The Hard-Cut EM Algorithm for Mixture of Sparse Gaussian Processes

Ziyi Chen and Jinwen Ma(✉)

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

Abstract. The mixture of Gaussian Processes (MGP) is a powerful and fast developed machine learning framework. In order to make its learning more efficient, certain sparsity constraints have been adopted to form the mixture of sparse Gaussian Processes (MSGP). However, the existing MGP and MSGP models are rather complicated and their learning algorithms involve various approximation schemes. In this paper, we refine the MSGP model and develop the hard-cut EM algorithm for MSGP from its original version for MGP. It is demonstrated by the experiments on both synthetic and real datasets that our refined MSGP model and the hard-cut EM algorithm are feasible and can outperform some typical regression algorithms on prediction. Moreover, with sparse technique, the parameter learning of our proposed MSGP model is much more efficient than that of the MGP model.

Keywords: Mixture of Gaussian Processes · Sparsity · Hard-cut EM algorithm · Big data

1 Introduction

Gaussian process (GP) is a powerful model for a wide range of applications in machine learning, including regression [1], classification [2], dimensionality reduction [3], reinforcement learning [4], etc. Nevertheless, GP model fails to describe multimodality dataset and the training of GP consumes $O(N^3)$ time for N training samples [5, 6]. In order to solve these problems, Tresp [7] proposed the Mixture of Gaussian Processes (MGP) in 2000, which was adjusted from Mixture of Experts. Since then, various MGP models have been proposed, and the most of them are special cases of mixture of experts where each expert is a GP.

Another useful way of reducing the time cost of training a GP is to adopt the model of sparse Gaussian Process (SGP), which computes the GP likelihood with a pseudo dataset being smaller than the training dataset in size [8]. To combine the advantages of MGP and SGP, the Mixture of Sparse Gaussian Processes (MSGP) has been also proposed, in which each component is a SGP instead of a GP to accelerate the parameter learning [5, 6, 9–11].

For these MGP and MSGP models, there are three main training algorithms: Monte Carlo Markov Chain (MCMC), variational Bayesian inference (VB), and EM algorithm. Actually, MCMC approximated the posterior of parameters by sampling several

D.-S. Huang and K. Han (Eds.): ICIC 2015, Part III, LNAI 9227, pp. 13–24, 2015.
DOI: 10.1007/978-3-319-22053-6_2

long sequences of Markov Chains [12–19]. However, it was quite time consuming as pointed out in [9]. VB tried to approximate the posterior of parameters with factorized forms, which actually may deviate a lot from the true posterior [20–24].

Generally, EM algorithm is an effective approach to the learning of mixture models. However, since the exact posterior of latent variables and the Q function are intractable for MGP and MSGP models, several approximation strategies have been adopted. For example, variational EM algorithm approximated the posterior in E step with variational inference [5, 9–11, 25], which had the same drawbacks as VB. In [26, 27], the Q function was decomposed via leave-one-out cross-validation (LOOCV), which was a complicated form involving much computation. Stachniss et al. [6] and Tresp [7] attempted to simplify the learning process with the help of heuristic estimation in M step. However, kernel parameters were predetermined without learning in [7], whereas the parameters were sampled from data-irrelevant distributions in [6]. Therefore, both learning processes in fact needed more guidance by datasets.

Recently, some hard-cut EM algorithms have also been adopted to improve the learning efficiency, which partition each sample into the component with the maximum posterior in E-step and learn the parameters of each component independently in M-step [9, 28]. Moreover, in the same way, Yang and Ma [26] has adjusted its proposed soft EM algorithm for MGP with the LOOCV probability decomposition into a hard-cut EM algorithm for MGP, which is here referred to as the LOOCV hard-cut EM algorithm for convenience.

After all, these algorithms resorted to some approximation strategies since MGP and MSGP models are usually very complex. Recently, Chen et al. [29] refined the MGP model to a more simplified form, and strictly derived a precise hard-cut EM algorithm for it. In this paper, we develop this approach by adding sparsity constraints and adjusting the refined MGP model into the MSGP model. As a result, the parameter learning becomes more efficient as the MSGP model is refined and the hard-cut EM algorithm is still strictly derived. The experimental results demonstrates that our proposed model and algorithm are feasible and can outperform some typical regression algorithms.

