SCIENCE CHINA
Information Sciences

• **Supplementary File** •

A Variational Hardcut EM Algorithm for the Mixtures of Gaussian Processes

Tao Li & Jinwen Ma*

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

Appendix A Detailed Derivation of the Variational Hardcut EM (VHEM) Algorithm

Appendix A.1 Derivation of Equation (3)

Recall that we approximate $q(z_j; \boldsymbol{\lambda}_j)$ via a deterministic allocation

$$\tilde{q}(z_j; \boldsymbol{\lambda}_j) = \mathbb{I}(z_j = \operatorname*{arg\,max}_{k=1,2,\cdots,K} \lambda_{j,k}).$$

Under $\tilde{q}(\mathbf{z}_{-i}; \boldsymbol{\Lambda})$, the latent variables $\mathbf{z}_{-i}$ are deterministic, and we write as $\tilde{\mathbf{z}}_{-i}$. We can approximate the intractable expectation by

$$\mathbb{E}_{q(\mathbf{z}_{-i};\boldsymbol{\Lambda})}[\log p(\mathbf{X}, \mathbf{y}, \mathbf{z}_{-i}, z_i = k; \boldsymbol{\Theta})] \approx \mathbb{E}_{\tilde{q}(\mathbf{z}_{-i};\boldsymbol{\Lambda})}[\log p(\mathbf{X}, \mathbf{y}, \mathbf{z}_{-i}, z_i = k; \boldsymbol{\Theta})] = \log p(\mathbf{X}, \mathbf{y}, \tilde{\mathbf{z}}_{-i}, z_i = k; \boldsymbol{\Theta})\,.$$

Substituting this approximation into eq (2), and utilizing eq (1), we immediately get

$$\begin{aligned} \lambda_{i,k} &\propto \exp\left(\log p(\mathbf{X}, \mathbf{y}, \tilde{\mathbf{z}}_{-i}, z_i = k; \boldsymbol{\Theta})\right) \\ &= \pi_k p(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \prod_{l=1}^{K} p(\mathbf{y}_l(\tilde{\mathbf{z}}_{-i} \cup \{z_i = k\}) | \mathbf{X}_l(\tilde{\mathbf{z}}_{-i} \cup \{z_i = k\}); \boldsymbol{\theta}_l). \end{aligned} \tag{A1}$$

The right-hand side of the above equation is computable, but involves evaluating the likelihood of K Gaussian processes, which is still inefficient in practice. In the following, we show that we only need to calculate the likelihood of 2 Gaussian processes, which derives the eq (3). First, let

$$\mathbf{X}_{-i,k}(\mathbf{z}) = \{\mathbf{x}_j | z_j = k, j = 1, \cdots, N \texttt{ and } j \neq i\} \quad , \quad \mathbf{y}_{-i,k}(\mathbf{z}) = \{y_j | z_j = k, k = 1, \cdots, N \texttt{ and } j \neq i\},$$

then we have

$$\begin{aligned} &\prod_{l=1}^{K} p(\mathbf{y}_l(\{\tilde{\mathbf{z}}_{-i} \cup \{z_i = k\}) | \mathbf{X}_l(\tilde{\mathbf{z}}_{-i} \cup \{z_i = k\}\}); \boldsymbol{\theta}_l) \\ =& p(\mathbf{y}_k(\tilde{\mathbf{z}}_{-i} \cup \{z_i = k\}) | \mathbf{X}_k(\tilde{\mathbf{z}}_{-i} \cup \{z_i = k\}); \boldsymbol{\theta}_k) \prod_{l \neq k} p(\mathbf{y}_{-i,l}(\tilde{\mathbf{z}}_{-i}) | \mathbf{X}_{-i,l}(\tilde{\mathbf{z}}_{-i}); \boldsymbol{\theta}_l) \\ =& \frac{p(\mathbf{y}_k(\tilde{\mathbf{z}}_{-i} \cup \{z_i = k\}) | \mathbf{X}_k(\tilde{\mathbf{z}}_{-i} \cup \{z_i = k\}); \boldsymbol{\theta}_k)}{p(\mathbf{y}_{-i,k}(\tilde{\mathbf{z}}_{-i}) | \mathbf{X}_{-i,k}(\tilde{\mathbf{z}}_{-i}); \boldsymbol{\theta}_k)} \prod_{l=1}^{K} p(\mathbf{y}_{-i,l}(\tilde{\mathbf{z}}_{-i}) | \mathbf{X}_{-i,l}(\tilde{\mathbf{z}}_{-i}); \boldsymbol{\theta}_l). \end{aligned}$$

The last product of K Gaussian process likelihoods are common for all $\boldsymbol{\lambda}_{i,k}$, thus eq. (A1) can be modified as

$$\lambda_{i,k} \propto \pi_k p(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \frac{p(\mathbf{y}_k(\tilde{\mathbf{z}}_{-i} \cup \{z_i = k\}) | \mathbf{X}_k(\tilde{\mathbf{z}}_{-i} \cup \{z_i = k\}); \boldsymbol{\theta}_k)}{p(\mathbf{y}_{-i,k}(\tilde{\mathbf{z}}_{-i}) | \mathbf{X}_{-i,k}(\tilde{\mathbf{z}}_{-i}); \boldsymbol{\theta}_k)}\,,$$

which is exactly eq (3).

