.88	-			
Hard-cut EM	30.76	*32.50*	39.02	*27.97*	36.61			
MCMC EM	**29.67**	**31.09**	**37.98**	**26.28**	**35.14**			

avoid some special situations. During each Gibbs sampling of the MCMC EM algorithm, twenty five effective simulated samples are generated after the first ten initial simulated samples ($I_0 = 10$). Therefore, the total number of simulated samples during the iteration of MCMC EM algorithm is about 500, which is just about 1/40 of those of the general MCMC approach. Overall, the MCMC EM algorithm is more efficient than the MCMC approach.

There are some versions of variational EM algorithm, but we only compare the open-source code of the variational EM algorithm in [16] in our experiments. We set K as the true number of components, i.e., $K = 3$, $K = 5$ or $K = 10$, for the MCMC EM, hard-cut EM and LOOCV EM algorithms. As for the variational EM algorithm, K is selected adaptively on the data set with a Dirichlet process, which always leads to a wrong result. So, it is unfair to compare the Classification Accuracy Rates (CARs) of the variational EM algorithm to those of other algorithms. Thus, we just list the CARs of the MCMC EM, hard-cut EM and LOOCV EM algorithms in Table 2. The MCMC EM algorithm generally obtains a better result of sample classification (or component identification) than the hard-cut EM and LOOCV EM algorithms. However, on some data sets, the average CARs of the LOOCV EM algorithm are better than those of the MCMC EM algorithm. Clearly, these data sets are always medium and strong overlapping data sets.

The Root Mean Square Error (RMSE) between y_n and $\hat{y}_n$ for $n = 1, \ldots, N$, is calculated by $\sqrt{\sum_{n=1}^{N} \left(y_n - \hat{y}_n\right)^2 / N}$ to measure the prediction error. In Table 3, we compare four EM algorithms for MGPs with three competitive regression algorithms, i.e., Support Vector Machine (SVM) [27], Feed-forward Neural Network (FNN) [28] and single Gaussian process (GP). The SVM and FNN are popular regression-based methods on this type of data sets. In experiments, we utilize the SVM and FNN toolboxes in MATLAB (MathWorks, Natick, MA). Moreover, we adopt the GPML toolbox [25] for learning the GP model.

In Table 3, the average predicted RMSEs of the MCMC EM algorithm are always the smallest. The advantage of the MCMC EM algorithm is different on these synthetic data sets. In fact, the advantage of the MCMC EM algorithm is remarkable on $\mathcal{S}_3$, $\mathcal{S}_8$ and $\mathcal{S}_{11}$, while it is only a bit better on $\mathcal{S}_4$. In Tables 2 and 3, there is no strong relation between the CAR and the prediction RMSE.

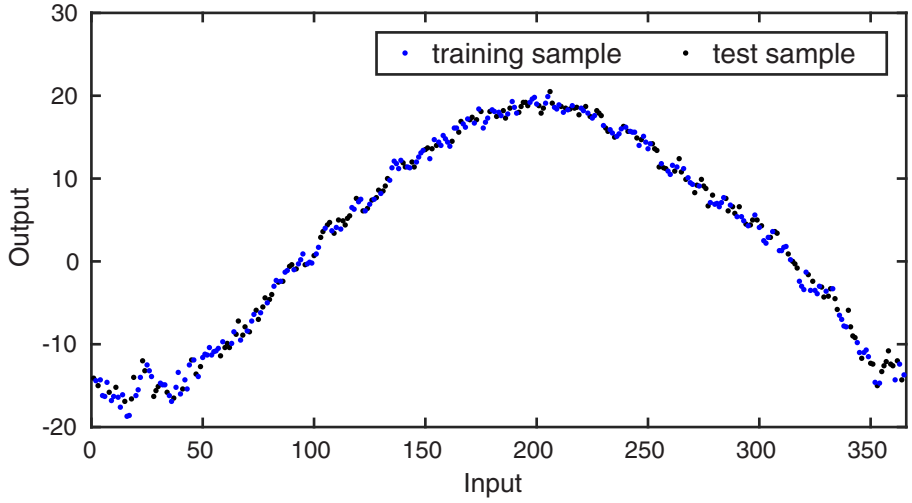


Fig. 2. The training and test sample points of the weather data set $\mathcal{R}_2$.

Table 4
The average predicted RMSEs, as well as the average time consumptions, of the MCMC EM algorithm and comparative algorithms over thirty trials on real-world test data sets. For $\mathcal{R}_2$, "average ± standard deviation" of RMSEs are listed. LM and SMO are short for Levenberg–Marquardt and Sequential Minimal Optimization, respectively.