2 The Mixture of Sparse Gaussian Processes

2.1 Gaussian Process (GP)

A GP model for regression is mathematically defined by

$$\begin{cases} F = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix} \sim N\left(\begin{bmatrix} m(x_1) \\ m(x_2) \\ \vdots \\ m(x_N) \end{bmatrix}, \begin{bmatrix} K(x_1,x_1) & K(x_1,x_2) & \cdots & K(x_1,x_N) \\ K(x_2,x_1) & K(x_2,x_2) & \cdots & K(x_2,x_N) \\ \vdots & \vdots & \ddots & \vdots \\ K(x_N,x_1) & K(x_N,x_2) & \cdots & K(x_N,x_N) \end{bmatrix} \right), \\ y_t \sim N(f_t, \sigma^2) \end{cases} \quad (1)$$

where x_t, f_t and y_t denote the input, latent response and output of a training sample, respectively, K(u, v) is a mercer kernel function, and σ^2 denotes the noise intensity. As in most cases, we adopt zero mean function (m ≡ 0) and the most popular kernel function—the squared exponential (SE) kernel [30]:

$$K(u,v) = l^2 \exp\left[-\frac{1}{2}\sum_{k=1}^{d} b_k^2 (u_k - v_k)^2\right], \tag{2}$$

where d is the dimensionality of inputs and each dimension has a different weight b_k to realize automatic feature selection.

There are several ways for learning the hyper-parameters of a GP model, including the approaches of maximum likelihood estimation (MLE), maximizing a posteriori (MAP), surrogate predictive probability (GPP), cross validation (CV), etc. [31].

With the estimated hyper-parameters, the predictive distribution of the output at a test input x* can be obtained as follows:

$$p(y^*|x^*) \sim N\left[K(x^*,X)K(X,X)^{-1}y, K(x^*,x^*) - K(x^*,X)K(X,X)^{-1}K(X,x^*)\right], \tag{3}$$

where $K(x^*, X) = [K(x^*, x_j)]_{1xN}$, $K(X, x^*) = [K(x_i, x^*)]_{Nx1}$, $K(X, X) = [K(x_i, x_j)]_{NxN}$ are kernel matrices [30].

2.2 Mixture of Gaussian Processes (MGP)

MGP is a special mixture model in which each component is just a Gaussian process, and these components are independent in most existing MGP models. Denote $z_t = c$ iff (x_t, y_t) belongs to the c-th GP, and denote

$$X_c = \left[x_{t_c(1)}, x_{t_c(2)}, \cdots, x_{t_c(N_c)}\right]^T \quad Y_c = \left[y_{t_c(1)}, y_{t_c(2)}, \cdots, y_{t_c(N_c)}\right]^T$$

as the inputs and outputs from the c-th GP, respectively, where N_c is the number of samples in the c-th GP. Therefore, the output likelihood is mathematically given by

$$Y_c \sim N\left[0, K(X_c, X_c|\theta_c) + \sigma_c^2 I\right]; c = 1, 2, \cdots, C, \tag{4}$$

where the SE kernel given by Eq. (2) is parameterized by $\theta_c = \{l_c, \sigma_c\} \cup \{b_{ck} : k = 1, 2, \cdots, d\}$ for the c-th GP.

MGP has two main advantages over a single GP. Firstly, MGP can capture the heterogeneity among different input locations by fitting them with different GPs. For example, the toy dataset used by some MGP models [12, 17, 26, 29] was generated by 4 continuous functions with Gaussian noise, as is shown in Fig. 1. Therefore, it is better to fit the Toy dataset by MGP with four GPs than only one GP. Secondly, the computational cost can be reduced by dividing the large kernel matrix of one GP into small matrices of components.