Appendix A.2 VHEM Algorithm Design

Equation (4) establishes the iteration formulae in the variational E-step. Once the iteration converges, we obtain an approximate posterior $\tilde{q}(\mathbf{z}) = \mathbb{I}(\mathbf{z} = \tilde{\mathbf{z}})$ and we can calculate the approximate Q-function $\tilde{\mathcal{Q}}(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{\text{old}}) = \mathbb{E}_{\tilde{q}(\mathbf{z})}[\mathcal{L}(\boldsymbol{\Theta}, \mathbf{z})] = \mathcal{L}(\boldsymbol{\Theta}, \tilde{\mathbf{z}})$. From eq (1), we can see the optimal estimations of $\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ are similar as in the Gaussian mixture model. The hyper-parameters $\{\boldsymbol{\theta}_k\}_{k=1}^K$ of Gaussian processes can be learned via the gradient descent algorithm for each Gaussian process component separately. Once we find the latent variables $\mathbf{z}$ remains invariant in two consecutive iterations, then the iteration process has converged and we terminate the EM iteration. In case that the algorithm fails to converge or spends too many steps to converge, we set the maximum number of iterations to be 20. However, in practice we find the VHEM algorithm converges in several EM iterations. The entire algorithm is summarized in algorithm A1.

* Corresponding author (email: jwma@math.pku.edu.cn)

Algorithm A1 The variational hard-cut EM algorithm for MGPs

Input: dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, the number of components K.
Parameters: mixing proportions $\{\pi_k\}_{i=1}^K$, Gaussian mixture model parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$, Gaussian process parameters $\{\boldsymbol{\theta}_k\}_{k=1}^K$.
Latent variables: latent variables $\{z_i\}_{i=1}^N$.

Initialize $\{z_i\}_{i=1}^N$ via the k-means algorithm.
while not converged **do**
  %% M-step
  for $k = 1, 2, \cdots, K$ **do**
    Update the mixture parameters of k-th Gaussian mixture component:

$$\mathcal{I}_k = \{i|z_i = k\}, \pi_k = \frac{|\mathcal{I}_k|}{N}, \boldsymbol{\mu}_k = \frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}\mathbf{x}_i, \boldsymbol{\Sigma}_k = \frac{1}{|\mathcal{I}_k|}\sum_{i\in\mathcal{I}_k}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\intercal.$$

    Update $\boldsymbol{\theta}_k, \sigma_k$ using gradient ascent in each Gaussian process components.
  end for
  %% E-step
  while not converged **do**
    for $i = 1, 2, \cdots, N$ **do**

$$z_i = \arg\max_{k=1,2,\cdots,K} \pi_k p(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\frac{p(\mathbf{y}_k(\tilde{\mathbf{z}}_{-i}\cup\{\tilde{z}_i = k\})|\mathbf{X}_k(\tilde{\mathbf{z}}_{-i}\cup\{\tilde{z}_i = k\}); \boldsymbol{\theta}_k)}{p(\mathbf{y}_{-i,k}(\tilde{\mathbf{z}}_{-i})|\mathbf{X}_{-i,k}(\tilde{\mathbf{z}}_{-i}); \boldsymbol{\theta}_k)}.$$

    end for
  end while
end while

Appendix B Further Comparisons with the Other Algorithms on the Learning of MGPs

VHEM v.s. MCMC EM. Despite the similarity between the VHEM algorithm and the MCMC EM algorithm, their derivations are very different. The MCMC-EM algorithm is time-consuming in practice since we have to perform sampling in each step, and it usually takes a long time for the Markov chain to attain the desired stationary distribution. Besides, it is difficult to diagnose whether the Markov chain has converged. On the contrary, the VHEM algorithm is faster since we avoid the sampling step, and we can conclude the fixed-point iteration has converged as long as two consecutive iterations lead to the same $\tilde{\mathbf{z}}$.

VHEM v.s. hardcut EM. The relationship between the VHEM algorithm and the hardcut EM algorithm has been discussed in the letter. Here, we present the detailed derivation of equation (5). We ignore the dependency on $\tilde{\mathbf{z}}, i, k$ for now and write

$$\begin{aligned}
&\mathbf{y}_- = \mathbf{y}_{-i,k}(\tilde{\mathbf{z}}_{-i}) &&,\mathbf{X}_- = \mathbf{X}_{-i,k}(\tilde{\mathbf{z}}_{-i}) &&,t = |\{j|\tilde{z}_j = k \text{ and } j \neq i\}| &&,\mathbf{C}_- = c(\mathbf{X}_-, \mathbf{X}_-; \boldsymbol{\theta}_k),\\
&\mathbf{y}_+ = [\mathbf{y}_-; y_i] &&,\mathbf{X}_+ = [\mathbf{X}_-; \mathbf{x}_i^\intercal] &&,\mathbf{c} = c(\mathbf{X}_-, \mathbf{x}_i; \boldsymbol{\theta}_k) &&,\mathbf{C}_+ = \begin{pmatrix}\mathbf{C}_- & \mathbf{c}\\ \mathbf{c}^\intercal & \boldsymbol{\theta}_{k,0}^2 + \sigma_k^2\end{pmatrix}.
\end{aligned}$$

Then

$$\begin{aligned}
&\log\frac{p(\mathbf{y}_k(\tilde{\mathbf{z}}_{-i}\cup\{\tilde{z}_i = k\})|\mathbf{X}_k(\tilde{\mathbf{z}}_{-i}\cup\{\tilde{z}_i = k\}); \boldsymbol{\theta}_k)}{p(\mathbf{y}_{-i,k}(\tilde{\mathbf{z}}_{-i})|\mathbf{X}_{-i,k}(\tilde{\mathbf{z}}_{-i}); \boldsymbol{\theta}_k)} = \log\frac{\mathcal{N}(\mathbf{y}_+|\mathbf{0}, \mathbf{C}_+)}{\mathcal{N}(\mathbf{y}_-|\mathbf{0}, \mathbf{C}_-)}\\
&= -\frac{t+1}{2}\log(2\pi) - \frac{1}{2}\log\det\mathbf{C}_+ - \frac{1}{2}\mathbf{y}_+^\intercal\mathbf{C}_+^{-1}\mathbf{y}_+ + \frac{t}{2}\log(2\pi) + \frac{1}{2}\log\det\mathbf{C}_- + \frac{1}{2}\mathbf{y}_-\mathbf{C}_-^{-1}\mathbf{y}_-.
\end{aligned}$$

