Mach Learn (2018) 107:1947–1986
https://doi.org/10.1007/s10994-018-5721-5

Stochastic variational hierarchical mixture of sparse Gaussian processes for regression

Thi Nhat Anh Nguyen[1] · **Abdesselam Bouzerdoum**[1,2] · **Son Lam Phung**[1]

Received: 1 April 2017 / Accepted: 26 May 2018 / Published online: 6 July 2018
© The Author(s) 2018

Abstract In this article, we propose a scalable Gaussian process (GP) regression method that combines the advantages of both global and local GP approximations through a two-layer hierarchical model using a variational inference framework. The upper layer consists of a global sparse GP to coarsely model the entire data set, whereas the lower layer comprises a mixture of sparse GP experts which exploit local information to learn a fine-grained model. A two-step variational inference algorithm is developed to learn the global GP, the GP experts and the gating network simultaneously. Stochastic optimization can be employed to allow the application of the model to large-scale problems. Experiments on a wide range of benchmark data sets demonstrate the flexibility, scalability and predictive power of the proposed method.

Keywords Gaussian processes · Variational inference · Hierarchical structure · Graphical model

1 Introduction

Gaussian process models have become the dominant approach to nonparametric Bayesian regression (O'Hagan and Kingman 1978; Williams and Rasmussen 1996; Boyle and Frean 2005; Goldberg et al. 1998; Gramacy et al. 2004; Sollich and Williams 2005). However, GP

Editors: Lee Wee Sun and Robert Durrant.

✉ Thi Nhat Anh Nguyen
tnan287@uowmail.edu.au

Abdesselam Bouzerdoum
bouzer@uow.edu.au

Son Lam Phung
phung@uow.edu.au

[1] School of Electrical, Computer and Telecommunication Engineering, University of Wollongong, Wollongong, Australia

[2] College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar

models in general suffer from high computational complexity, which is $O(N^3)$ in training time and and $O(N^2)$ in memory for N training data points. These computation costs arise mainly from the inversion and storage of the covariance matrix. The unfavorable complexity prevents the application of GP regression to large-scale data sets.

There has been much interest in sparse approximation methods for GPs to overcome the limitation of high computational cost (Seeger et al. 2003; Lawrence et al. 2003; Smola and Bartlett 2001; Seeger 2003; Csató 2002; Tresp 2000a; Snelson and Ghahramani 2006). A comprehensive review of many popular sparse approximation methods can be found in Quiñonero-Candela and Rasmussen (2005). In these methods, the entire training set is approximated using a small set of inducing points. The covariance matrix among all data points is thereby approximated by a low-rank one. In this way, a lower complexity of $O(NM^2)$ in training time and $O(NM)$ in memory is achieved, where M is the number of inducing points. However, even these reduced storage methods are prohibitive for big data that contain millions or billions of samples.

There are generally two approaches to scale up sparse GP models to be able to handle big data. One approach is to spread computation across many nodes in a distributed system (Hoang et al. 2016; Gal et al. 2014). This approach often requires abundant computational resources (processors and memory) though. Another approach is to learn sparse GP models in stochastic fashion, where a mini-batch of data is used at each optimization iteration. Examples for this approach are Hensman et al. (2013) and Hoang et al. (2015), in which stochastic variational inference (Hoffman et al. 2013) is employed for model learning. This approach allows the application of sparse GP regression to large-scale problems even with limited available resources.

The sparse GPs normally work well for simple data sets. However, in complex data sets, the dependencies among the observations cannot be well-captured by a small number of inducing points. In addition, a single GP accompanied by a small set of global inducing points cannot account for the non-stationarity and locality in such data sets, as argued in Rasmussen and Ghahramani (2002). Mixture of Gaussian processes is another approach to reduce the computational complexity of GPs as presented in Rasmussen and Ghahramani (2002), Tresp (2000b), Shi et al. (2003, 2005), Meeds and Osindero (2006), Yuan and Neubauer (2009), Nguyen and Bonilla (2014). In the mixture of GPs approach, a gating network divides the input space into regions within which a specific GP expert is responsible for making predictions. In this way, the computational complexity is reduced since the storage and inversion of a large covariance matrix are replaced by those of multiple smaller matrices. The non-stationarity and locality in the data can also be naturally addressed.

Mixtures of GPs have two main limitations. The first limitation is the complexity of the inference problem, which usually involves simultaneous learning of both the experts and the gating network. Therefore, approximation techniques are often required for the inference. Many existing mixtures of GPs, such as those in Rasmussen and Ghahramani (2002), Meeds and Osindero (2006), Shi et al. (2003, 2005), resort to the intensive Markov chain Monte Carlo (MCMC) sampling methods, which can be very slow, especially for large-scale data sets. As a result, the limited scalability prohibits their application to even moderate-sized problems. Recently, several variational mixtures of GP experts have been proposed for GP regression using variational inference, which is a more flexible and faster alternative to MCMC sampling (Yuan and Neubauer 2009; Sun and Xu 2011; Nguyen and Bonilla 2014). However, there is still no clear way to apply stochastic optimization to variational mixtures of GPs to enable their application to big data. To the best of our knowledge, the largest experiments using the existing variational GP mixtures have been performed in Nguyen and Bonilla (2014) and Nguyen et al. (2016) with 100,000 data points. The second limitation of mixtures of GPs

is that each expert is independently trained using only the local data assigned to it, without taking into account the global information, i.e. the correlations between clusters. The trained experts are therefore likely to overfit the local training data.

In this paper, we propose a GP approximation method for regression that combines the advantages of sparse approximation and mixture of GPs in a variational inference framework to exploit both the global and local information from the training data. Our model has a two-layer hierarchical structure. The upper layer uses a sparse GP accompanied by a set of global inducing points to coarsely model the entire data set. The lower layer comprises a mixture of GP experts, each of which is also a sparse GP. These experts make use of the local information from the corresponding data points assigned to them for fine-grained modeling. The experts share a common prior mean function which is the latent function modeled by the upper layer in order to enforce correlation among themselves. This way, overfitting is avoided. For inference, we develop a two-step variational inference algorithm for simultaneous learning of the global sparse GP, the experts and the gating network. We also derive an objective function that appears in a factorized form necessary for stochastic optimization, thereby enabling the application of the model to large-scale data sets.