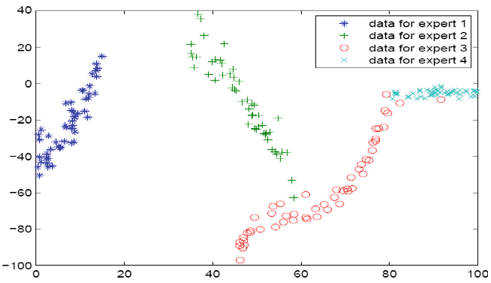

Fig. 1. Toy dataset

2.3 Sparse Gaussian Process (SGP)

Another good scaling technique for GP is to use the model of Sparse Gaussian Process. The main idea of SGP is to approximate N training samples with M pseudo samples (M<<N) [32]. Mathematically, we denote inputs of the training data, test data and pseudo data $X = [x_1, x_2, \cdots, x_N]^T$, $X^* = [x_1^*, x_2^*, \cdots, x_L^*]^T$, $U = [u_1, u_2, \cdots, u_M]^T$ respectively, and their corresponding latent responses are denoted as $F = [f_1, f_2, \cdots, f_N]^T$, $F^* = [f_1^*, f_2^*, \cdots, f_L^*]^T$ and $V = [v_1, v_2, \cdots, v_M]^T$, respectively. Almost, the sparse GP models can be mathematically defined by the following equations [32]:

$$V \sim N[0, K(U,U)], \tag{5}$$

$$p(F, F^*|V) = q(F|V)q(F^*|V), \tag{6}$$

$$y_t|f_t \sim N(f_t, \sigma^2)\ i.i.d., \tag{7}$$

$$y_t^*|f_t^* \sim N(f_t^*, \sigma^2)\ i.i.d., \tag{8}$$

where Eq. (5) means the latent pseudo outputs V fulfill a GP, as in Eqs. (1), and (6) gives the crucial assumption of SGP, i.e., the latent responses for training dataset and test dataset are conditionally independent given V. In Eq. (6), q(F|V) and q(F*|V) can be approximations of the following GP predictive distributions

$$\begin{aligned} p(F^*|V) &\sim N\left[K(X^*,U)^T K(U,U)^{-1}V, K(X^*,X^*) - K(X^*,U)K(U,U)^{-1}K(U,X^*)\right] \\ p(F|V) &\sim N\left[K(X,U)^T K(U,U)^{-1}V, K(X,X) - K(X,U)K(U,U)^{-1}K(U,X)\right], \end{aligned} \tag{9}$$

respectively, and the forms of q(F|V) and q(F*|V) determine the kind of SGP models, including SoR, DTC, FITC and PITC models [32]. Among these models, the Fully Independent Training Conditional (FITC) model proposed in [8] was highly recommended in [32], due to its rich covariance. Experimental results have shown the great advantage of FITC over the other sparse GPs on predictive accuracy, whereas the

increase on time consumption is trivial [8]. Therefore, we will use FITC as each component for our proposed mixture of sparse Gaussian Processes model (MSGP).

In FITC model, the conditional distribution of the latent test outputs is the exact GP predictive distribution:

$$\begin{aligned} &q(F^*|V) = p(F^*|V) \\ &\sim N\Big[K(X^*,U)^T K(U,U)^{-1}V, K(X^*,X^*) - K(X^*,U)K(U,U)^{-1}K(U,X^*)\Big], \end{aligned} \tag{10}$$

whereas the latent training outputs are assumed to be conditionally independent given the latent pseudo output V, as suggested by the name "Fully Independent Training Conditional", i.e.

$$q(F|V) \sim N\Big[K(X,U)K(U,U)^{-1}V, \Lambda\Big], \tag{11}$$

where $\Lambda = diag\Big[K(X,X) - K(X,U)K(U,U)^{-1}K(U,X)\Big]$

It can be inferred that the marginal likelihood for the outputs is

$$p(Y) \sim N\Big[0, K(X,U)K(U,U)^{-1}K(U,X) + \Lambda + \sigma^2 I\Big], \tag{12}$$

and the predictive distribution can be derived as follows.

$$\begin{aligned} p(Y^*|Y) \sim N\Big\{&K(X^*,U)Q^{-1}K(U,X^*)(\Lambda + \sigma^2 I)^{-1}Y, \\ &K(X^*,X^*) - K(X^*,U)[K(U,U)^{-1} - Q^{-1}]K(U,X^*) + \sigma^2 I\Big\}, \end{aligned} \tag{13}$$

where $Q = K(U,U) + K(U,X)(\Lambda + \sigma^2 I)^{-1}K(X,U)$.