Note that

$$\begin{aligned}
\log\det\mathbf{C}_+ &= \log\det\begin{pmatrix}\mathbf{C}_- & \mathbf{c}\\ \mathbf{c}^\intercal & \boldsymbol{\theta}_{k,0}^2 + \sigma_k^2\end{pmatrix} = \log\det\mathbf{C}_- + \log(\boldsymbol{\theta}_{k,0}^2 + \sigma_k^2 - \mathbf{c}^\intercal\mathbf{C}_-^{-1}\mathbf{c}),\\
\mathbf{C}_+^{-1} &= \begin{pmatrix}\mathbf{C}_- & \mathbf{c}\\ \mathbf{c}^\intercal & \boldsymbol{\theta}_{k,0}^2 + \sigma_k^2\end{pmatrix}^{-1} = \begin{pmatrix}\mathbf{C}_-^{-1} + \frac{1}{\boldsymbol{\theta}_{k,0}^2+\sigma_k^2-\mathbf{c}^\intercal\mathbf{C}_-^{-1}\mathbf{c}}\mathbf{C}_-^{-1}\mathbf{c}\mathbf{c}^\intercal\mathbf{C}_- & -\frac{1}{\boldsymbol{\theta}_{k,0}^2+\sigma_k^2-\mathbf{c}^\intercal\mathbf{C}_-^{-1}\mathbf{c}}\mathbf{C}_-^{-1}\mathbf{c}\\ -\frac{1}{\boldsymbol{\theta}_{k,0}^2+\sigma_k^2-\mathbf{c}^\intercal\mathbf{C}_-^{-1}\mathbf{c}}\mathbf{c}^\intercal\mathbf{C}_-^{-1} & \frac{1}{\boldsymbol{\theta}_{k,0}^2+\sigma_k^2-\mathbf{c}^\intercal\mathbf{C}_-^{-1}\mathbf{c}}\end{pmatrix},\\
\mathbf{y}_+^\intercal\mathbf{C}_+^{-1}\mathbf{y}_+ &= \mathbf{y}_-^\intercal\mathbf{C}_-^{-1}\mathbf{y}_- + \frac{\mathbf{y}_-^\intercal\mathbf{C}_-^{-1}\mathbf{c}\mathbf{c}^\intercal\mathbf{C}_-^{-1}\mathbf{y}_-}{\boldsymbol{\theta}_{k,0}^2 + \sigma_k^2 - \mathbf{c}^\intercal\mathbf{C}_-^{-1}\mathbf{c}} - \frac{2y_i\mathbf{y}_-^\intercal\mathbf{C}_-^{-1}\mathbf{c}}{\boldsymbol{\theta}_{k,0}^2 + \sigma_k^2 - \mathbf{c}^\intercal\mathbf{C}_-^{-1}\mathbf{c}+} + \frac{y_i^2}{\boldsymbol{\theta}_{k,0}^2 + \sigma_k^2 - \mathbf{c}^\intercal\mathbf{C}_-^{-1}\mathbf{c}}.
\end{aligned}$$

Therefore,

$$\log\frac{p(\mathbf{y}_k(\tilde{\mathbf{z}}_{-i}\cup\{\tilde{z}_i = k\})|\mathbf{X}_k(\tilde{\mathbf{z}}_{-i}\cup\{\tilde{z}_i = k\}); \boldsymbol{\theta}_k)}{p(\mathbf{y}_{-i,k}(\tilde{\mathbf{z}}_{-i})|\mathbf{X}_{-i,k}(\tilde{\mathbf{z}}_{-i}); \boldsymbol{\theta}_k)} = -\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\boldsymbol{\theta}_{k,0}^2 + \sigma_k^2 - \mathbf{c}^\intercal\mathbf{C}_-^{-1}\mathbf{c}) - \frac{(\mathbf{y}_-^\intercal\mathbf{C}_-^{-1}\mathbf{c} - y_i)^2}{2(\boldsymbol{\theta}_{k,0}^2 + \sigma_k^2 - \mathbf{c}^\intercal\mathbf{C}_-^{-1}\mathbf{c})}, \quad \text{(B1)}$$

which is exactly eq (5).

VHEM v.s. LOOCV. Equation (5) (or equivalently, eq. (B1)) gives an alternative perspective of the iteration formulae. We find that eq. (5) is exactly the log-probability of $y_i|\mathbf{y}_{-i,k}, \mathbf{X}_{-i,k}, \mathbf{x}_i$ given $\boldsymbol{\theta}_k$. Therefore, the update of $\tilde{z}_i$ can be rewritten as

$$\begin{aligned}
\tilde{z}_i &= \arg\max_{k=1,2,\cdots,K} \pi_k p(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)p(y_i|\mathbf{y}_{-i,k}(\tilde{\mathbf{z}}_{-i}), \mathbf{X}_{-i,k}(\tilde{\mathbf{z}}_{-i}), \mathbf{x}_i; \boldsymbol{\theta}_k)\\
&= \arg\max_{k=1,2,\cdots,K} \pi_k p\left((\mathbf{x}_i, y_i)|\mathbf{y}_{-i,k}(\tilde{\mathbf{z}}_{-i}), \mathbf{X}_{-i,k}(\tilde{\mathbf{z}}_{-i}); \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \boldsymbol{\theta}_k\right).
\end{aligned} \quad \text{(B2)}$$

Table C1 Parameter settings of $\mathcal{S}_1$, $\mathcal{S}_7$ and $\mathcal{S}_{16}$.