Next, we briefly highlight the advantages of the proposed model with respect to the two previous attempts in combining local and global information for GP approximation presented in Snelson and Ghahramani (2007) and Park and Choi (2010). In both of the above models, the input space is first partitioned into several clusters. The data set is then approximated based on this clustered structure. In the partially independent conditional (PIC) method (Snelson and Ghahramani 2007), the covariances within a cluster are calculated exactly, while the covariances between points belonging to different clusters are approximated using a set of inducing points. Since PIC uses this approximate covariance matrix to train a single GP, it has limited capability to model the non-stationarity in large complex data sets. Park and Choi (2010) models each cluster by a local GP. However, the relationships among these local GPs are only loosely presented using the prototype variables (one for each cluster), which are placed at the cluster centers and share a joint GP prior. A common drawback of the two above models is that the partitioning and inference are completely separated. Hence, they must rely on independent clustering methods such as k-means for local GP allocation, as well as for placing the prototype variables in the case of Park and Choi (2010). These methods might not provide the optimal partitioning for the inference of the GPs. For example, the generated partitions might not reflect different noise levels or different length-scales across the data set. In the proposed model, inference and the gating network are learned simultaneously in a common variational framework so that the results from inference can improve the clustering and vice versa. Another drawback of the two models discussed above is that their computational complexity depends on the size of the clusters. Therefore, it places a limit on the maximum size of the clusters given the time and memory limitations of the test computer. In contrast, the complexity of the proposed model is independent of the size of the clusters because the local GPs are also sparse.

For the experiments and validation, we consider three sets of experiments with data sets of varying size to investigate different aspects of the proposed model. In the first set of experiments, we visually investigate the model on two small-size data sets with input-dependent noise. The result shows that the proposed method is able to both detect the common trend and handle the non-stationarity in the data sets at the same time. In the second set of experiments, we evaluate the predictive performance of the proposed model and compare it with four other baseline models, using five medium-sized benchmark data sets. These baselines include Snelson and Ghahramani (2006), Hensman et al. (2013), Nguyen and Bonilla (2014), and Snelson and Ghahramani (2007). The proposed model outperforms with statistical significance all

the other baselines in 4 out of 5 data sets. Finally, we compare the proposed method to the GP with stochastic variational inference (SVI) (Hensman et al. 2013) on large-scale data sets with up to 2 million samples when stochastic optimization is enabled. The proposed method is shown to outperform SVI in terms of the accuracy-time trade-off.

The rest of the paper is organized as follows. Section 2 introduces the background of GP regression and sparse GP approximation. Section 3 presents the proposed model: a variational hierarchical mixture of GP experts for regression. Section 4 describes the inference approach for the model. Section 5 presents the experimental results and their analysis. Finally, Sect. 6 concludes the paper.

2 Background

In this section, we first briefly introduce the theoretical background of GP regression (Sect. 2.1). We then give a review on various state-of-the-art sparse GP approximation methods (Sect. 2.2). These methods either inspire or relate to our proposed model.

2.1 Overview of Gaussian process regression

Consider a typical regression problem where a training set $\mathcal{D}$ has N pairs of D-dimensional inputs $\mathbf{x}_n$ and one-dimensional outputs y_n, i.e., $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$ with $\mathbf{x}_n \in \mathcal{X} \subset \mathbb{R}^D$ and $y_n \in \mathbb{R}$. Here $\mathbf{x}_n$ is a row vector for $n = 1, \ldots, N$. Let $\mathbf{X}$ and $\mathbf{y}$ collectively represent the training inputs and outputs, respectively: $\mathbf{X} = ((\mathbf{x}_1)^{\mathrm{T}}, \ldots, (\mathbf{x}_N)^{\mathrm{T}})^{\mathrm{T}}$ and $\mathbf{y} = (y_1, \ldots, y_N)^{\mathrm{T}}$. Our task is to compute the outputs $\mathbf{y}^*$ at new test locations $\mathbf{X}^*$, given $\mathbf{X}$ and $\mathbf{y}$.

By definition, a GP is a collection of random variables, any finite number of which have a joint Gaussian distribution. A GP is completely specified by a mean function $m(\mathbf{x})$ and a covariance function $\kappa(\mathbf{x}, \mathbf{x}')$. In GP regression, we assume that there is an underlying latent function $f(\mathbf{x}) : \mathcal{X} \longmapsto \mathbb{R}$ which follows a GP prior. We can write the GP as: $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. According to the definition of GP, any collection of function values has a joint prior Gaussian distribution. We have

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{m}_{\mathbf{X}}, \mathbf{K}_{\mathbf{XX}}), \tag{1}$$

where $\mathbf{f} = [f_1, \ldots, f_N]^{\mathrm{T}}$ with $f_n \equiv f(\mathbf{x}_n)$, $\mathbf{m}_{\mathbf{X}} = [m(\mathbf{x}_1), \ldots, m(\mathbf{x}_N)]^{\mathrm{T}}$, and $\mathbf{K}_{\mathbf{XX}}$ denotes the covariance matrix formed by evaluating $\kappa(\mathbf{x}, \mathbf{x}')$ at all pairs of input vectors. The observed output y_n is then related to the latent variable f_n by

$$y_n = f_n + \epsilon_n, \tag{2}$$

where ϵ_n is a zero-mean independent and identically distributed Gaussian noise with variance σ^2, i.e. $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$. The above relationship between y_n and f_n can also be expressed in the form of a normal distribution as

$$p(y_n|f_n) = \mathcal{N}(f_n, \sigma^2). \tag{3}$$

The likelihood $p(\mathbf{y}|\mathbf{f})$ is then a factorized Gaussian:

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma^2 \boldsymbol{I}). \tag{4}$$

Our objects of interest are the marginal likelihood $p(\mathbf{y})$ and the prediction for $\mathbf{f}^*$ at new test points $\mathbf{X}^*$, i.e., $p(\mathbf{f}^*|\mathbf{y})$. Their derivations are given below.