The crucial issue on training a SGP is to find appropriate pseudo inputs. One method is to greedily select a fixed number of training inputs one by one with a certain criterion [33–36]. Snelson and Ghahramani [32] have extended the range of pseudo inputs from the training sets to the whole Euclidean space, and learnt pseudo inputs and the hyper parameters together by maximizing the likelihood, which is more flexible than greedy selection and has better performance in experiments.

2.4 The FITC Mixture Model

As the MGP model proposed in [29] is refined and can be trained by the precise learning algorithm, we can inherit its general framework for our MSGP model. The only difference is that SGP is adopted for each component for higher speed in the training process.

As in [29], we adopt the following gating function and Gaussian inputs.

$$\Pr(z_t = c) = \pi_c; c = 1 \sim C \ i.i.d \ for \ t = 1 \sim N, \tag{14}$$

$$p(x_t|z_t = c) \sim N(\mu_c, S_c); c = 1 \sim C, \ \ i.i.d \ for \ \ t = 1 \sim N. \tag{15}$$

Furthermore, for each component, we use the FITC model due to its advantages mentioned in Sect. 2.3 to describe the SGP input-output relation

$$p(Y_c|X_c, U_c, \theta_c) \sim N\Big[0, K(X_c, U_c|\theta_c)K(U_c, U_c|\theta_c)^{-1}K(U_c, X_c|\theta_c) + \Lambda_c + \sigma_c^2 I\Big], \tag{16}$$

where U_c denotes the pseudo inputs in the c-th component,

$$\Lambda_c = diag\Big[K(X_c, X_c|\theta_c) - K(X_c, U_c|\theta_c)K(U_c, U_c|\theta_c)^{-1}K(U_c, X_c|\theta_c)\Big], \tag{17}$$

and X_c, Y_c, θ_c follow the meanings in Sect. 2.2.

Our MSGP model can be fully described by Eqs. (14)–(16), and it has fewer parameters than previous MSGP models [5, 6, 9–11]. However, it still retains the advantages of combining sparse GPs with the mixture model. For this refined model, we strictly derive its training algorithm, called the precise hard-cut EM algorithm.

3 The Hard-Cut EM Algorithm for the FITC Mixtures

Since the outputs from both MGP and MSGP models are dependent, the computational complexity of Q function is exponential with summation over multiple labels. In order to avoid such an expensive computation, a hard-cut EM algorithm can be adopted by partitioning the training samples into the corresponding component with the maximum posterior in E-step, so that the parameters of each component can be learnt separately via the MLE in M-step. Because our model is developed from the MGP model in [29], where the precise hard-cut EM algorithm was strictly derived and performed quite well on both the prediction accuracy and the learning efficiency, we adjust its general learning framework to train the FITC mixture.

It can be easily derived that the marginal output likelihood for each sample in fact remains exactly the same as [29], i.e.

$$p(y_t|x_t, z_t = c) = N\big[y_t|0, K(x_t, x_t|\theta_c) + \sigma_c^2\big] = N(y_t|0, l_c^2 + \sigma_c^2), \tag{18}$$

so, the same posterior in E-step can be derived using the Bayesian formula:

$$\Pr(z_t = c|x_t, y_t) = \frac{\pi_c N(x_t|\mu_c, S_c)N(y_t|l_c^2 + \sigma_c^2)}{\sum_{k=1}^{C} \pi_k N(x_t|\mu_k, S_k)N(y_t|l_k^2 + \sigma_k^2)}. \tag{19}$$

However, in M-step, the GP likelihood in [29] can be replaced by the FITC likelihood given by Eq. (16) with less computation, thus pseudo inputs are added during

the learning of the hyper-parameters. With such an adjustment from [29], we obtain the procedure of our proposed precise hard-cut EM algorithm as follows.