Model	Algorithm	$\mathcal{R}_1$	$\mathcal{R}_2$		$\mathcal{R}_3$	$\mathcal{R}_4$	$\mathcal{R}_5$
		RMSE	RMSE	Time	RMSE	RMSE	RMSE
FNN	LM	2.4666	1.0408 ± 0.3570	5.855	0.8886	1.1583	82, 445
SVM	SMO	2.4201	1.0137 ± 0.0227	32.94	0.8650	0.8490	84, 710
GP	Polak-Ribiere	2.5478	0.9783 ± 0.0000	0.937	0.8860	1.1485	83, 689
MGP	Variational EM	27.456	2.1491 ± 3.0687	6.236	2.0823	3.6265	165, 496
	LOOCV EM	14.215	2.0736 ± 0.2371	104.4	3.8705	2.3241	153, 370
	Hard-cut EM	3.2781	0.9612 ± 0.0214	104.1	0.8504	1.0318	79,083
	MCMC EM	**1.9431**	$\mathbf{0.9444 \pm 0.0105}$	113.4	**0.8357**	**0.8354**	**78,238**

For example, on $\mathcal{S}_{12}$, although the LOOCV EM algorithm has the best performance on classification, its prediction results are not accurate. When classification and prediction are considered together, the hard-cut EM algorithm is better than the LOOCV EM algorithm. On many data sets, the SVM is not only better than the FNN and GP model, but also slightly better than the hard-cut EM algorithm. On some data sets, the hard-cut EM algorithm and GP model are much better than the SVM. The MCMC EM algorithm with the general prediction strategy is poorer than the MCMC EM algorithm with the overall MCMC prediction strategy. Moreover, the MCMC EM algorithm with the overall MCMC prediction strategy is more powerful than the hard-cut EM algorithm with the overall MCMC prediction strategy. Consequently, both the MCMC EM algorithm and the overall MCMC prediction strategy are important for prediction.

5.2. Real-world data sets

We test and compare the MCMC EM algorithm on 5 real-world data sets $\mathcal{R}_1 - \mathcal{R}_5$, respectively. Samples in $\mathcal{R}_1$ consist of the monthly Purchasing Managers Index (%) from the National Bureau of Statistics of the People's Republic of China. In $\mathcal{R}_1$, there are 80 training samples and 80 test samples. The weather data sets $\mathcal{R}_2 - \mathcal{R}_4$ consist of daily temperature (°C) averages over the year from 1961 to 1994 in three weather stations of Canada (i.e., the Arvida, Bagottvi. and Calgary stations). In each weather data set, there are 200 training samples and 165 test samples. The Ethernet data set $\mathcal{R}_5$ consists of the Ethernet speeds along the time, with 600 training samples and 1200 test samples.

For those real-world data sets, there is no real component number and no real indicator, so it is crucial to determine an appropriate number of components for a data set. As being well-known, this is a specific model selection problem related with the finite mixture model and the EM algorithm. In fact, the model selection of MGPs is generally a very challenging problem in practical applications. Certainly, we can implement certain automated model selection algorithms (e.g., [29–32]) directly on the sample points of a given data set to get a proper value of K, but the results may be poor since they do not consider the temporal relation between the sample points. Zhao et al. [33] proposed a synchronously balancing criterion (SBC) for the model selection of MGPs along the input region, but the penalty coefficient in the SBC is unknown and should be carefully determined for a data set. As these methods cannot be easily used for the model selection of MGPs, we simply implement 30-fold cross validation procedure [34] to make model selection for the MCMC EM, hard-cut EM, LOOCV EM and FNN algorithms.

Next, we estimate the MGP model on training samples and predict on test samples. In Fig. 2, we illustrate the training and test sample points of $\mathcal{R}_2$. The stochastic process on the input interval [1,50] is different from those on the other intervals, so it is a multimodal data set. In Table 4, the first two columns contain models and their related algorithms. Four EM algorithms in the second column are used for MGPs. The indices in Table 4 are the predicted RMSEs on the test data sets. It is clear that the MCMC EM algorithm shows a good prediction result. In addition, the hard-cut EM algorithm also performs well on these real-world data sets, and the prediction performances of SVM, FNN and GP models are almost on the same level.

