A Specialized Probability Density Function for the Input of Mixture of Gaussian Processes

Longbo Zhao and Jinwen Ma(✉)

Department of Information Science, School of Mathematical Sciences and LMAM, Peking University, Beijing 100871, People's Republic of China
jwma@math.pku.edu.cn

Abstract. Mixture of Gaussian Processes (MGP) is a generative model being powerful and widely used in the fields of machine learning and data mining. However, when we learn this generative model on a given dataset, we should set the probability density function (pdf) of the input in advance. In general, it can be set as a Gaussian distribution. But, for some actual data like time series, this setting or assumption is not reasonable and effective. In this paper, we propose a specialized pdf for the input of MGP model which is a piecewise-defined continuous function with three parts such that the middle part takes the form of a uniform distribution, while the two side parts take the form of Gaussian distribution. This specialized pdf is more consistent with the uniform distribution of the input than the Gaussian pdf. The two tails of the pdf with the form of a Gaussian distribution ensure the effectiveness of the iteration of the hard-cut EM algorithm for MGPs. It demonstrated by the experiments on the simulation and stock datasets that the MGP model with these specialized pdfs can lead to a better result on time series prediction in comparison with the general MGP models as well as the other classical regression methods.

Keywords: Gaussian distribution · Mixture of Gaussian processes
Hard-cut EM algorithm · Probability density function
Time series prediction

1 Introduction

Gaussian process (GP) is a powerful model and widely used in machine learning and data mining [1–3]. However, there are two main limitations. Firstly, it cannot fit the multi-modal dataset well because GP model employs a global scale parameter [4]. Secondly, its parameter learning consumes $O(N^3)$ computational time [5,6], where N is the number of training samples. In order to overcome those difficulties, Tresp [4] proposed mixture of Gaussian processes (MGP) in 2000, which was developed from the mixture of experts. Since then, many kinds

Z. Shi et al. (Eds.): ICIS 2018, IFIP AICT 539, pp. 70–80, 2018.
https://doi.org/10.1007/978-3-030-01313-4_8

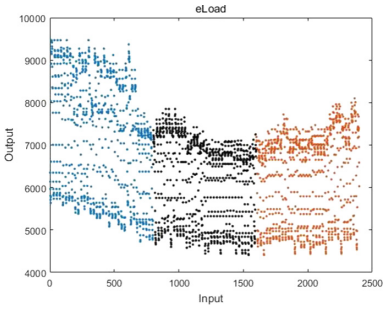

Fig. 1. The sketch of the eLoad data.

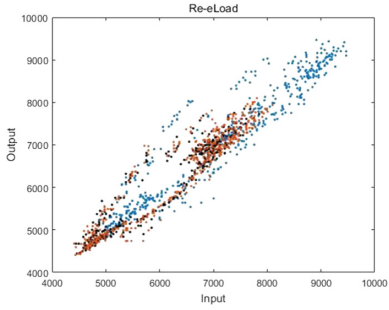

Fig. 2. The sketch of the transformed eLoad data.

of MGP model have been proposed and can be classified into two main forms: the generative model [7–10] and conditional model [4,6,11–13]. In comparison with the conditional model, the generative model has two main advantages: (1) The missing features can be easily inferred from the outputs; (2) The influence of the inputs on the outputs is more clear [8]. Therefore, many scholars have studied the generative model [14–20].

However, when we learn the generative model on a given dataset, we should set the probability density function (pdf) of the input in advance. In general, it can be set as a Gaussian distribution [14–20]. But, for some actual data like time series, this setting or assumption is not so reasonable and effective. When we learn MGP model on these actual data, we usually need to utilize the ARMA model [14–21] to transform the data, and then use the transformed data on the MGP model. However, this transformation can destroy the correlation of samples, which is very important for MGP model. Figure 1 shows the eLoad data [14] from which we can see that samples in three different colors (blue, black, and red) represent three temporally sequential samples, respectively. Figure 2 shows the transformed eLoad data from which we can find that three temporally sequential samples are mixed together and cannot be classified effectively. In this paper, we propose a specialized pdf for the input of the MGP model to solve this problem. As shown in Fig. 3, this pdf consists of three components. The left and right side parts are Gaussian distributions, while the middle is a uniform distribution. For the training of the MGP model, we use the hard-cut EM algorithm [17] as the basic learning framework for parameter estimation. Actually, the hard-cut EM algorithm can get better result than some popular learning algorithms.