- Step1 (Initialize the partition of the training data). Divide the vectors $\{[x_t^T, y_t]\}_{t=1}^N$ into C clusters, and set $z_t \leftarrow$the indicator of the t-th sample to the cluster.
- Step2 (M-step). Learn the parameters with MLE for each component:

$$\pi_c \leftarrow \frac{N_c}{N}, \quad \mu_c \leftarrow \frac{1}{N_c}\sum_{t=1}^{N} I(z_t = c)x_t, \quad S_c \leftarrow \frac{1}{N_c}\sum_{t=1}^{N} I(z_t = c)(x_t - \mu_c)(x_t - \mu_c)^T, \tag{20}$$

$$(\hat{U}_c, \hat{\theta}_c) = \arg\max_{U_c, \theta_c} \ln p(Y_c | X_c, U_c, \theta_c), \tag{21}$$

where N_c is the number of samples in the c-th component, Eq. (20) is the same as learning the parameters of the Gaussian mixture model and Eq. (21) follows from [8].

- Step3 (E-step). Repartition the samples into the corresponding component based on the maximizing a posterior criterion:

$$z_t \leftarrow \arg\max_c \Pr(z_t = c | x_t, y_t) = \arg\max_c [\pi_c N(x_t | \mu_c, S_c) N(y_t | l_c^2 + \sigma_c^2)] \tag{22}$$

- Step4. Stop if the number of changed labels falls below η % of the number of training samples ($0 \le \eta \le 10$ is a threshold). Otherwise, return to Step 2.

After the learning process above, we have obtained the estimated parameters and the partition of the training data. Then, for a test input x*, we can classify it into the z-th component of the MSGP by maximizing the posterior as in [29]:

$$z^* \leftarrow \arg\max_c \Pr(z^* = c | D, x^*) = \arg\max_c [\pi_c N(x^* | \mu_c, S_c)] \tag{23}$$

According to this classification, we can predict the output of the test input via the z-th component using Eq. (13).

4 Experimental Results

4.1 On a Small Synthetic Dataset from the MGP Model

In order to evaluate the validity and feasibility of the FITC mixture model and its hard-cut EM algorithm, we generate a typical synthetic dataset from MGP model with 3 components. Actually, there are 500 training samples and 100 test samples in each component and each input has 3 features. Then we compare our algorithms with the following typical baselines for regression:

1. The FITC model and its MLE algorithm [8].
2. The MGP model and its precise hard-cut EM algorithm [29].
3. The MGP model and its LOOCV hard-cut EM algorithm [26].
4. The mixture of sparse GPs model and its variational hard-cut EM algorithm [9].
5. Linear regression [37].
6. SVM regression with Gaussian kernel [38].

Each of these algorithms is implemented on this synthetic dataset five times, with an Intel (R) Core (TM) i5 CPU and 4.00 GB of RAM running Matlab R2014b source codes on Secure CRT. For each of these algorithms, the mean and standard deviation of the root mean squared errors (RMSEs) for prediction and the training times are listed in Table 1. In addition, to make the prediction of FITC and its mixture model more accurate, we initialize the kernel parameters by training a GP model on 20 randomly selected training samples before the MLE learning process, as did in [8].

Table 1. The mean and standard deviation of the predictive RMSEs and the training times for each algorithm on the typical synthetic dataset from MGP model

	RMSE	Training time (s)
Our proposed precise hard-cut EM algorithm for FITC mixture (C = 3 components, M = 20 pseudo inputs, threshold η % = 5 %)	0.5139 ± 0.0222	27.0198 ± 3.9183
MLE for FITC model (M = 20)	0.6495 ± 0.0994	22.2497 ± 1.5894
Precise hard-cut EM algorithm for MGP (C = 3)	0.4963 ± 0	190.4125 ± 42.4939
LOOCV hard-cut EM algorithm for MGP (C = 3)	0.4977 ± 0	2552.0 ± 559.5477
Variational hard-cut EM algorithm (C = 3, M = 20)	0.6086 ± 0.1634	93.4882 ± 11.3112
Linear regression	0.8514 ± 0	0.0571 ± 0.1273
SVM	0.7109 ± 0	221.1791 ± 5.5132