Dataset	Component	π_k	$\boldsymbol{\mu}_k$	$\boldsymbol{\Sigma}_k$	$[\theta_0, \theta_1, \sigma]$
$\mathcal{S}_1$	1	1/3	-6	1.5	$[2.00, 3.33, 0.10]$
	2	1/3	0	1.5	$[0.75, 10.00, 0.10]$
	3	1/3	6	1.5	$[1.50, 1.25, 0.10]$
$\mathcal{S}_7$	1	0.2	-12	1.5	$[2.00, 3.33, 0.10]$
	2	0.2	-6	1.5	$[0.75, 10.00, 0.10]$
	3	0.2	0	1.5	$[1.50, 1.25, 0.10]$
	4	0.2	6	1.5	$[0.50, 2.50, 0.10]$
	5	0.2	12	1.5	$[1.50, 5.00, 0.10]$
$\mathcal{S}_{16}$	1	0.25	$[-3, -3]$	$\begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}$	$[2.00, 3.33, 0.30]$
	2	0.25	$[-3, 3]$	$\begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}$	$[0.75, 10.00, 0.30]$
	3	0.25	$[3, -3]$	$\begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}$	$[1.50, 1.25, 0.30]$
	4	0.25	$[3, 3]$	$\begin{bmatrix}2 & 0\\0 & 2\end{bmatrix}$	$[0.50, 2.50, 0.30]$

This formula is rather intuitive: we assign z_i to be the class that best explains $(\mathbf{x}_i, y_i)$ given all other samples. Similar formulae also appears in the LOOCV (**L**eave-**O**ne-**O**ut **C**ross **V**alidation) algorithm. However, the LOOCV algorithm is established for the discriminative mixture of Gaussian processes, which is a different structure of MGP, and leave-one-out probabilities are used as gating functions. Besides, the LOOCV algorithm is derived by maximizing the leave-one-out likelihood, while the VHEM algorithm is based on performing variational inference and hardcut approximation on data likelihood. Equation (B2) can also be used for a more efficient implementation of the VHEM algorithm, which only requires computing one Gaussian process likelihood.

Appendix C Experimental Results

We conduct experiments on various typical synthetic datasets. These synthetic datasets are referred to as $\mathcal{S}_1, \cdots, \mathcal{S}_{16}$, respectively. Among them, $\mathcal{S}_1$ and $\mathcal{S}_7$ are basic datasets which have 3 and 5 Gaussian processes components respectively. The parameter settings of $\mathcal{S}_1$ and $\mathcal{S}_7$ are listed in table C1. Other datasets excluding $\mathcal{S}_{16}$ are modified based on $\mathcal{S}_1$ and $\mathcal{S}_7$ by varying the noise level, mixing proportions and overlapping level. Specifically, $\mathcal{S}_2 - \mathcal{S}_6$ are modified based on $\mathcal{S}_1$ as follows:
(a) $\mathcal{S}_2$ (a more noisy dataset): $\sigma_1 = \sigma_2 = \sigma_3 = 0.2$.
(b) $\mathcal{S}_3$ (a less noisy dataset): $\sigma_1 = \sigma_2 = \sigma_3 = 0.05$.
(c) $\mathcal{S}_4$ (an unbalanced dataset): $\pi_1 = 0.2, \pi_2 = 0.5, \pi_3 = 0.3$.
(d) $\mathcal{S}_5$ (a mildly overlapping dataset): $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}_3 = 1.0$.
(e) $\mathcal{S}_6$ (a heavily overlapping dataset): $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}_3 = 2.0$.
$\mathcal{S}_8 - \mathcal{S}_{12}$ are generated based on $\mathcal{S}_7$ in a similar way. We generate 900 samples in $\mathcal{S}_1 - \mathcal{S}_6$, 1500 samples in $\mathcal{S}_7 - \mathcal{S}_{12}$. $\mathcal{S}_{13} - \mathcal{S}_{15}$ are also modified based on $\mathcal{S}_1$, but in these datasets the task is more challenging. Specifically, in $\mathcal{S}_{13}$ we set $\sigma_1 = \sigma_2 = \sigma_3 = 0.5$, and in $\mathcal{S}_{14}$ we set $\sigma_1 = \sigma_2 = \sigma_3 = 1.0$. Therefore, the noise levels are relatively high in $\mathcal{S}_{13}$ and $\mathcal{S}_{14}$. As for $\mathcal{S}_{15}$, we set $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}_3 = 3.0$, thus the components are heavily overlapped. In $\mathcal{S}_{16}$, the input variable $\mathbf{x}$ is two-dimensional, and there are 4 Gaussian process components. The parameter settings of $\mathcal{S}_{16}$ are also listed in table C1. For each dataset, 1/3 of the samples are randomly selected for training, and the rest samples are used for testing. These datasets are illustrated in fig. C1.

For comparison, we also report the results of other MGP learning algorithms such as hardcut EM, MCMC EM, and LOOCV, as well as typical non-linear regression methods: a single Gaussian process, Support Vector Regression (SVR), and Feedforward Neural Network (FNN). Besides, to illustrate the superiority of the MGP model, we consider k-means GP, which split the datasets into several parts via the k-means algorithm and then trains GP in each component separately. We use the squared exponential covariance function in all Gaussian processes. For SVR, we use the Gaussian kernel with an adaptive kernel scale. The feedforward neural network consists of three hidden layers with 10, 10 and 5 neurons respectively. The structure of the FNN is selected from 30 possible configurations by 5-fold cross-validation. We set the number of components equal to the ground-truth K in the generative model. For MCMC EM algorithm, we generate 25 samples in each E-step. For the hardcut EM algorithm, the VHEM algorithm and the LOOCV algorithm, since the latent variables $\mathbf{z}$ are deterministic in the M-step, we terminate the iteration once we find $\mathbf{z}$ remains invariant in two consecutive EM iterations.