The rest of the paper is organized as follows. Section 2 introduces the GP and MGP models. We describe the specialized probability density function in Sect. 3. We further propose the learning algorithm for the MGP model of the specialized pdfs in Sect. 4. The experimental results are contained in Sect. 5. Finally, we make a brief conclusion in Sect. 6.

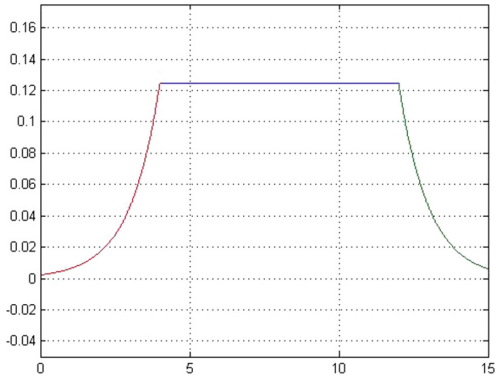

Fig. 3. The sketch of the specialized input distribution.

2 GP and MGP Models

2.1 GP Model

We mathematically define the GP model as follows:

$$\boldsymbol{Y} \sim N(m(\boldsymbol{X}), K(\boldsymbol{X}, \boldsymbol{X})) \tag{1}$$

where $\mathbf{D} = \{\boldsymbol{X}, \boldsymbol{Y}\} = \{(\boldsymbol{x}_i, y_i)\text{: } i = 1,2,...,N\}$, $\boldsymbol{x}_i$ denotes a d-dimensional input vector, and y_i is the corresponding output. $m(\boldsymbol{X})$ and $K(\boldsymbol{X},\boldsymbol{X})$ denote the mean vector and covariance matrix, respectively. Without loss of generality, we assume $m(\boldsymbol{X}) = \boldsymbol{0}$. There are many choices for covariance function, such as linear, Gaussian noise, squared exponential function and so on. Here, we adopt the squared exponential (SE) covariance function [10]:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j; \theta) = \sigma_f^2 exp(-\frac{\sigma_l^2}{2}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2) + \sigma_n^2 I_{(i=j)} \tag{2}$$

where $\boldsymbol{\theta} = \{\sigma_f^2,\sigma_l^2,\sigma_n^2\}$ denote the vector. On the given sample dataset $\mathbf{D}$, the log-likelihood function can be expressed as follows:

$$\log p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta}) = \log N(\boldsymbol{Y}|\boldsymbol{0}, K(\mathbf{X}, \mathbf{X})) \tag{3}$$

In order to obtain the estimation of parameters $\boldsymbol{\theta}$, we perform the maximum likelihood estimation (MLE) procedure [10], that is, we get

$$\hat{\boldsymbol{\theta}} = argmax_{\boldsymbol{\theta}} \log N(\boldsymbol{Y}|\boldsymbol{0}, K(\boldsymbol{X}, \boldsymbol{X})) \tag{4}$$

2.2 MGP Model

Denote C and N as the number of GP components and training samples in the MGP model, respectively. On the basis of the GP model, we define MGP model by the following steps:

Step 1. Partition samples into each GP components by the Multinomial distribution:

$$p(z_n = c) = \pi_c \tag{5}$$

where $c = 1,...,C$ and $n = 1,...,N$.

Step 2. Accordingly, each input $\boldsymbol{x}_i$ fulfills the following distribution:

$$p(\boldsymbol{x}_i|z_n = c) \sim p(\boldsymbol{x}|\boldsymbol{\psi}_c) \tag{6}$$

where $\{\boldsymbol{\psi}_c : c = 1, ..., C\}$ is the parameter set. In general, $p(\boldsymbol{x}|\boldsymbol{\psi}_c)$ is a Gaussian distribution.