It can be seen from Table 1 that on the synthetic dataset, our proposed algorithm is much more accurate than the single FITC since the dataset is multimodal. With sparse technique, our algorithm is much more efficient than the two EM algorithms of the MGP models, whereas the loss of accuracy is trivial compared with the increase on speed. The variational hard-cut EM algorithm makes a coarse conditionally independent assumption to the posterior and thus it is not as accurate as our proposed algorithm. Besides, the variational hard-cut EM algorithm also takes longer to train than our proposed algorithm. Traditional regression algorithms like the linear regression and the SVM are rough for such a non-linear synthetic dataset.

Furthermore, our proposed algorithm correctly partitions all the training samples and test samples into their components each time, which, along with the results in Table 1, means that our algorithm is feasible and effective.

4.2 On a Large Synthetic Dataset from the FITC Mixture Model

Moreover, to test the advantage of FITC mixture model and its hard-cut EM algorithm, we generate a typical synthetic dataset from FITC mixture model with 100 components. In each component, there are 200 training samples and 200 test samples, with 1 dimensional inputs. Then we compare our algorithm with the baselines above. The computational environment and experimental details remain the same as above.

On such a big dataset (20000 training samples), some algorithms are prohibitively slow and fail to converge, such as the SVM, the precise hard-cut EM algorithm and the LOOCV hard-cut EM algorithm of MGP, so we omit them in the experiment. It also indicates that our proposed hard-cut EM algorithm is more efficient than the precise hard-cut EM algorithm of the MGP model due to sparsity mechanism.

All the other algorithms are implemented on this synthetic dataset five times and the mean and standard deviation of their RMSEs for prediction and training times are listed in Table 2. Similarly, for FITC model and our proposed FITC mixture model, we initialize the kernel parameters by training a GP model on 10 randomly selected training samples before the MLE learning process, as in [8].

Table 2. The mean and standard deviation of the predictive RMSEs and the training times for each algorithm on the typical synthetic dataset from FITC mixture model

	RMSE	Training time (s)
Our proposed precise hard-cut EM algorithm for FITC mixture (C = 100, M = 10, η % = 5 %)	0.1115 ± 0.0106	86.9691 ± 2.8167
MLE for FITC model (M = 10)	1.0149 ± 0.0190	25.3531 ± 12.8932
Variational hard-cut EM algorithm (C = 100, M = 10)	0.8649 ± 0.0339	890.1411 ± 159.3509
Linear regression	1.0492 ± 0	0.0006 ± 0.0001

Table 2 indicates that for the synthetic dataset with much more components, our proposed FITC mixture model has greater advantage than the FITC model, on both predictive precision and learning efficiency. The conditionally independent assumption of the variational hard-cut EM algorithm, as well as the linear assumption of the linear regression, is not suitable for such a highly non-linear dataset with dependent samples. Therefore, the two algorithms have big prediction errors on this dataset.

4.3 On Kin40k Dataset

Finally, we compare these algorithms on a popular real dataset called kin40k, which is generated by a robot arm simulator, with 10000 training samples, 30000 test samples and 9 attributes [34]. The computational environment and implementation details remain the same as above. The mean and standard deviation of the predictive RMSEs as well as the training times for each algorithm are listed in Table 3. Similarly, for FITC and FITC mixture model, we initialize the kernel parameters by training a GP model on 500 randomly selected training samples before the MLE learning process.

Table 3. The mean and standard deviation of the predictive RMSEs and the training times for each algorithm on kin40k dataset

	RMSE	Training time (s)
Our proposed precise hard-cut EM algorithm for FITC mixture (C = 4, M = 650, η % = 5 %)	0.2007 ± 0.0094	14358.4108 ± 5321.6807
MLE for FITC model (M = 650)	0.2823 ± 0.0033	1547.4120 ± 66.1614
Variational hard-cut EM algorithm for MGP (C = 4, M = 650)	0.2475 ± 0.0269	2627.1 ± 84.1696
Linear regression	1.0492 ± 0	0.0006 ± 0.0001

Still, on the large dataset, the SVM, the precise hard-cut EM algorithm and the LOOCV hard-cut EM algorithm of MGP are prohibitively slow and fail to converge.