Marginal likelihood The marginal likelihood $p(\mathbf{y})$ is used for model selection. In particular, the hyperparameters of the mean function $m(\mathbf{x})$ and covariance function $\kappa(\mathbf{x}, \mathbf{x}')$ as well

as the noise variance σ^2 can be fixed by maximizing $p(\mathbf{y})$. Using the general properties of Gaussian distributions given in Eq. (52), $p(\mathbf{y})$ can be calculated in closed-form as:

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f} = \mathcal{N}(\mathbf{m_X}, \mathbf{K_{XX}} + \sigma^2\boldsymbol{I}). \tag{5}$$

Prediction We consider how to make prediction for $\mathbf{f}^*$ at test points $\mathbf{X}^*$. Let $\mathbf{K_{AB}}$ denote a covariance matrix formed by evaluating the function $\kappa(\mathbf{x}, \mathbf{x}')$ at all pairs of points $(\mathbf{x}, \mathbf{x}')$ with $\mathbf{x}$ in $\mathbf{A}$ and $\mathbf{x}'$ in $\mathbf{B}$. The property of GP gives the following joint prior distribution for $\mathbf{f}$ and $\mathbf{f}^*$:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m_X} \\ \mathbf{m_{X^*}} \end{bmatrix}, \begin{bmatrix} \mathbf{K_{XX}} & \mathbf{K_{XX^*}} \\ \mathbf{K_{X^*X}} & \mathbf{K_{X^*X^*}} \end{bmatrix} \right) \tag{6}$$

Using the Gaussian identities presented in Section A.2 of Williams and Rasmussen (2006a), the predictive distribution for $\mathbf{f}^*$ given the noise-free observations $\mathbf{f}$ can be computed as:

$$p(\mathbf{f}^*|\mathbf{f}) = \mathcal{N}\left(\mathbf{K_{X^*X}}\mathbf{K_{XX}^{-1}}(\mathbf{f} - \mathbf{m_X}) + \mathbf{m_{X^*}}, \quad \mathbf{K_{X^*X^*}} - \mathbf{K_{X^*X}}\mathbf{K_{XX}^{-1}}\mathbf{K_{XX^*}}\right) \tag{7}$$

In realistic situations, we do not have access to the function values $\mathbf{f}$ but the noisy observation $\mathbf{y}$. To make prediction for $\mathbf{f}^*$ using $\mathbf{y}$, we first derive the joint prior distribution for $\mathbf{y}$ and $\mathbf{f}^*$. This can be done by replacing the term corresponding to $p(\mathbf{f})$ in Eq. (6) with that of $p(\mathbf{y})$:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{m_X} \\ \mathbf{m_{X^*}} \end{bmatrix}, \begin{bmatrix} \mathbf{K_{XX}} + \sigma^2\boldsymbol{I} & \mathbf{K_{XX^*}} \\ \mathbf{K_{X^*X}} & \mathbf{K_{X^*X^*}} \end{bmatrix} \right) \tag{8}$$

Finally, the predictive distribution for $\mathbf{f}^*$ given $\mathbf{y}$ can be computed as:

$$\begin{aligned} p(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}\big(&\mathbf{K_{X^*X}}[\mathbf{K_{XX}} + \sigma^2\boldsymbol{I}]^{-1}(\mathbf{y} - \mathbf{m_X}) + \mathbf{m_{X^*}}, \\ &\mathbf{K_{X^*X^*}} - \mathbf{K_{X^*X}}[\mathbf{K_{XX}} + \sigma^2\boldsymbol{I}]^{-1}\mathbf{K_{XX^*}}\big) \end{aligned} \tag{9}$$

By subtracting the mean of $f(\mathbf{x})$ from $f(\mathbf{x})$ if necessary, we can assume, without loss of generality, that $m(\mathbf{x})$ is equal to $\mathbf{0}$. For notational simplicity, we will take the mean function to be zero hereinafter, unless otherwise stated.

The dominant cost in GP inference is the inversion of the covariance matrix $[\mathbf{K_{XX}} + \sigma^2\boldsymbol{I}]$ in Eqs. (5) and (9), which requires a computational time of $O(N^3)$. This is prohibitive for large data sets. In the next section, we discuss sparse approximation methods to reduce the computational cost for GP regression.

2.2 Sparse approximation for Gaussian process regression

We first provide an overview on sparse GP approximation while focusing on the approximation framework by Quiñonero-Candela and Rasmussen (2005) on which many popular sparse GP approximation methods can be constructed. We then discuss the two alternative sparse approximations based on variational inference presented in Titsias (2009) and Hensman et al. (2013), which provide the theoretical framework for inference in the proposed model.

2.2.1 Overview of sparse GP approximation

Sparse GP approximation methods aim to reduce the computation cost of GP regression by representing all the training data using a small set of M inducing points. Each inducing point consists of an inducing input $\mathbf{u}_m \in \mathcal{X}$ and the corresponding inducing variable g_m, which is the latent function value evaluated at $\mathbf{u}_m$, i.e. $g_m = f(\mathbf{u}_m)$. Let $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_M]^{\mathrm{T}}$ and