5.3. The limitations of the MCMC EM algorithm

Computational time complexity (in seconds) including the model selection procedures on $\mathcal{R}_2$ are listed in Table 4 when custom MATLAB code is run on a server [Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.4GHz, 28 Core, 128GB RAM]. The MCMC EM algorithm is quite effective on prediction. As a trade-off, the MCMC EM algorithm is much slower than the FNN, SVM, GP model and variational EM algorithm. In comparison with the LOOCV EM and hard-cut EM algorithms, the MCMC EM algorithm is just slightly slower, but get the remarkably better results. Fortunately, the time complexity of the MCMC EM algorithm may be overcome with the development of modern hardware, especially the prediction accuracy is more concerned in some practical applications.

6. Conclusion and discussion

We establish the MCMC EM algorithm for MGPs in which the Q-function is approximated by MC simulated samples. Based on these simulated samples, the computation of Q-function becomes feasible. In fact, the MCMC EM algorithm combines the merits of the MCMC approximation and EM algorithm. Experiments on synthetic and real-world data sets demonstrate that our MCMC EM algorithm is more effective than the state-of-the-art EM algorithms on classification and prediction problems.

In the future, we plan to improve the computational complexity by combining the MCMC EM algorithm and the Mixture of Sparse GP (MSGP) model [21,35], since the MSGP model can also learn this type of data sets and is generally faster than the MGP model. Nevertheless, the predicted accuracy of the MSGP model is usually lower than that of the MGP model. Therefore, it is crucial to consider the trade-off between the computational complexity and prediction accuracy. On the other hand, we also plan to improve the computation for some large data set by speeding up the model selection procedure [22,33].

Acknowledgements

This work is supported by the National Science Foundation of China for Grant no. 61171138.

Appendix A. The Gibbs sampling for the $\hat{Q}$-function

There are many MCMC methods in machine learning and data analyses, but most of them are complex to sample the indicator matrix $\mathbf{Z}$. Fortunately, the Gibbs sampling is a simple yet effective way to generate simulated samples $\mathbf{Z}_1, \ldots, \mathbf{Z}_{l_0+I}$ for the $\hat{Q}$-function. In the beginning, we set an initial state $\mathbf{Z}_0$ of the Markov chain according to the MAP principle, which was already adopted in the E-step of the hard-cut EM algorithm. That is, $\mathbf{z}_{n0} = \mathbf{e}_{\hat{k}}$ if and only if

$$\hat{k} = \max_k p(\mathbf{z}_n = \mathbf{e}_k | x_n, y_n, \Theta) = \max_k \pi_k p(x_n|\mu_k, \sigma_k,) p\big(y_n|x_n, \boldsymbol{\theta}_k\big),$$

where $y_n|x_n, \boldsymbol{\theta}_k \sim \mathcal{N}\big(0, \big[\boldsymbol{\theta}_k\big]_1 + \big[\boldsymbol{\theta}_k\big]_3\big)$.

On the basis of $\mathbf{Z}_{i-1} = \big[\mathbf{z}_{1,i-1}, \ldots, \mathbf{z}_{N,i-1}\big]$, we can generate the ith simulated sample $\mathbf{Z}_i = [\mathbf{z}_{1i}, \ldots, \mathbf{z}_{Ni}]$ via N steps in Algorithm 2. Indicatory matrices $\mathbf{Z}_1, \ldots, \mathbf{Z}_{l_0+I}$ are generated sequentially from $\mathbf{Z}_0$.

At the nth step, we generate a component indicator vector $\mathbf{z}_{ni}$. The total number of steps for sampling all $l_0 + I$ simulated samples is $(l_0 + I) \times N$, so the computational complexity of the Gibbs sampling procedure is high. As for the posterior probability of $\mathbf{z}_n$, it can be computed by

$$P\big(\mathbf{z}_n = \mathbf{e}_k | \mathbf{Z}^{(-n)}, \mathcal{D}, \Theta\big) = \frac{p\big(\mathbf{z}_n = \mathbf{e}_k; \mathbf{Z}^{(-n)}, \mathcal{D}|\Theta\big)}{\sum_{j=1}^{K} p\big(\mathbf{z}_n = \mathbf{e}_j; \mathbf{Z}^{(-n)}, \mathcal{D}|\Theta\big)},$$

Algorithm 2 The Gibbs sampling for simulated sample $\mathbf{Z}_i$.