From Table 3, we can observe that on the kin40 k dataset, our proposed algorithm is more precise than the other algorithms due to fewer approximations. A possible reason why our algorithm consumes so much time is that on a real dataset that doesn't come from a GP or MGP model, our algorithm needs much more iterations. Therefore, a further improvement can be made by preprocessing a real dataset so that it is more like a dataset simulated from a MGP. Linear regression is still rather coarse since the kin40k dataset is highly non-linear [34].

In general, our algorithm is practical on the real dataset, and its computational cost is acceptable with 10000 training samples.

5 Conclusion

We have refined the MSGP model and developed the hard-cut EM algorithm for its parameter learning. The general framework is adapted from [29] whereas the learning efficiency significantly improves with the sparsity mechanism. The experimental results on both the synthetic dataset and the real dataset demonstrate that the developed hard-cut EM algorithm for MSGP model is precise and efficient, and generally outperforms the conventional linear regression, SVM and some other learning algorithms for GP, MGP and MSGP models.

Acknowledgement. This work was supported by the Natural Science Foundation of China for Grant 61171138. The authors would like to thank Dr. E. Snelson and Dr. Z. Ghahramani for their valuable advice about FITC model.

References

1. Rasmussen, C.E.: Evaluation of Gaussian processes and other methods for non-linear regression. The University of Toronto (1996)
2. Williiams, C.K.I., Barber, D.: Bayesian classification with Gaussian processes. IEEE Trans. Pattern Anal. Mach. Intell. **20**(12), 1342–1351 (1998)

3. Gao, X.B., Wang, X.M., Tao, D.C.: Supervised Gaussian process latent variable model for dimensionality reduction. IEEE Trans. Syst. Man Cybern. B Cyberne. **41**(2), 425–434 (2011)
4. Rasmussen, C.E., Kuss, M.: Gaussian processes in reinforcement learning. In: Thrun, S., Saul, L., Schölkopf, B. (eds.) Advances in Neural Information Processing Systems, vol. 16, pp. 751–759. MIT Press, Cambridge (2003)
5. Yuan, C., Neubauer, C.: Variational mixture of Gaussian process experts. In: Advances in Neural Information Processing Systems, vol. 21, pp. 1897–1904 (2008)
6. Stachniss, C., Plagemann, C., Lilienthal, A.J., et al.: Gas distribution modeling using sparse Gaussian process mixture models. In: Proceedings of Robotics: Science and Systems, pp. 310–317 (2008)
7. Tresp, V.: Mixtures of Gaussian processes. In: Advances in Neural Information Processing Systems, vol. 13, pp. 654–660 (2000)
8. Snelson, E., Ghahramani, Z.: Sparse Gaussian processes using pseudo-inputs. In: Advances in Neural Information Processing Systems, vol. 18, pp. 1257–1264 (2005)
9. Nguyen, T., Bonilla, E.: Fast allocation of Gaussian process experts. In: Proceedings of the 31st International Conference on Machine Learning, pp. 145–153 (2014)
10. Wang, Y., Khardon, R.: Sparse Gaussian processes for multi-task learning. In: The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, pp. 711–727 (2012)
11. 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)
12. Meeds, E., Osindero, S.: An alternative infinite mixture of Gaussian process experts. In: Advances in Neural Information Processing Systems, vol. 18, pp. 883–890 (2005)
13. Gramacy, R.B., Lee, H.K.H.: Bayesian treed Gaussian process models with an application to computer modeling. J. Am. Stat. Assoc. **103**(483), 1119–1130 (2008)
14. Shi, J.Q., Murray-Smith, R., Titterington, D.M.: Bayesian regression and classification using mixtures of Gaussian processes. Int. J. Adapt. Control Sig. Process. **17**(2), 149–161 (2003)
15. Shi, J.Q., Murray-Smith, R., Titterington, D.M.: Hierarchical Gaussian process mixtures for regression. Stat. Comput. **15**(1), 31–41 (2005)
16. Rasmussen, C.E., Ghahramani, Z.: Infinite mixtures of Gaussian process experts. In: Advances in Neural Information Processing Systems, vol. 14, pp. 881–888 (2001)
17. Fergie, M.P.: Discriminative Pose Estimation Using Mixtures of Gaussian Processes. The University of Manchester (2013)
18. Sun, S.: Infinite mixtures of multivariate Gaussian processes. In: Proceedings of the International Conference on Machine Learning and Cybernetics, pp. 1011–1016 (2013)
19. Tayal, A., Poupart, P., Li, Y.: Hierarchical double Dirichlet process mixture of Gaussian processes. In: Proceedings of the 26th Association for the Advancement of Artificial Intelligence, pp. 1126–1133 (2012)
20. Ross, J., Dy, J.: Nonparametric mixture of Gaussian processes with constraints. In: Proceedings of the 30th International Conference on Machine Learning, pp. 1346–1354 (2013)
21. 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)
22. Platanios, E.A., Chatzis, S.P.: Mixture Gaussian process conditional heteroscedasticity. IEEE Trans. Pattern Anal. Mach. Intell. **36**(5), 888–900 (2014)
23. 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)