$\mathbf{g} = [g_1, \ldots, g_M]^{\mathrm{T}}$. Given the inducing inputs $\mathbf{U}$ and the posterior $p(\mathbf{g}|\mathbf{y})$, predictions can be made in $O(M^3)$ time complexity:

$$\begin{aligned} p(\mathbf{f}^*|\mathbf{y}) &= \int p(\mathbf{f}^*|\mathbf{g}) p(\mathbf{g}|\mathbf{y}) d\mathbf{g} \\ &= \int \mathcal{N}\left(\mathbf{K}_{\mathbf{X}^*\mathbf{U}}\mathbf{K}_{\mathbf{U}\mathbf{U}}^{-1}\mathbf{g},\ \ \mathbf{K}_{\mathbf{X}^*\mathbf{X}^*} - \mathbf{K}_{\mathbf{X}^*\mathbf{U}}\mathbf{K}_{\mathbf{U}\mathbf{U}}^{-1}\mathbf{K}_{\mathbf{U}\mathbf{X}^*}\right) p(\mathbf{g}|\mathbf{y}) d\mathbf{g}. \end{aligned}$$

Learning the posterior $p(\mathbf{g}|\mathbf{y})$ efficiently requires an additional assumption about the relationship between the training data and the inducing points. The approximation framework presented in Quiñonero-Candela and Rasmussen (2005) focuses on representing this relationship by the conditional distribution $p(\mathbf{f}|\mathbf{g})$, which can be calculated exactly as:

$$p(\mathbf{f}|\mathbf{g}) = \mathcal{N}\left(\mathbf{K}_{\mathbf{X}\mathbf{U}}\mathbf{K}_{\mathbf{U}\mathbf{U}}^{-1}\mathbf{g},\, \mathbf{K}_{\mathbf{X}\mathbf{X}} - \mathbf{Q}_{\mathbf{X}\mathbf{X}}\right), \tag{10}$$

where $\mathbf{Q}_{\mathbf{X}\mathbf{X}} = \mathbf{K}_{\mathbf{X}\mathbf{U}}\mathbf{K}_{\mathbf{U}\mathbf{U}}^{-1}\mathbf{K}_{\mathbf{U}\mathbf{X}}$. Quiñonero-Candela and Rasmussen (2005) shows that by imposing different approximation assumptions on $p(\mathbf{f}|\mathbf{g})$, various sparse approximation methods proposed in the literature can be derived. We take the popular approximation methods FITC (Snelson and Ghahramani 2006) and PIC (Snelson and Ghahramani 2007) for examples. FITC is based on the assumption that the training latent variables $\mathbf{f}$ are independent given $\mathbf{g}$ so that the conditional distribution $p(\mathbf{f}|\mathbf{g})$ is approximated by $q(\mathbf{f}|\mathbf{g}) = \prod_{n=1}^{N} p(f_n|\mathbf{g})$, i.e.,

$$p(\mathbf{f}|\mathbf{g}) \approx q(\mathbf{f}|\mathbf{g}) = \mathcal{N}\left(\mathbf{K}_{\mathbf{X}\mathbf{U}}\mathbf{K}_{\mathbf{U}\mathbf{U}}^{-1}\mathbf{g},\, \mathrm{diag}[\mathbf{K}_{\mathbf{X}\mathbf{X}} - \mathbf{Q}_{\mathbf{X}\mathbf{X}}]\right).$$

The PIC method (Snelson and Ghahramani 2007) first partitions the data set into several clusters. It is then based on the assumption that the latent variables from different partitions are independent given $\mathbf{g}$, resulting in a block diagonal covariance in the approximate conditional:

$$p(\mathbf{f}|\mathbf{g}) \approx q(\mathbf{f}|\mathbf{g}) = \mathcal{N}\left(\mathbf{K}_{\mathbf{X}\mathbf{U}}\mathbf{K}_{\mathbf{U}\mathbf{U}}^{-1}\mathbf{g},\, \mathrm{blockdiag}[\mathbf{K}_{\mathbf{X}\mathbf{X}} - \mathbf{Q}_{\mathbf{X}\mathbf{X}}]\right).$$

The above approximation, combined with the exact GP prior for the inducing point $p(\mathbf{g}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{U}\mathbf{U}})$, is equivalents to approximating the GP prior for training latent values $p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{X}\mathbf{X}})$ by a new distribution $q(\mathbf{f})$, where

$$q(\mathbf{f}) = \int q(\mathbf{f}|\mathbf{g}) p(\mathbf{g}) d\mathbf{g} = \mathcal{N}(\mathbf{0}, \mathbf{Q}_{\mathbf{X}\mathbf{X}} + \mathrm{diag}[\mathbf{K}_{\mathbf{X}\mathbf{X}} - \mathbf{Q}_{\mathbf{X}\mathbf{X}}]) \tag{11}$$

for FITC, and

$$q(\mathbf{f}) = \int q(\mathbf{f}|\mathbf{g}) p(\mathbf{g}) d\mathbf{g} = \mathcal{N}(\mathbf{0}, \mathbf{Q}_{\mathbf{X}\mathbf{X}} + \mathrm{blockdiag}[\mathbf{K}_{\mathbf{X}\mathbf{X}} - \mathbf{Q}_{\mathbf{X}\mathbf{X}}]). \tag{12}$$

for PIC. See Quiñonero-Candela and Rasmussen (2005) for a proof of Eqs. (11) and (12). It can be seen from Eqs. (11) and (12) that the covariance matrix $\mathbf{K}_{\mathbf{X}\mathbf{X}}$ in the original GP prior $p(\mathbf{f})$ is approximated by a low-rank covariance matrix in $q(\mathbf{f})$, effectively reducing the cost of matrix conversion, and hence the overall computational complexity, to $O(NM^2)$.

In the above sparse approximation methods, model selection, including the selection of the inducing inputs $\mathbf{U}$, is done through maximizing the approximated marginal likelihood $p(\mathbf{y}) \approx q(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f}) q(\mathbf{f}) d\mathbf{f}$. In this way, the inducing inputs become additional kernel hyperparameters. Optimizing with respect to all unknown hyperparameters may lead to overfitting. In addition, the solution is not guaranteed to be close to the original model since the prior has been modified in response to training data. Next, we discuss two alternative GP approximation methods based on variational inference that overcome the above limitation by treating inducing inputs as variational parameters.