Input: $\mathcal{D}$, Θ; $\mathbf{Z}_{i-1} = \big[\mathbf{z}_{1,i-1}, \ldots, \mathbf{z}_{N,i-1}\big]$.
Output: $\mathbf{Z}_i = [\mathbf{z}_{1i}, \ldots, \mathbf{z}_{Ni}]$.
1: Sample $\mathbf{z}_{1i}$ from $P\big(\mathbf{z}_1|\mathbf{z}_{2,i-1}, \ldots, \mathbf{z}_{N,i-1}, \mathcal{D}, \Theta\big)$.
. . . . . .
n: Sample $\mathbf{z}_{ni}$ from $P\big(\mathbf{z}_n|\mathbf{z}_{1i}, \ldots, \mathbf{z}_{n-1,i}, \mathbf{z}_{n+1,i-1}, \ldots, \mathbf{z}_{N,i-1}, \mathcal{D}, \Theta\big)$.
. . . . . .
N: Sample $\mathbf{z}_{Ni}$ from $P\big(\mathbf{z}_N|\mathbf{z}_{1i}, \ldots, \mathbf{z}_{N-1,i}, \mathcal{D}, \Theta\big)$.

where the index i is ignored for the specific sampling, $\mathbf{Z}^{(-n)} = [\mathbf{z}_1, \ldots, \mathbf{z}_{n-1}, \mathbf{z}_{n+1}, \ldots, \mathbf{z}_N]$ for $n = 2, \ldots, N-1$, $\mathbf{Z}^{(-1)} = [\mathbf{z}_2, \ldots, \mathbf{z}_N]$, $\mathbf{Z}^{(-N)} = [\mathbf{z}_1, \ldots, \mathbf{z}_{N-1}]$ and

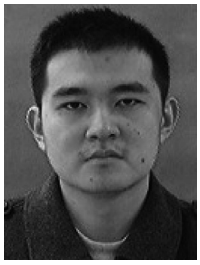
$$p\big(\mathbf{z}_n = \mathbf{e}_k; \mathbf{Z}^{(-n)}, \mathcal{D}|\Theta\big) \propto \pi_k p(x_n|\mu_k, \sigma_k) p\big(\mathbf{y}_{+n,k}|\mathbf{x}_{+n,k}, \boldsymbol{\theta}_k\big) / p\big(\mathbf{y}_{-n,k}|\mathbf{x}_{-n,k}, \boldsymbol{\theta}_k\big).$$

In addition, $\mathbf{y}_{+n,k} = \big[y_j | z_{jk} = 1 \text{ or } j = n\big]$ and $\mathbf{y}_{-n,k} = \big[y_j | z_{jk} = 1 \text{ and } j \neq n\big]$ are two complementary vectors for y_j, and $\mathbf{x}_{+n,k}$ and $\mathbf{x}_{-n,k}$ are defined in the same way as $\mathbf{y}_{+n,k}$ and $\mathbf{y}_{-n,k}$.