Step 3. Denote $\boldsymbol{I}_c = \{n|z_n = c\}$, $\boldsymbol{X}_c = \{\boldsymbol{x}_n|z_n = c\}$, $\boldsymbol{Y}_c = \{y_n|z_n = c\}$ (c=1,...,C, n=1,...,N) as the sample indexes, inputs and outputs of the training samples in the *c-th* component, respectively. Given $\boldsymbol{X}_c$, the corresponding *c-th* GP component can be mathematically defined as follows:

$$\boldsymbol{Y}_c \sim N(\boldsymbol{0}, K(\boldsymbol{X}_c, \boldsymbol{X}_c)) \tag{7}$$

where $K(\boldsymbol{X}_c,\boldsymbol{X}_c)$ is given by Eq.(2) with the hyper-parameter $\boldsymbol{\theta}_c = \{\sigma_{fc}^2, \sigma_{lc}^2, \sigma_{nc}^2\}$.

Based on Eqs. (5), (6) and (7), we mathematically define the MGP model. The log-likelihood function is derived as follows:

$$\begin{aligned}\log(p(\boldsymbol{Y}_c|\boldsymbol{X}_c, \boldsymbol{\Theta}, \boldsymbol{\Psi})) = \sum_{c=1}^{C}(\sum_{n\in \boldsymbol{I}_c} (\log(\pi_c p(\boldsymbol{x}_n|\boldsymbol{\mu}_c, \boldsymbol{S}_c))) \\ + \log(p(\boldsymbol{Y}_c|\boldsymbol{X}_c, \boldsymbol{\theta}_c)))\end{aligned} \tag{8}$$

where $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_c : c = 1, ..., C\}$ and $\boldsymbol{\Psi} = \{\boldsymbol{\psi}_c, \pi_c : c = 1, ..., C\}$ denote the hyper-parameters and parameters of the MGP model, respectively.

3 Specialized Input Distribution and Its Learning Algorithm

For many real world datasets, such as UCI machine learning repository, Gaussian distribution is not appropriate for the input. In order to solve this problem, we propose a specialized distribution for this situation.

3.1 Specialized PDF

This specialized distribution is a piecewise-defined continuous function, which consists of three parts, the middle part is a uniform distribution density, both sides are Gaussian distribution densities, shown in Fig. 3. We mathematically defined the specialized distribution as follows:

$$P(\boldsymbol{x}; \boldsymbol{\psi}) = \begin{cases} \frac{\lambda_1}{(\sqrt{2\pi}\tau_1)} \exp^{-\frac{(x-a)^2}{2\tau_1^2}} & \boldsymbol{x} < \boldsymbol{a} \\ \lambda & \boldsymbol{a} \le \boldsymbol{x} \le \boldsymbol{b} \\ \frac{\lambda_2}{(\sqrt{2\pi}\tau_2)} \exp^{-\frac{(x-b)^2}{2\tau_2^2}} & \boldsymbol{x} > \boldsymbol{b} \end{cases} \tag{9}$$

where we redefine $\boldsymbol{\psi}$=$\{\lambda, \lambda_1, \lambda_2, \tau_1, \tau_2, \boldsymbol{a}, \boldsymbol{b}\}$ as the parameter vector.

3.2 Learning Algorithm for the Specialized PDF

In order to learn $\boldsymbol{\psi}$, we set that the input interval ($\boldsymbol{a}$,$\boldsymbol{b}$) contains the number of the samples with probability p_0. Denote $\boldsymbol{X}$ and N as the training sample set and the number of training sample, respectively. We summarize the algorithm framework as following steps:

Step 1. Learn $\boldsymbol{a}$, $\boldsymbol{b}$, and λ:

$$\boldsymbol{a} = X_{\frac{N(1-p_0)}{2}}; \boldsymbol{b} = X_{\frac{N(1+p_0)}{2}}; \lambda = \frac{p_0}{(\boldsymbol{b}-\boldsymbol{a})} \tag{10}$$

where $p(x< X_{\frac{N(1-p_0)}{2}} \ | x \in \boldsymbol{X}) = \frac{(1-p_0)}{2}$. In order to reduce the effect of the misclassified (or outlier) point on the middle part, we estimate $\boldsymbol{a}$ and $\boldsymbol{b}$ as Eq.(10) do.

Step 2. Estimate λ_1, λ_2, τ_1 and τ_2.