We evaluate these methods by the Rooted Mean Square Error (RMSE) and running time. Suppose $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ are testing samples, and $\{\hat{y}_i\}_{i=1}^M$ are prediction results, then the RMSE is defined by

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}\,. \tag{C1}$$

The results are reported in table C2. The results of these methods may be influenced by the initialization, therefore the reported results are averaged over 10 runs. We observe that the VHEM algorithm outperforms the hardcut EM algorithm consistently. While the MCMC EM algorithm obtains better results on some datasets, it spends significantly longer running time. The VHEM is a little slower than the hardcut EM algorithm since it involves iterations in the E-step, but it obtains comparable or even better results than the MCMC EM algorithm. Theoretically, the MCMC EM is supposed to obtain the best result since there is no approximation in the E-step. However, the Gibbs sampling steps may introduce fluctuations. More severely, it is challenging to diagnose the convergence of the Markov chain in practice, and it may take prohibitively long to converge. Therefore, the results of the MCMC EM algorithm usually have larger standard deviations, and sometimes the average RMSEs are even not as good as VHEM. Besides, the differences hardcut EM, MCMC EM, and VHEM are relatively small when the overlapping level is low (*i.e.*, on $\mathcal{S}_5$ and $\mathcal{S}_{11}$), and relatively large when the overlapping level is high (*i.e.*, on $\mathcal{S}_6$, $\mathcal{S}_{12}$ and $\mathcal{S}_{15}$). This is consistent with the theoretical analysis in appendix Appendix B. When the components are mildly overlapped, the results of k-means GP are comparable or even better than MGP methods. However, when the components are heavily overlapped, MGP (MCMC EM) and

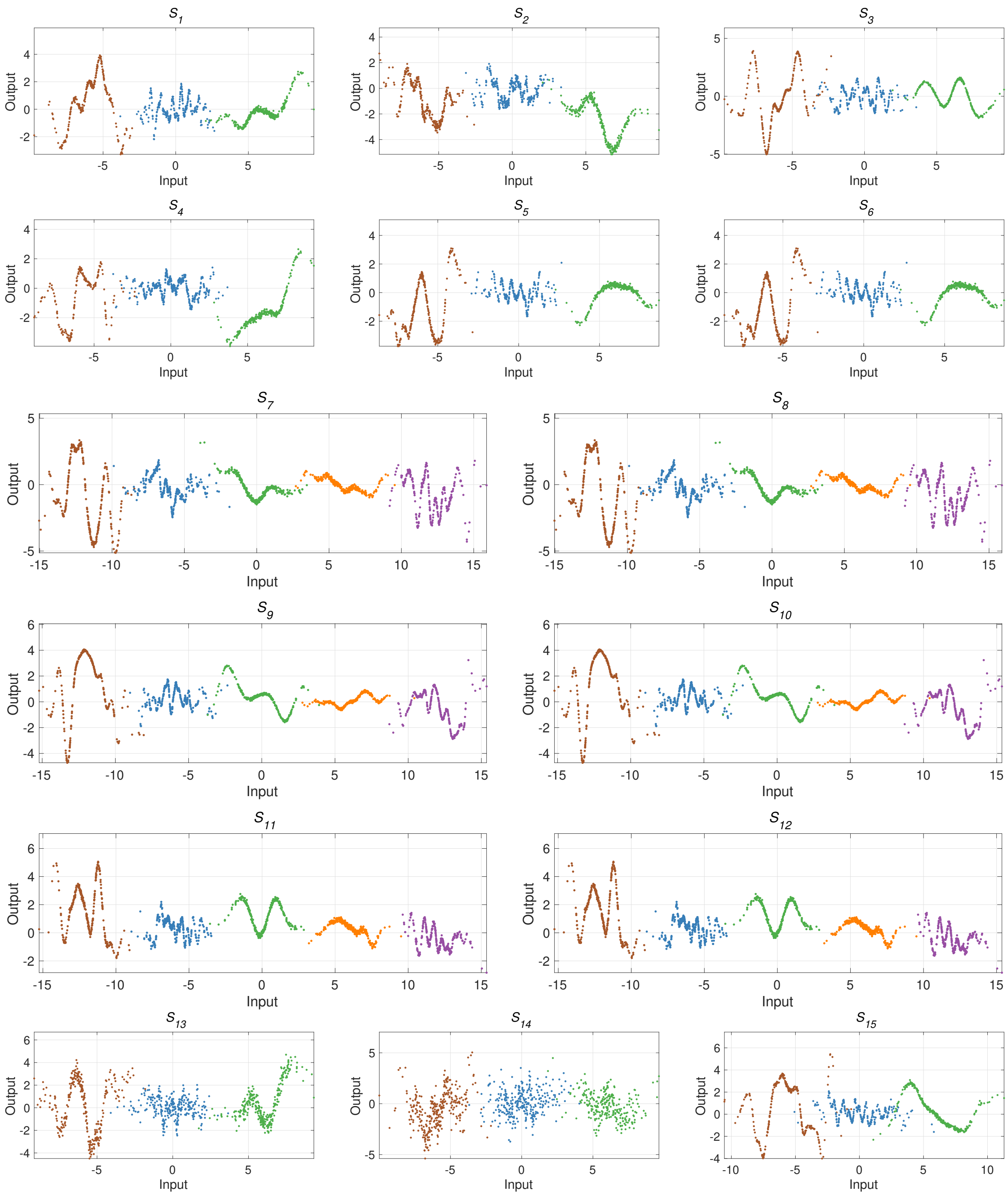

Figure C1 Illustrations of the typical synthetic datasets $\mathcal{S}1 - \mathcal{S}_{15}$. Samples belonging to different components are shown in different colors.