24. Reece, S., Mann, R., Rezek, I., et al.: Gaussian process segmentation of co-moving animals. Proc. Am. Inst. Phys. **1305**(1), 430–437 (2011)
25. Lázaro-Gredilla, M., Van, V.S., Lawrence, N.D.: Overlapping mixtures of Gaussian processes for the data association problem. Pattern Recogn. **45**(4), 1386–1395 (2012)
26. 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, Part II. LNCS, vol. 6676, pp. 165–174. Springer, Heidelberg (2011)
27. Schiegg, M., Neumann, M., Kersting, K.: Markov logic mixtures of Gaussian processes: towards machines reading regression data. In: Proceedings of the 15th International Conference on Artificial Intelligence and Statistics, JMLR:W&CP, vol. 22, pp. 1002–1011 (2012)
28. Yu, J., Chen, K., Rashid, M.M.: A Bayesian model averaging based multi-kernel Gaussian process regression framework for nonlinear state estimation and quality prediction of multiphase batch processes with transient dynamics and uncertainty. Chem. Eng. Sci. **93** (19), 96–109 (2013)
29. 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, vol. 8589, pp. 68–75. Springer, Heidelberg (2014)
30. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)
31. Sundararajan, S., Keerthi, S.: Predictive approaches for choosing hyperparameters in Gaussian processes. Neural Comput. **13**(5), 1103–1118 (2001)
32. Quiñonero-Candela, J., Rasmussen, C.E.: A unifying view of sparse approximate Gaussian process regression. J. Mach. Learn. Res. **6**, 1935–1959 (2005)
33. Csató, L., Opper, M.: Sparse online Gaussian processes. Neural Comput. **14**(3), 641–669 (2002)
34. Seeger, M., Williams, C.K.I., Lawrence, N.D.: Fast forward selection to speed up sparse Gaussian process regression. In: Bishop, C.M., Frey, B.J. (eds.) Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics (2003)
35. Keerthi, S.S., Chu, W.: A matching pursuit approach to sparse Gaussian process regression. In: Advances in Neural Information Processing Systems, vol. 18 (2005)
36. Smola, A., Bartlett, P.: Sparse greedy Gaussian process regression. Adv. Neural Inf. Process. Syst. **13**, 619–625 (2000)
37. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. MIT Press, Cambridge (2012)
38. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Stat. Comput. **14**(3), 199–222 (2004)

http://www.springer.com/978-3-319-22052-9

Advanced Intelligent Computing Theories and Applications
11th International Conference, ICIC 2015, Fuzhou, China, August 20-23, 2015. Proceedings, Part III
Huang, D.-S.; Han, K. (Eds.)
2015, XXII, 797 p. 235 illus., Softcover
ISBN: 978-3-319-22052-9