2.2.2 Sparse GP approximation based on variational inference

The variational method proposed by Titsias (2009) selects the inducing inputs and the hyper-parameters by maximizing a lower bound of the exact marginal likelihood. In particular, the bound is derived as follows. First, the following inequality is used to obtain a lower bound on $p(\mathbf{y}|\mathbf{g})$:

$$\ln p(\mathbf{y}|\mathbf{g}) \geq \mathbb{E}_{p(\mathbf{f}|\mathbf{g})}[\ln p(\mathbf{y}|\mathbf{f})]. \tag{13}$$

This bound is substituted into the equation $p(y) = \int p(\mathbf{y}|\mathbf{g})p(\mathbf{g})d\mathbf{g}$, and the inducing latent variables $\mathbf{g}$ are then marginalized out to give a tractable lower bound on the marginal likelihood. As this bound is maximized, the Kullback–Leibler (KL) divergence between the variational distribution and the exact GP posterior distribution over the latent function value is minimized. The inducing inputs are defined as the variational parameters which are tuned to minimize this divergence. This way, overfitting is avoided, and the solution provided by the sparse model is indeed an approximation to the exact one since their distance is minimized.

The above variational method has computational complexity of $O(NM^2)$, which is still prohibitive for large data sets. Stochastic variational inference (Hoffman et al. 2013), where optimization can be carried out using mini-batches of data, is one possible way to scale down variational inference framework. However, it can only be applied to probabilistic models that have a set of global variables and that factorize in the observations and latent variables. Hensman et al. (2013) propose to employ stochastic variational inference for GP regression by introducing additional variational parameters into the bound derived in Titsias (2009) to act as global variables. In particular, instead of marginalizing the latent variables $\mathbf{g}$ out as in Titsias (2009), they explicitly approximate the posterior distribution for $\mathbf{g}$ by a variational normal distribution $q(\mathbf{g}) = \mathcal{N}(\mathbf{g}|\mathbf{m}, \mathbf{S})$, and use the variational parameters $\mathbf{m}$ and $\mathbf{S}$ as global variables. The following standard variational inequality is applied on the log marginal likelihood:

$$\ln p(\mathbf{y}) \geq \mathbb{E}_{q(\mathbf{g})}[\ln p(\mathbf{y}|\mathbf{g})] - \mathrm{KL}[q(\mathbf{g})||p(\mathbf{g})]. \tag{14}$$

Here $\mathrm{KL}(q||p)$ denotes the Kullback–Leibler (KL) divergence between distributions p and q. Substituting (13) into (14) results in a further bound on the marginal likelihood

$$\ln p(\mathbf{y}) \geq \mathbb{E}_{q(\mathbf{g})}[\mathbb{E}_{p(\mathbf{f}|\mathbf{g})}[\ln p(\mathbf{y}|\mathbf{f})]] - \mathrm{KL}[q(\mathbf{g})||p(\mathbf{g})]. \tag{15}$$

Since the likelihood $p(\mathbf{y}|\mathbf{f})$ is a factorized Gaussian, the bound given in (15) can be calculated as

$$\begin{aligned}\ln p(\mathbf{y}) \geq \sum_{n=1}^{N} \Bigg\{ &\ln \mathcal{N}\left(y_n|\mathbf{K}_{\mathbf{x}_n\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{m}, \sigma^2\mathbf{I}\right) - \frac{1}{2\sigma^2}\mathbf{K}_{\mathbf{x}_n\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{S}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{K}_{\mathbf{U}\mathbf{x}_n} \\ &- \frac{1}{2\sigma^2}\left(\mathbf{K}_{\mathbf{x}_n\mathbf{x}_n} - \mathbf{K}_{\mathbf{x}_n\mathbf{U}}\mathbf{K}_{\mathbf{UU}}^{-1}\mathbf{K}_{\mathbf{U}\mathbf{x}_n}\right)\Bigg\} - \mathrm{KL}[q(\mathbf{g})||p(\mathbf{g})].\end{aligned} \tag{16}$$

The above bound has a unique optimum in terms of $\mathbf{m}$ and $\mathbf{S}$, at which point it becomes equal to the original bound derived by Titsias (2009). In addition, since the first part of this bound can be written as sum of N terms, each corresponds to a training data point, optimization can then be performed using mini-batches of data. This results in a complexity of $O(BM^2)$, where B is the batch size.

3 Variational hierarchical mixture of Gaussian process experts

This section presents a variational hierarchical mixture of Gaussian process experts for regression. The proposed model has a two-layer hierarchical structure. In the upper layer, a sparse GP, hereafter referred to as the *global* GP, is used to coarsely model the entire data set. In the lower layer, a gating network divides the input space into regions within which a specific local GP *expert* is used for finer modeling. The graphical representation of the proposed hierarchical mixture of Gaussian process experts model is shown in Fig. 1.

To simplify inference, let $\mathbf{y}_0$ and $\mathbf{y}$ denote the training outputs of the upper and lower layers, respectively: $\mathbf{y}_0$ is a duplicate of $\mathbf{y}$. We now have a new training set: $\mathcal{D}' = \{\mathbf{X}, \mathbf{y}, \mathbf{y}_0\}$.