Table C2 Average rooted mean square errors and running times (in seconds) of various methods on synthetic datasets $\mathcal{S}_1$-$\mathcal{S}_{16}$. The results are averaged over 10 runs, and the best results are in bold.

Method \ Dataset	$\mathcal{S}_1$ RMSE	$\mathcal{S}_1$ Time	$\mathcal{S}_2$ RMSE	$\mathcal{S}_2$ Time	$\mathcal{S}_3$ RMSE	$\mathcal{S}_3$ Time	$\mathcal{S}_4$ RMSE	$\mathcal{S}_4$ Time
GP	0.2986 ± 0.0000	0.56 ± 0.11	0.4216 ± 0.0000	0.58 ± 0.03	0.2853 ± 0.0000	0.58 ± 0.05	0.3602 ± 0.0000	0.58 ± 0.02
SVR	0.5775 ± 0.0270	0.01 ± 0.01	0.5365 ± 0.0204	0.01 ± 0.01	0.7280 ± 0.0801	0.01 ± 0.00	0.5054 ± 0.0318	0.01 ± 0.00
FNN	0.5390 ± 0.0728	0.35 ± 0.15	0.5510 ± 0.0784	0.27 ± 0.12	0.4585 ± 0.0942	0.46 ± 0.22	0.4339 ± 0.0775	0.27 ± 0.07
k-means GP	0.2281 ± 0.0000	0.41 ± 0.00	0.4339 ± 0.0000	0.41 ± 0.00	0.3504 ± 0.1040	0.44 ± 0.01	0.3475 ± 0.0100	0.45 ± 0.00
MGP (LOOCV)	0.1947 ± 0.0000	1.28 ± 0.19	0.4462 ± 0.0000	0.48 ± 0.11	0.3501 ± 0.0000	17.87 ± 0.22	0.3478 ± 0.0006	28.56 ± 18.96
MGP (hardcut EM)	0.2340 ± 0.0000	0.70 ± 0.10	0.4129 ± 0.0000	0.68 ± 0.02	0.2686 ± 0.0000	1.15 ± 0.06	0.2834 ± 0.0000	1.50 ± 0.04
MGP (MCMC EM)	0.1639 ± 0.0037	312.31 ± 130.79	0.3079 ± 0.0014	173.91 ± 97.31	0.2336 ± 0.0060	460.59 ± 72.63	0.2306 ± 0.0012	361.89 ± 118.68
MGP (VHEM)	$\mathbf{0.1633 \pm 0.0000}$	6.02 ± 0.05	$\mathbf{0.3050 \pm 0.0000}$	5.94 ± 0.03	$\mathbf{0.2278 \pm 0.0000}$	8.10 ± 0.09	$\mathbf{0.2304 \pm 0.0000}$	10.30 ± 0.29

Method \ Dataset	$\mathcal{S}_5$ RMSE	$\mathcal{S}_5$ Time	$\mathcal{S}_6$ RMSE	$\mathcal{S}_6$ Time	$\mathcal{S}_7$ RMSE	$\mathcal{S}_7$ Time	$\mathcal{S}_8$ RMSE	$\mathcal{S}_8$ Time
GP	0.3014 ± 0.0000	0.65 ± 0.01	0.3345 ± 0.0000	0.44 ± 0.00	0.4447 ± 0.0000	2.48 ± 0.01	0.5437 ± 0.0000	1.89 ± 0.01
SVR	0.4406 ± 0.0232	0.01 ± 0.01	0.4981 ± 0.0309	0.01 ± 0.00	1.1147 ± 0.0657	0.02 ± 0.00	0.7824 ± 0.0215	0.02 ± 0.00
FNN	0.4480 ± 0.0648	0.25 ± 0.07	0.4144 ± 0.0454	0.29 ± 0.06	0.9026 ± 0.2473	0.35 ± 0.15	0.6020 ± 0.0902	0.47 ± 0.24
k-means GP	0.1813 ± 0.0000	0.42 ± 0.00	0.2962 ± 0.0000	0.37 ± 0.00	0.3469 ± 0.0094	0.73 ± 0.04	0.4722 ± 0.0301	0.73 ± 0.02
MGP (LOOCV)	0.1743 ± 0.0000	12.92 ± 0.38	0.2615 ± 0.0000	7.09 ± 0.06	0.3451 ± 0.0424	4.47 ± 0.66	0.8197 ± 0.1615	3.22 ± 3.59
MGP (hardcut EM)	0.1724 ± 0.0000	1.10 ± 0.02	0.2986 ± 0.0000	2.28 ± 0.01	0.3408 ± 0.0004	1.30 ± 0.05	0.4739 ± 0.0000	2.87 ± 0.03
MGP (MCMC EM)	0.1689 ± 0.0025	157.50 ± 96.41	$\mathbf{0.2229 \pm 0.0047}$	408.96 ± 106.23	$\mathbf{0.3139 \pm 0.0137}$	722.66 ± 411.33	0.3208 ± 0.0084	740.45 ± 376.68
MGP (VHEM)	$\mathbf{0.1675 \pm 0.0000}$	5.97 ± 0.04	0.2256 ± 0.0000	13.09 ± 0.07	0.3172 ± 0.0000	33.10 ± 2.54	$\mathbf{0.3162 \pm 0.0000}$	26.39 ± 0.14