References

[1] C.E. Rasmussen, Evaluation of Gaussian Processes and Other Methods for Non–Linear Regression, Department of Computer Science, University of Toronto, 1996 Ph.D. thesis.
[2] C. E. Rasmussen, C. K. I. Williams, Gaussian Process for Machine Learning, MIT Press, Cambridge.
[3] V. Tresp, Mixtures of Gaussian processes, Adv. Neural Inf. Process. Syst. 13 (2000) 654–660.
[4] L. Xu, M.I. Jordan, G.E. Hinton, An alternative model for mixtures of experts, Adv. Neural Inf. Process. Syst. 7 (1995) 633–640.
[5] Y. Yang, J. Ma, Asymptotic convergence properties of the EM algorithm for mixture of experts, Neural Comput. 23 (8) (2011) 2140–2168.
[6] C. Andrieu, N.D. Freitas, A. Doucet, M.I. Jordan, An introduction to MCMC for machine learning, Mach. Learn. 50 (1) (2003) 5–43.
[7] J. Ma, L. Xu, M.I. Jordan, Asymptotic convergence rate of the EM algorithm for Gaussian mixtures, Neural Comput. 12 (12) (2000) 2881–2907.
[8] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. Ser. B Methodol. 39 (1977) 1–38.
[9] D. Wu, Z. Chen, J. Ma, An MCMC based EM algorithm for mixtures of Gaussian processes, Adv. Neural Netw. ISNN 9377 (2015) 327–334. LNCS
[10] E. Meeds, S. Osindero, An alternative infinite mixture of Gaussian process experts, Adv. Neural Inf. Process. Syst. 18 (2006) 883–896.
[11] C. Yuan, C. Neubauer, Variational mixture of Gaussian process experts, Adv. Neural Inf. Process. Syst. 21 (2008) 1897–1904.
[12] C.E. Rasmussen, Z. Ghahramani, Infinite mixture of Gaussian process experts, Adv. Neural Inf. Process. Syst. 2 (2002) 881–888.
[13] A. Tayal, P. Poupart, Y. Li, Hierarchical double Dirichlet process mixture of Gaussian processes, Assoc. Adv. Artif. Intell. (2012) 1126–1133.
[14] S. Sun, Infinite mixtures of multivariate Gaussian processes, in: Proceedings of the International Conference on Machine Learning and Cybernetics (ICMLC), 2013, pp. 1011–1016.
[15] P. Chen, N. Zabaras, I. Bilionis, Uncertainty propagation using infinite mixture of Gaussian processes and variational Bayesian inference, J. Comput. Phys. 284 (2015) 291–333.
[16] S. Sun, X. Xu, Variational inference for infinite mixtures of Gaussian processes with applications to traffic flow prediction, IEEE Trans. Intell. Transp. Syst. 12 (2) (2011) 466–475.
[17] T. Nguyen, E. Bonilla, Fast allocation of Gaussian process experts, in: Proceedings of the Thirty-First International Conference on Machine Learning (ICML), 2014, pp. 145–153.
[18] Y. Yang, J. Ma, An efficient EM approach to parameter learning of the mixture of Gaussian processes, Adv. Neural Netw. ISNN 6676 (2011) 165–174. LNCS
[19] G. Celeux, G. Govaert, A classification EM algorithm for clustering and two stochastic versions, Comput. Stat. Data Anal. 14 (3) (1992) 315–332.
[20] Z. Chen, J. Ma, Y. Zhou, A precise hard-cut EM algorithm for mixtures of Gaussian processes, in: Proceedings of the Tenth International Conference on Intelligent Computing (ICIC), 8589, 2014, pp. 68–75. LNCS
[21] Z. Chen, J. Ma, The hard-cut EM algorithm for mixture of sparse Gaussian processes, in: Proceedings of the Eleventh International Conference on Intelligent Computing (ICIC), 9227, 2015, pp. 13–24. LNCS
[22] L. Zhao, J. Ma, A dynamic model selection algorithm for mixtures of Gaussian processes, in: Proceedings of the IEEE Thirteenth International Conference on Signal Processing (ICSP), 2016, pp. 1095–1099, doi:10.1109/ICSP.2016.7877998.

[23] G.C.G. Wei, M.A. Tanner, A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms, J. Am. Stat. Assoc. 85 (1990) 699–704.
[24] D. Wu, J. Ma, A two-layer mixture model of Gaussian process functional regressions and its MCMC EM algorithm, IEEE Trans. Neural Netw. Learn. Syst. 29 (10) (2018) 4894–4904.
[25] C.E. Rasmussen, H. Nickisch, Gaussian processes for machine learning (GPML) toolbox, J. Mach. Learn. Res. 11 (2010) 3011–3015.
[26] S. Sundararajan, S.S. Keerthi, Predictive approaches for choosing hyperparameters in Gaussian processes, Neural Comput. 13 (5) (2001) 1103–1118.
[27] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 27.
[28] D. Svozil, V. Kvasnicka, J. Pospichal, Introduction to multi-layer feed-forward neural networks, Chemom. Intell. Lab. Syst. 39 (1) (1997) 43–62.
[29] Y. Cheung, Maximum weighted likelihood via rival penalized EM for density mixture clustering with automatic model selection, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 750–761.
[30] J. Ma, T. Wang, A cost-function approach to rival penalized competitive learning (RPCL), IEEE Trans. Syst. Man Cybern. Part B 36 (4) (2006) 722–737.
[31] J. Ma, J. Liu, The BYY annealing learning algorithm for Gaussian mixture with automated model selection, Pattern Recognit. 40 (7) (2007) 2029–2037.
[32] H. Jia, Y.M. Cheung, J. Liu, Cooperative and penalized competitive learning with application to kernel-based clustering, Pattern Recognit. 47 (2014) 3060–3069.
[33] L. Zhao, Z. Chen, J. Ma, An effective model selection criterion for mixtures of Gaussian processes, Adv. Neural Netw. ISNN 9377 (2015) 345–354. LNCS
[34] S. Arlot, A. Celisse, A survey of cross-validation procedures for model selection, Stat. Surv. 4 (2010) 40–79.
[35] J. Quinonero-Candela, C.E. Rasmussen, A unifying view of sparse approximate Gaussian process regression, J. Mach. Learn. Res. 6 (2005) 1939–1959.



Di Wu received the B.S. degree in mathematics from Shaanxi Normal University, Xi'an, China, in 2012, and he is currently pursuing the Ph.D. degree in applied mathematics with the School of Mathematical Sciences, Peking University, Beijing, China. His current research interests include machine learning, data mining and neural networks.

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