The upper layer is associated with a latent function $f_0(\mathbf{x})$. The function is modeled with a global sparse GP which has a zero mean function and a covariance function $\kappa_0(\mathbf{x}, \mathbf{x}')$. The covariance function is parameterized with the hyperparameter set $\boldsymbol{\theta}_0$. The global sparse GP is augmented with a set of inducing inputs $\{\mathbf{u}_1^{(0)}, \ldots, \mathbf{u}_P^{(0)}\} \subset \mathcal{X}$, which are collectively represented by $\mathbf{U}_0$. The latent function values at the training inputs and inducing inputs are denoted by $\mathbf{f}_0 = (f_0(\mathbf{x}_1), \ldots, f_0(\mathbf{x}_N))^{\mathrm{T}}$ and $\mathbf{g}_0 = (f_0(\mathbf{u}_1^{(0)}), \ldots, f_0(\mathbf{u}_P^{(0)}))^{\mathrm{T}}$, respectively. The latent function values $\mathbf{f}_0$ and the observed outputs $\mathbf{y}_0$ are related by a Gaussian distributed likelihood $p(\mathbf{y}_0|\mathbf{f}_0) = \mathcal{N}(\mathbf{f}_0, \sigma_0^2\mathbf{I})$. Let $\mathbf{K}^{(0)}$ denote the covariance matrix evaluated using the function $\kappa_0(\mathbf{x}, \mathbf{x}')$. The following distributions, defined by their pdfs, can be obtained using standard Gaussian process methodologies:

$$p(\mathbf{g}_0) = \mathcal{N}\left(\mathbf{g}_0|\mathbf{0}, \mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right), \tag{17}$$

$$p(\mathbf{f}_0|\mathbf{g}_0) = \mathcal{N}\left(\mathbf{f}_0|\mathbf{K}^{(0)}_{\mathbf{X}\mathbf{U}_0}\left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1}\mathbf{g}_0, \mathbf{K}^{(0)}_{\mathbf{X}\mathbf{X}} - \mathbf{K}^{(0)}_{\mathbf{X}\mathbf{U}_0}\left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1}\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{X}}\right). \tag{18}$$

There are T GP experts in the lower layer. The k-th expert is associated with a latent function $f_k(\mathbf{x})$, which is modeled using a local sparse GP. The GP has a covariance function

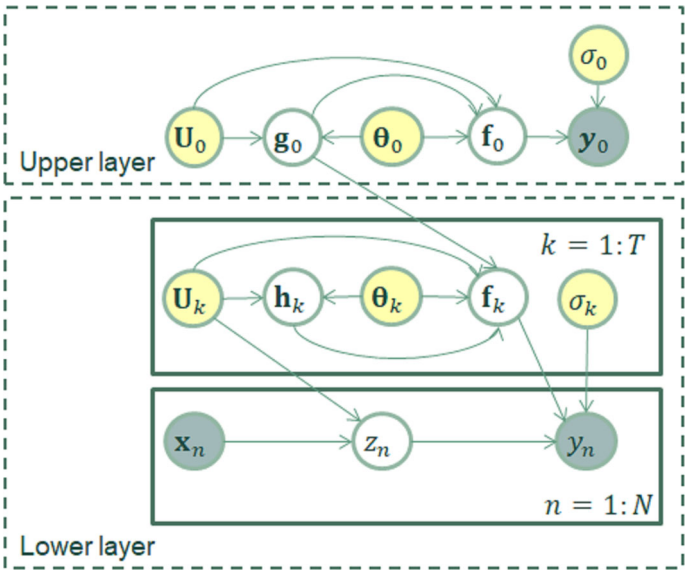


Fig. 1 Graphical representation of the hierarchical mixture of Gaussian process experts model

$\kappa_k(\mathbf{x}, \mathbf{x}')$, which is parameterized with a hyperparameter set $\boldsymbol{\theta}_k$. Each local sparse GP is augmented with a set of M inducing inputs $\{\mathbf{u}_1^{(k)}, \ldots, \mathbf{u}_M^{(k)}\}$, collectively represented by $\mathbf{U}_k$. To enforce correlation among the local experts, all the local sparse GPs share a prior mean function $m(\mathbf{x})$, which encodes global information from the upper layer:

$$m(\mathbf{x}) = \mathbf{K}_{\mathbf{x}\mathbf{U}_0}^{(0)} \left[\mathbf{K}_{\mathbf{U}_0\mathbf{U}_0}^{(0)}\right]^{-1} \mathbf{g}_0. \tag{19}$$

Equation (19) implies that the mean function value at a point $\mathbf{x}$ is calculated as the mean of the global latent variable $f_0(\mathbf{x})$ conditioned on the global inducing variables $\mathbf{g}_0$.

Let $\mathbf{f}_k$ and $\mathbf{g}_k$ denote the vectors of latent function variables of the k-th local expert at training and inducing points, respectively. The properties of GP result in the following distributions

$$p(\mathbf{g}_k|\mathbf{g}_0) = \mathcal{N}\left(\mathbf{g}_k|\bar{\mathbf{g}}_k, \mathbf{K}_{\mathbf{U}_k\mathbf{U}_k}^{(k)}\right), \tag{20}$$

$$p(\mathbf{f}_k|\mathbf{g}_k, \mathbf{g}_0) = \mathcal{N}\left(\mathbf{f}_k|\mathbf{K}_{\mathbf{X}\mathbf{U}_k}^{(k)}\left[\mathbf{K}_{\mathbf{U}_k\mathbf{U}_k}^{(k)}\right]^{-1}(\mathbf{g}_k - \bar{\mathbf{g}}_k) + \bar{\mathbf{f}},\right.$$
$$\left.\mathbf{K}_{\mathbf{X}\mathbf{X}}^{(k)} - \mathbf{K}_{\mathbf{X}\mathbf{U}_k}^{(k)}\left[\mathbf{K}_{\mathbf{U}_k\mathbf{U}_k}^{(k)}\right]^{-1}\mathbf{K}_{\mathbf{U}_k\mathbf{X}}^{(k)}\right), \tag{21}$$

where $\mathbf{K}^{(k)}$ denotes the covariance matrix evaluated using the local kernel function $\kappa_k(\mathbf{x}, \mathbf{x}')$, and $\bar{\mathbf{f}}$ and $\bar{\mathbf{g}}_k$ denote the prior mean values at the training data and at the inducing inputs $\mathbf{U}_k$, respectively, i.e., $\bar{\mathbf{f}} = m(\mathbf{X}) = \mathbf{K}_{\mathbf{X}\mathbf{U}_0}^{(0)}[\mathbf{K}_{\mathbf{U}_0\mathbf{U}_0}^{(0)}]^{-1}\mathbf{g}_0$ and $\bar{\mathbf{g}}_k = m(\mathbf{U}_k) = \mathbf{K}_{\mathbf{U}_k\mathbf{U}_0}^{(0)}[\mathbf{K}_{\mathbf{U}_0\mathbf{U}_0}^{(0)}]^{-1}\mathbf{g}_0$. For simplicity, we introduce new latent variables $\mathbf{h}_k = \mathbf{g}_k - \bar{\mathbf{g}}_k$ to substitute for $\mathbf{g}_k$ at the inducing inputs. The prior and conditional distributions given in Eqs. (20) and (21) become:

$$p(\mathbf{h}_k) = \mathcal{N}\left(\mathbf{h}_k|\mathbf{0}, \mathbf{K}_{\mathbf{U}_k\mathbf{U}_k}^{(k)}\right), \tag{22}$$

$$p(\mathbf{f}_k|\mathbf{h}_k, \mathbf{g}_0) = \mathcal{N}\left(\mathbf{f}_k|\mathbf{K}_{\mathbf{X}\mathbf{U}_k}^{(k)}\left[\mathbf{K}_{\mathbf{U}_k\mathbf{U}_k}^{(k)}\right]^{-1}\mathbf{h}_k + \mathbf{K}_{\mathbf{X}\mathbf{U}_0}^{(0)}\left[\mathbf{K}_{\mathbf{U}_0\mathbf{U}_0}^{(0)}\right]^{-1}\mathbf{g}_0,\right.$$
$$\left.\mathbf{K}_{\mathbf{X}\mathbf{X}}^{(k)} - \mathbf{K}_{\mathbf{X}\mathbf{U}_k}^{(k)}\left[\mathbf{K}_{\mathbf{U}_k\mathbf{U}_k}^{(k)}\right]^{-1}\mathbf{K}_{\mathbf{U}_k\mathbf{X}}^{(k)}\right). \tag{23}$$

For each observation $(\mathbf{x}_n, y_n)$, we have a corresponding latent variable z_n indicating the expert it belongs to. Subsequently, the observed outputs $\mathbf{y}$ of the lower layer have the following likelihood:

$$\begin{aligned} p(y_n|f_1(\mathbf{x}_n), \ldots, f_T(\mathbf{x}_n)) &= \prod_{k=1}^{\mathrm{T}} p(y_n|f_k(\mathbf{x}_n))^{[z_n==k]} \\ &= p(y_n|f_{z_n}(\mathbf{x}_n)) \\ &= \mathcal{N}\left(f_{z_n}(\mathbf{x}_n), \sigma_{z_n}^2\right), \end{aligned}$$

where σ_k denotes the noise variance hyperparameter for expert k.

Expert indicators are specified by a gating network based on the inputs. Since the target here is large-scale problems, the simple gating network suggested in Nguyen and Bonilla (2014) is employed to facilitate fast expert allocation. For this gating network, data points closer to the underlying inducing points of an expert are given higher probabilities to be assigned to that expert. The prior over the expert indicator variable z_n is defined as

$$p(z_n = k|\mathbf{x}_n) = \frac{\mathcal{N}(\mathbf{x}_n|\mathbf{m}_k, \mathbf{V})}{\sum_{j=1}^{\mathrm{T}} \mathcal{N}(\mathbf{x}_n|\mathbf{m}_j, \mathbf{V})}, \tag{24}$$

where each mean $\mathbf{m}_k$ and the covariance $\mathbf{V} = \text{diag}(v_1, \ldots, v_D)$ are given by

$$\mathbf{m}_k = \frac{1}{M} \sum_{m=1}^{M} \mathbf{u}_m^{(k)}, \tag{25}$$

$$v_d = \frac{1}{T(M-1)} \sum_{k=1}^{\mathrm{T}} \sum_{m=1}^{M} \left(u_{md}^{(k)} - m_{kd} \right)^2 . \tag{26}$$

Equation (24) can be interpreted as a probabilistic assignment of data point $\mathbf{x}_n$ to one of the T experts. This prior is based on the observation that the closer a data point to $\mathbf{m}_k$, the more similar it is to the inducing inputs of expert k and the better its output can be predicted by that expert. The rationale behind the choice of this expert allocation mechanism is twofold. First, the formulation of the prior over expert indicators as proportional to a Gaussian distribution makes learning of their approximate posterior analytically tractable via variational inference. Second, this expert allocation prior gives rise to further approximation on the expert indicator variables to reduce the overall computational complexity of the model as will be seen in Sect. 4.3.

4 Inference

Learning of the model is realized through a two-step variational inference algorithm which optimizes an evidence lower bound of the log marginal likelihood. The derivation of this bound is presented in Sect. 4.1. The two-step variational inference algorithm is given Sect. 4.2. The computational cost of the algorithm can be reduced using the cost reduction approximation and the application of stochastic optimization, which are presented in Sects. 4.3 and 4.4, respectively. Finally, the formulation of the predictive distribution is given in Sect. 4.5.

4.1 The evidence lower bound

For the sake of brevity, we introduce the variables $\mathbf{f}$, $\mathbf{h}$, $\mathbf{U}$, $\boldsymbol{\theta}$ and $\mathbf{z}$ to represent the set of all variables $\mathbf{f}_k$, $\mathbf{h}_k$, $\mathbf{U}_k$, $\boldsymbol{\theta}_k$ and z_n, respectively, with $k = 1, \ldots, T$ and $n = 1, \ldots, N$.