Method \ Dataset	$\mathcal{S}_9$ RMSE	$\mathcal{S}_9$ Time	$\mathcal{S}_{10}$ RMSE	$\mathcal{S}_{10}$ Time	$\mathcal{S}_{11}$ RMSE	$\mathcal{S}_{11}$ Time	$\mathcal{S}_{12}$ RMSE	$\mathcal{S}_{12}$ Time
GP	0.3186 ± 0.0000	1.96 ± 0.01	0.3139 ± 0.0000	2.21 ± 0.01	0.2652 ± 0.0000	3.45 ± 0.01	0.4067 ± 0.0000	2.07 ± 0.01
SVR	0.8760 ± 0.0590	0.02 ± 0.00	0.8818 ± 0.0304	0.02 ± 0.00	0.8447 ± 0.0355	0.02 ± 0.01	1.0229 ± 0.0584	0.02 ± 0.00
FNN	0.5602 ± 0.0764	0.40 ± 0.16	0.5815 ± 0.1120	0.46 ± 0.19	0.5261 ± 0.0467	0.45 ± 0.19	0.6737 ± 0.0748	0.37 ± 0.10
k-means GP	0.2905 ± 0.0000	0.67 ± 0.00	0.2827 ± 0.0023	0.74 ± 0.03	$\mathbf{0.1682 \pm 0.0003}$	0.73 ± 0.00	0.4009 ± 0.0000	0.63 ± 0.01
MGP (LOOCV)	0.2880 ± 0.0000	2.14 ± 0.07	0.2535 ± 0.0000	5.15 ± 0.11	0.2375 ± 0.0162	2.86 ± 3.07	0.3995 ± 0.0000	18.09 ± 0.09
MGP (hardcut EM)	0.2821 ± 0.0000	2.89 ± 0.09	0.2666 ± 0.0000	2.10 ± 0.01	0.1695 ± 0.0000	1.43 ± 0.02	0.3791 ± 0.0000	2.18 ± 0.02
MGP (MCMC EM)	0.2255 ± 0.0473	1236.03 ± 156.74	$\mathbf{0.2234 \pm 0.0035}$	681.89 ± 327.97	0.1694 ± 0.0002	744.05 ± 423.48	0.3402 ± 0.0014	704.16 ± 359.57
MGP (VHEM)	$\mathbf{0.2092 \pm 0.0161}$	26.88 ± 4.85	0.2248 ± 0.0000	31.48 ± 0.21	0.1695 ± 0.0000	6.12 ± 0.07	$\mathbf{0.3365 \pm 0.0000}$	27.34 ± 0.14

Method \ Dataset	$\mathcal{S}_{13}$ RMSE	$\mathcal{S}_{13}$ Time	$\mathcal{S}_{14}$ RMSE	$\mathcal{S}_{14}$ Time	$\mathcal{S}_{15}$ RMSE	$\mathcal{S}_{15}$ Time	$\mathcal{S}_{16}$ RMSE	$\mathcal{S}_{16}$ Time
GP	0.6980 ± 0.0000	0.59 ± 0.01	1.1408 ± 0.0000	0.72 ± 0.05	0.6659 ± 0.0000	0.76 ± 0.01	0.9737 ± 0.0000	4.93 ± 2.20
SVR	0.7514 ± 0.0175	0.01 ± 0.00	1.2630 ± 0.0163	0.04 ± 0.15	0.8577 ± 0.0497	0.02 ± 0.01	1.2807 ± 0.0205	0.04 ± 0.03
FNN	0.9356 ± 0.3466	0.32 ± 0.08	1.4482 ± 0.5019	0.39 ± 0.62	0.8646 ± 0.2342	0.54 ± 0.30	1.2415 ± 0.1202	0.69 ± 0.17
k-means GP	0.7545 ± 0.0000	0.43 ± 0.00	1.1386 ± 0.0000	0.58 ± 0.40	0.6425 ± 0.0023	0.49 ± 0.08	0.8923 ± 0.0004	0.96 ± 0.38
MGP (LOOCV)	0.7415 ± 0.0000	5.29 ± 0.04	1.1653 ± 0.0000	4.44 ± 0.58	0.7570 ± 0.0000	11.92 ± 0.68	0.9445 ± 0.0001	3.56 ± 0.86
MGP (hardcut EM)	0.6418 ± 0.0000	1.27 ± 0.02	1.1411 ± 0.0000	1.06 ± 0.20	0.6579 ± 0.0000	1.20 ± 0.05	0.8899 ± 0.0001	3.40 ± 0.22
MGP (MCMC EM)	0.6206 ± 0.0055	595.62 ± 50.75	$\mathbf{1.1361 \pm 0.0073}$	529.00 ± 124.51	0.6174 ± 0.0290	564.71 ± 36.73	0.8632 ± 0.0996	3889.66 ± 866.21
MGP (VHEM)	$\mathbf{0.6140 \pm 0.0000}$	11.11 ± 0.02	1.1366 ± 0.0000	5.59 ± 0.29	$\mathbf{0.6097 \pm 0.0000}$	9.89 ± 0.16	$\mathbf{0.8320 \pm 0.0042}$	32.58 ± 9.59

Table C3 Classification accuracy rates (CARs) of k-means GP, LOOCV, hardcut-EM, MCMC-EM and VHEM on synthetic datasets $\mathcal{S}_1$-$\mathcal{S}_{16}$. The best results are in bold.