Denote p_1 and p_2 as the sample ratio at both left side and right side, respectively. The probability density function is continuously integrable, and the integral of the probability density function is equal to 1. In other word:

$$\int P(\boldsymbol{x};\boldsymbol{\psi})dx = \begin{cases} p_1 & \boldsymbol{x} < \boldsymbol{a} \\ p_0 & \boldsymbol{a} \le \boldsymbol{x} \le \boldsymbol{b} \\ p_2 & \boldsymbol{x} > \boldsymbol{b} \end{cases}; \qquad p_0 + p_1 + p_2 = 1 \tag{11}$$

According to the continuity of the probability density function, we only need do same simple calculations to get $\{\lambda_1, \lambda_2, \tau_1, \tau_2\}$:

$$\lambda_1 = 2p_1; \lambda_2 = 2p_2; \tau_1 = \frac{\lambda_1}{\sqrt{2\pi\lambda}}; \tau_2 = \frac{\lambda_2}{\sqrt{2\pi\lambda}} \tag{12}$$

4 The MGP Model of the Specialized PDFs and Its Learning Algorithm

We now consider the MGP model with these specialized pdfs. For the parameter learning of the MGP model, there are main three kinds of learning algorithms: MCMC methods [22,23], variational Bayesian inference [24,25], and EM algorithm [5,9,11]. However, the MCMC methods and variational Bayesian inference methods have their own limitations: the time complexity of the MCMC method is very high, and variational Bayesian inference may lead to a rather deviation from the true objective function. EM algorithm is an important and effective iterative algorithm to do maximum likelihood or maximum a posterior(MAP) estimates of parameters for mixture model. However, for such a complex MGP model, the posteriors of latent variables and Q function are rather complicated. In order to overcome this difficulty, we implement the hard-cut EM algorithm [17] to learn parameter, which makes certain approximations in E-step.

Denote $\boldsymbol{z}_{nc}$ be the latent variables, where $\boldsymbol{z}_{nc}$ is a Kronecker delta function, $\boldsymbol{z}_{nc} = 1$, if the sample ($\boldsymbol{x}_n$,y_n) belongs to the *c-th* GP component. Therefore,

we can obtain the log likelihood function of the complete data from Eq. (8) as follows:

$$\log(p(\boldsymbol{Y}, \boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\Theta}, \boldsymbol{\Psi})) = \sum_{c=1}^{C}(\sum_{n=1}^{N}(\boldsymbol{z}_{nc}\log(\pi_c p(\boldsymbol{x}_n|\boldsymbol{\psi}_c))) + \log(p(\boldsymbol{Y}_c|\boldsymbol{X}_c, \boldsymbol{\theta}_c))) \tag{13}$$

The main idea of hard-cut EM algorithm can be expressed as the following steps:

E-step. Assign the samples to the corresponding GP component according to the maximum a posterior (MAP) criterion:

$$\widehat{k}_n = argmax_{1\leq c\leq C}\{\pi_c p(\boldsymbol{x}_n|\boldsymbol{\psi}_c)p(y_n|\boldsymbol{\theta}_c)\} \tag{14}$$

that is, latent variable $\boldsymbol{z}_{\widehat{k}_n n}$=1.

M-step. With the known partition, we can estimate the parameters $\boldsymbol{\Psi}$ and hyper-parameters $\boldsymbol{\Theta}$ via the MLE procedure:

(1) For learning the parameters $\{\boldsymbol{\psi}_c\}_c$, we perform the learning algorithm in the last section.
(2) For estimating the hyper-parameter $\boldsymbol{\Theta}$, we perform the MLE procedure on each *c-th* component to estimate $\boldsymbol{\theta}_c$ as shown in Eq. (4).

5 Experimental Results

In order to test the accuracy and effectiveness of the specialized pdf for MGP model, we carry out several experiments on the simulation and stock datasets. We employ the root mean squared error (RMSE) to measure the prediction accuracy, which is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N}(\boldsymbol{y}_n - \hat{\boldsymbol{y}}_n)^2}{N}} \tag{15}$$

where $\hat{\boldsymbol{y}}_n$ and $\boldsymbol{y}_n$ denote the predicted value and true value, respectively. We also compare our algorithm with some classical machine learning algorithms: kernel, RBF, SVM, and denote 'OURS' as our proposed model with the hard-cut EM algorithm.

5.1 Simulation Experiments

In the simulation experiments, we generate three groups of synthetic datasets from MGP model. those three MGP models contain 4, 6, 8 GP components, respectively. The number of samples in each group is 2600, 3900, 5000, respectively. In each group, there are three datasets, which are the same except the degree of overlap. Figure 4 shows the dataset with the smallest degree of overlap

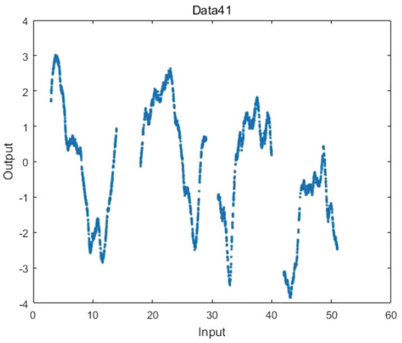

Fig. 4. The dataset with the least degree of overlap from MGP with 4 components.

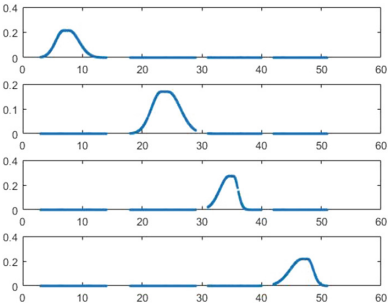

Fig. 5. The distribution of the probability density function of the input on each Gaussian processes component.

with 4 GP components. On each group dataset, we run each algorithm 100 times, and randomly extract training samples and test samples, where 1/3 are training samples and other 2/3 are test samples. The RMSE of each algorithm is listed in Table 1. From the Table 1, We can see that our proposed algorithm obtains the better results. Figure 5 shows the specialized pdfs on the first group dataset with the smallest overlapping degree. We can obtain that the specialized pdf at both ends of the data in the form of a Gaussian distribution of attenuation, the specialized pdf in the middle of the data is a uniform distribution. This shape of the specialized pdf is more consistent with the uniform distribution than Gaussian distribution. The attenuation of both ends of the specialized pdf in the form of a Gaussian distribution ensures the effectiveness of the iteration of the hard-cut EM algorithm. Then, the class label of the samples can be updated according to the MAP criteria in the iteration of hard-cut EM algorithm. If we apply uniform distribution only, the iterative steps of hard-cut EM algorithm is invalid.

5.2 Prediction on Stock Data

In this section, we obtain the closing price data of three stocks from Shanghai Stock Exchange, and the IDs are 300015, 002643, and 601058, respectively.

From Eq. (10), we can know that the specialized pdf is closely related to the interval length of the middle data. In order to check the effect of different input lengths on the prediction accuracy of the algorithm, we do some transformations on the input. Since the range of output changes is too large, we use a linear function to narrow the output down to the same range as the synthetic data. In summary, we transform the datasets as follows:

(i) Transform the input as following equation:

$$X_n = \frac{n}{\delta} \tag{16}$$

where $i = 1,...,N$, N is the sample number, $\delta = \{101, 51, 23, 11, 7, 3, 1\}$.

Table 1. The RMSEs of the four algorithms on the three groups.

C = 4	Data41	Data42	Data43
Kernel	0.2143 ± 0.0000	0.6871 ± 0.0000	0.6537 ± 0.0000
RBF	0.1965 ± 0.0000	0.7065 ± 0.0000	0.6594 ± 0.0000
SVM	0.2548 ± 0.0002	0.3739 ± 0.0060	0.3704 ± 0.0038
OURS	0.0604 ± 0.0003	0.2991 ± 0.0319	0.3326 ± 0.0174
C = 6	Data61	Data62	Data63
Kernel	0.5212 ± 0.0000	0.5211 ± 0.0000	0.5392 ± 0.0000
RBF	0.5411 ± 0.0000	0.5498 ± 0.0000	0.5554 ± 0.0000
SVM	0.2678 ± 0.0013	0.3378 ± 0.0033	0.3942 ± 0.0080
OURS	0.2558 ± 0.0366	0.3653 ± 0.0384	0.3392 ± 0.0210
C = 8	Data81	Data82	Data83
Kernel	0.4247 ± 0.0000	0.4245 ± 0.0000	0.4993 ± 0.0000
RBF	0.4530 ± 0.0000	0.4265 ± 0.0000	0.4934 ± 0.0000
SVM	0.3642 ± 0.003	0.4238 ± 0.0023	0.4831 ± 0.0017
OURS	0.3220 ± 0.0335	0.3639 ± 0.0898	0.4668 ± 0.3434

Table 2. The RMSEs of the four algorithms on the three groups of the transformed stock datasets.