Method \ Dataset	$\mathcal{S}_1$ Train	$\mathcal{S}_1$ Test	$\mathcal{S}_2$ Train	$\mathcal{S}_2$ Test	$\mathcal{S}_3$ Train	$\mathcal{S}_3$ Test	$\mathcal{S}_4$ Train	$\mathcal{S}_4$ Test	$\mathcal{S}_5$ Train	$\mathcal{S}_5$ Test	$\mathcal{S}_6$ Train	$\mathcal{S}_6$ Test	$\mathcal{S}_7$ Train	$\mathcal{S}_7$ Test	$\mathcal{S}_8$ Train	$\mathcal{S}_8$ Test
k-means GP	98.33%	**99.50**%	99.33%	**99.33**%	99.20%	**98.50**%	98.60%	98.58%	99.67%	**99.67**%	97.00%	**98.33**%	92.03%	92.46%	96.24%	95.84%
LOOCV	98.00%	99.00%	99.33%	**99.33**%	90.33%	93.00%	68.62%	68.62%	99.00%	**99.67**%	68.00%	65.83%	98.25%	**98.61**%	96.41%	95.64%
Hardcut EM	98.33%	99.17%	99.33%	**99.33**%	99.33%	**98.50**%	98.67%	98.50%	**100.00**%	**99.67**%	95.67%	97.33%	98.54%	98.43%	99.20%	98.70%
MCMC EM	**98.67**%	98.99%	**99.67**%	**99.33**%	**99.39**%	98.40%	98.98%	98.53%	99.50%	**99.67**%	97.63%	98.18%	94.78%	94.58%	99.67%	**98.80**%
VHEM	**98.67**%	99.00%	**99.67**%	**99.33**%	99.33%	98.33%	**99.00**%	**98.67**%	99.33%	**99.67**%	**98.00**%	98.00%	**98.80**%	98.60%	**99.80**%	**98.80**%

Method \ Dataset	$\mathcal{S}_9$ Train	$\mathcal{S}_9$ Test	$\mathcal{S}_{10}$ Train	$\mathcal{S}_{10}$ Test	$\mathcal{S}_{11}$ Train	$\mathcal{S}_{11}$ Test	$\mathcal{S}_{12}$ Train	$\mathcal{S}_{12}$ Test	$\mathcal{S}_{13}$ Train	$\mathcal{S}_{13}$ Test	$\mathcal{S}_{14}$ Train	$\mathcal{S}_{14}$ Test	$\mathcal{S}_{15}$ Train	$\mathcal{S}_{15}$ Test	$\mathcal{S}_{16}$ Train	$\mathcal{S}_{16}$ Test
k-means GP	98.40%	99.10%	96.97%	97.56%	**99.80**%	99.73%	97.40%	**97.40**%	98.67%	**98.33**%	**99.00**%	98.17%	94.55%	**94.67**%	96.57%	**96.90**%
LOOCV	98.00%	99.00%	98.40%	98.60%	94.99%	96.52%	96.00%	97.00%	**99.33**%	97.67%	98.67%	98.00%	**99.00**%	**94.67**%	96.59%	96.75%
Hardcut EM	**98.60**%	**99.20**%	98.40%	98.40%	**99.80**%	**99.80**%	96.80%	97.30%	98.67%	**98.33**%	**99.00**%	**98.50**%	94.67%	**94.67**%	96.54%	96.73%
MCMC EM	97.44%	96.85%	**99.40**%	99.05%	**99.80**%	**99.80**%	**99.03**%	97.21%	99.08%	97.66%	98.72%	98.07%	98.12%	94.63%	**96.92**%	96.71%
VHEM	97.60%	97.50%	99.00%	**99.10**%	**99.80**%	**99.80**%	98.20%	97.30%	99.00%	97.50%	98.67%	98.00%	**99.00**%	**94.67**%	94.83%	95.14%

MGP (VHEM) outperform k-means GP significantly. The noise level of $\mathcal{S}_{13}$ is relatively high, and we still find that MGP (VHEM) achieves better results than other methods. However, if we further increase the noise level, the task becomes very difficult and the results of various methods are similar on $\mathcal{S}_{14}$. The results on $\mathcal{S}_{16}$ demonstrate the effectiveness of the VHEM algorithm when the dimension of the input variable is larger than 1. It is worth noting that the time consumption of the MCMC EM algorithm on $\mathcal{S}_{16}$ is very high because of expensive sampling steps, and the standard deviation of the result is relatively high, while the VHEM algorithm can obtain better results within one minute.

To further demonstrate the effectiveness of VHEM and compare it with hardcut EM and MCMC EM, we also consider their performances on component identification. Component identification concerns whether the algorithm successfully reveals the underlying mixture structure and correctly assigns the samples to their corresponding components. To evaluate the component identification performances, we calculate the Classification Accuracy Rate (CAR), which is defined as

$$\mathrm{CAR} = \max_{\xi \in \Pi_K} \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(z_i = \xi(\hat{z}_i)),$$

where Π_K denotes the set of K-permutations. Since there are no theoretical guarantees that the EM algorithm will classify the samples into correct clusters, this metric reflects the effectiveness of the algorithms in terms of revealing the mixture structure empirically. The CARs of four algorithms are shown in table C3. We can see that the component identification results of these algorithms are almost perfect on all datasets, except LOOCV. Hardcut EM, MCMC EM, and VHEM achieve comparable CARs and none of them has obvious advantages in general. However, when the overlapping level is high (*i.e.*, on $\mathcal{S}_6$, $\mathcal{S}_{12}$ and $\mathcal{S}_{15}$), the CARs of hardcut EM and k-means GP drop marginally, while VHEM still achieves high CARs. Besides, k-means GP rarely achieves the best CAR, therefore we conclude that it is necessary to consider the temporal structure to accurately model the data rather than directly dividing samples into groups based on the input variable alone.