300015	X1	X2	X3	X4	X5	X6	X7
Kernel	0.6347	0.5439	0.4121	0.2753	0.2241	0.1846	0.2149
RBF	0.5218	0.3717	0.2918	0.2671	0.3435	0.6930	0.9934
SVM	0.3634	0.2532	0.1971	0.1707	0.1869	0.1707	0.1842
OURS	0.3531	0.2325	0.1784	0.1676	0.1601	0.1467	0.1583
002643	X1	X2	X3	X4	X5	X6	X7
Kernel	0.6487	0.4722	0.3116	0.2323	0.1972	0.1533	0.1542
RBF	0.5357	0.4462	0.2744	0.2280	0.2116	0.2110	0.8882
SVM	0.2815	0.2487	0.2364	0.2175	0.1998	0.1902	0.1792
OURS	0.2681	0.2024	0.1835	0.1799	0.1930	0.1805	0.1544
601058	X1	X2	X3	X4	X5	X6	X7
Kernel	0.7200	0.5267	0.3939	0.2738	0.2150	0.1503	0.1256
RBF	0.5296	0.4547	0.3636	0.2644	0.2279	0.2785	0.9705
SVM	0.4079	0.2325	0.1748	0.1502	0.1439	0.1439	0.1458
OURS	0.3271	0.1966	0.1318	0.1507	0.1545	0.1315	0.1472

(ii) Transform the output by a linearly compressed, and the compressed interval is $[-4.5, 4.5]$.

$$\tilde{y} = \frac{9y}{M-m} + \frac{4.5}{M-m} \tag{17}$$

where M and m denote the maximum value and minimum value of the stock, respectively.

Through the above transformations, each stock can produce 7 datasets. In each 7 datasets of three stock datasets, we repeat each regression algorithm 100 times, and randomly extracted 1/3 as training samples and the other 2/3 as test samples. The RMSE of each algorithm on those three transformed stock datasets is listed in Table 2. From Table 2, we can obtain that our proposed algorithm can get a better predict accuracy than other classical regression algorithms, and our algorithm obtain the better result with the smaller δ, but this is not absolute.

6 Conclusion

We have designed a specialized pdf for the input of MGP model which consists of three parts: the right and left side parts still take the form of Gaussian distributions, while the middle part takes the form of a uniform distribution. This specialized pdf has the advantages of both the Gaussian distribution and the uniform distribution. That is, the tail Gaussian distributions in the left and right side parts ensure that the hard-cut EM algorithm can perform more efficiently during each iteration, and the uniform distribution in the middle part is more reasonable for the time series data. The experiments are conducted on three groups of synthetic datasets and stock datasets. It is demonstrated by the experimental results that the hard-cut EM algorithm for the MGPs with the specialized pdfs can obtain a better prediction accuracy than the other classical regression algorithms. This specialized input pdf is more effective for the time series data.

Acknowledgment. This work was supported by the National Science Foundation of China under Grant 61171138.

References

1. Rasmussen, C.E.: Evaluation of Gaussian Processes and Other Methods for Non-linear Regression. University of Toronto (1999)
2. Williams, C.K.I., Barber, D.: Bayesian classification with Gaussian processes. IEEE Trans. Pattern Anal. Mach. Intell. **20**(12), 1342–1351 (1998)
3. Rasmussen, C.E., Kuss, M.: Gaussian processes in reinforcement learning. In: NIPS, vol. 4, p. 1 (2003)
4. Tresp, V.: Mixtures of Gaussian processes. In: Advances in Neural Information Processing Systems, pp. 654–660 (2001)
5. Yuan, C., Neubauer, C.: Variational mixture of Gaussian process experts. In: Advances in Neural Information Processing Systems, pp. 1897–1904 (2009)

6. Stachniss, C., Plagemann, C., Lilienthal, A.J., et al.: Gas Distribution Modeling using Sparse Gaussian Process Mixture Models. In: Robotics: Science and Systems, vol. 3 (2008)
7. Yang, Y., Ma, J.: An efficient EM approach to parameter learning of the mixture of Gaussian processes. In: Liu, D., Zhang, H., Polycarpou, M., Alippi, C., He, H. (eds.) ISNN 2011. LNCS, vol. 6676, pp. 165–174. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21090-7_20
8. Meeds, E., Osindero, S.: An alternative infinite mixture of Gaussian process experts. In: Advances in Neural Information Processing Systems, pp. 883–890 (2006)
9. Sun, S., Xu, X.: Variational inference for infinite mixtures of Gaussian processes with applications to traffic flow prediction. IEEE Trans. Intell. Transp. Syst. **12**(2), 466–475 (2011)
10. Williams, C.K.I., Rasmussen, C.E.: Gaussian processes for machine learning, MIT Press **2**(3), 4 (2006)
11. Nguyen, T., Bonilla, E.: Fast allocation of Gaussian process experts. In: International Conference on Machine Learning, pp. 145–153 (2014)
12. Lázaro-Gredilla, M., Van Vaerenbergh, S., Lawrence, N.D.: Overlapping mixtures of Gaussian processes for the data association problem. Pattern Recogn. **45**(4), 1386–1395 (2012)
13. Ross, J., Dy, J.: Nonparametric mixture of Gaussian processes with constraints. In: International Conference on Machine Learning, 1346–1354 (2013)
14. 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)
15. Wu, D., Chen, Z., Ma, J.: An MCMC based EM algorithm for mixtures of Gaussian processes. In: Hu, X., Xia, Y., Zhang, Y., Zhao, D. (eds.) ISNN 2015. LNCS, vol. 9377, pp. 327–334. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25393-0_36
16. Wu, D., Ma, J.: A DAEM algorithm for mixtures of Gaussian process functional regressions. In: Huang, D.-S., Han, K., Hussain, A. (eds.) ICIC 2016. LNCS (LNAI), vol. 9773, pp. 294–303. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42297-8_28
17. Chen, Z., Ma, J., Zhou, Y.: A precise hard-cut EM algorithm for mixtures of Gaussian processes. In: Huang, D.-S., Jo, K.-H., Wang, L. (eds.) ICIC 2014. LNCS (LNAI), vol. 8589, pp. 68–75. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09339-0_7
18. Chen, Z., Ma, J.: The hard-cut EM algorithm for mixture of sparse Gaussian processes. In: Huang, D.-S., Han, K. (eds.) ICIC 2015. LNCS (LNAI), vol. 9227, pp. 13–24. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22053-6_2
19. Zhao, L., Chen, Z., Ma, J.: An effective model selection criterion for mixtures of Gaussian processes. In: Hu, X., Xia, Y., Zhang, Y., Zhao, D. (eds.) ISNN 2015. LNCS, vol. 9377, pp. 345–354. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25393-0_38
20. Zhao, L., Ma, J.: A dynamic model selection algorithm for mixtures of Gaussian processes. In: 2016 IEEE 13th International Conference on Signal Processing (ICSP), pp. 1095–1099. IEEE (2016)
21. Liu, S., Ma, J.: Stock price prediction through the mixture of Gaussian processes via the precise hard-cut EM algorithm. In: Huang, D.-S., Han, K., Hussain, A. (eds.) ICIC 2016. LNCS (LNAI), vol. 9773, pp. 282–293. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42297-8_27

22. Shi, J.Q., Murray-Smith, R., Titterington, D.M.: Bayesian regression and classification using mixtures of Gaussian processes. Int. J. Adapt. Control. Signal Process. **17**(2), 149–161 (2003)
23. Tayal, A., Poupart, P., Li, Y.: Hierarchical double Dirichlet process mixture of Gaussian processes. In: AAAI (2012)
24. Chatzis, S.P., Demiris, Y.: Nonparametric mixtures of Gaussian processes with power-law behavior. IEEE Trans. Neural Netw. Learn. Syst. **23**(12), 1862–1871 (2012)
25. Kapoor, A., Ahn, H., Picard, R.W.: Mixture of Gaussian processes for combining multiple modalities. In: Oza, N.C., Polikar, R., Kittler, J., Roli, F. (eds.) MCS 2005. LNCS, vol. 3541, pp. 86–96. Springer, Heidelberg (2005). https://doi.org/10.1007/11494683_9