The inference problem for our model involves estimating the posterior distribution of the latent variables $p(\mathbf{f}, \mathbf{f}_0, \mathbf{h}, \mathbf{g}_0, \mathbf{z}|\mathbf{y}, \mathbf{y}_0)$, and fixing the kernel hyperparameters and the inducing inputs. Our target is to use variational inference with the possibility of applying stochastic optimization for very large data sets. For this purpose, a set of global hidden variables is required so that the model conditioned on these variables factorizes in the observations and latent variables; see Figure 1 in Hensman et al. (2013) for an illustration of such models. The inducing latent variables $\mathbf{g}_0$ and $\mathbf{h}_k$, for $k = 1, \ldots, T$, are well-suited to perform the role of global variables in our model. However, marginalizing these variables as in the variational sparse GP (Titsias 2009) eliminates the global parameters and re-introduces dependencies between the observations. Hence, following Hensman et al. (2013), we choose to represent the variational distributions of these variables explicitly as $q(\mathbf{g}_0)$ and $q(\mathbf{h}_k)$ for $k = 1, \ldots, T$. It can be seen later that the variational distributions for $\mathbf{f}$ and $\mathbf{f}_0$ can be derived in terms of $q(\mathbf{h})$ and $q(\mathbf{g}_0)$. We then approximate the joint posterior distribution of $\mathbf{h}$, $\mathbf{g}_0$ and $\mathbf{z}$ by a factorized tractable variational distribution,

$$p(\mathbf{z}, \mathbf{h}, \mathbf{g}_0|\mathbf{y}, \mathbf{y}_0) \approx q(\mathbf{z}, \mathbf{h}, \mathbf{g}_0) = \prod_{n=1}^{N} q(z_n) q(\mathbf{g}_0) \prod_{k=1}^{\mathrm{T}} q(\mathbf{h}_k). \tag{27}$$

A lower bound on the log marginal likelihood is first derived by applying the standard variational equation

$$\ln p(\mathbf{y}, \mathbf{y}_0) \geq \mathbb{E}_{q(\mathbf{z},\mathbf{h},\mathbf{g}_0)}[\ln p(\mathbf{y}, \mathbf{y}_0|\mathbf{z}, \mathbf{h}, \mathbf{g}_0)] - \mathrm{KL}(q(\mathbf{z}, \mathbf{h}, \mathbf{g}_0)||p(\mathbf{z}, \mathbf{h}, \mathbf{g}_0)). \tag{28}$$

Substituting for $q(\mathbf{z}, \mathbf{h}, \mathbf{g}_0)$ by its factorization given in (27) leads to the following lower bound:

$$\begin{aligned}\ln p(\mathbf{y}, \mathbf{y}_0) \geq\ & \mathbb{E}_{q(\mathbf{z})q(\mathbf{g}_0)q(\mathbf{h})}[\ln p(\mathbf{y}|\mathbf{z}, \mathbf{h}, \mathbf{g}_0)] \\ & + \mathbb{E}_{q(\mathbf{g}_0)}[\ln p(\mathbf{y}_0|\mathbf{g}_0)] - \mathrm{KL}(q(\mathbf{h})||p(\mathbf{h})) \\ & - \mathrm{KL}(q(\mathbf{g}_0)||p(\mathbf{g}_0)) - \mathrm{KL}(q(\mathbf{z})||p(\mathbf{z})).\end{aligned} \tag{29}$$

Applying Jensen's inequality to the conditional probabilities $p(\mathbf{y}|\mathbf{z}, \mathbf{h}, \mathbf{g}_0)$ and $p(\mathbf{y}_0|\mathbf{g}_0)$ yields

$$\ln p(\mathbf{y}|\mathbf{z}, \mathbf{h}, \mathbf{g}_0) \geq \mathbb{E}_{p(\mathbf{f}|\mathbf{h},\mathbf{g}_0)}[\ln p(\mathbf{y}|\mathbf{f}, \mathbf{z})], \tag{30}$$

and

$$\ln p(\mathbf{y}_0|\mathbf{g}_0) \geq \mathbb{E}_{p(\mathbf{f}_0|\mathbf{g}_0)}[\ln p(\mathbf{y}_0|\mathbf{f}_0)]. \tag{31}$$

Substituting Eqs. (30) and (31) into (29) results in a further lower bound on the log marginal likelihood:

$$\begin{aligned}\ln p(\mathbf{y}, \mathbf{y}_0) \geq\ & \mathbb{E}_{q(\mathbf{z})}[\mathbb{E}_{q(\mathbf{f})}[\ln p(\mathbf{y}|\mathbf{f}, \mathbf{z})]] + \mathbb{E}_{q(\mathbf{f}_0)}[\ln p(\mathbf{y}_0|\mathbf{f}_0)] \\ & - \mathrm{KL}(q(\mathbf{h})||p(\mathbf{h})) - \mathrm{KL}(q(\mathbf{g}_0)||p(\mathbf{g}_0)) - \mathrm{KL}(q(\mathbf{z})||p(\mathbf{z})),\end{aligned} \tag{32}$$

where $q(\mathbf{f}_0)$ and $q(\mathbf{f})$ are defined as

$$q(\mathbf{f}_0) \triangleq \int p(\mathbf{f}_0|\mathbf{g}_0)q(\mathbf{g}_0)d\mathbf{g}_0, \tag{33}$$

$$q(\mathbf{f}) \triangleq \int p(\mathbf{f}|\mathbf{h}, \mathbf{g}_0)q(\mathbf{h})q(\mathbf{g}_0)d\mathbf{h}d\mathbf{g}_0. \tag{34}$$

Note that the difference between the left hand side and the right hand side of Eq. (31) is given by the KL divergence $\mathrm{KL}(p(\mathbf{f}_0|\mathbf{g}_0)||p(\mathbf{f}_0|\mathbf{g}_0, \mathbf{y}_0))$. This KL divergence is minimized when $\mathbf{g}_0$ gives sufficient statistics for $\mathbf{f}_0$. In practice, this assumption of $\mathbf{g}_0$ being a sufficient statistic is unlikely to hold since the number of inducing points is less than the number of data points. However, the bound can be maximized with respect to (w.r.t.) $\mathbf{U}_0$. This minimizes the KL divergence keeping Jensen's bound tight and ensuring that $\mathbf{U}_0$ are well distributed among the training input $\mathbf{X}$. Similarly, the difference between the two sides of Eq. (30) is given by the KL divergence $\mathrm{KL}(p(\mathbf{f}|\mathbf{h}, \mathbf{g}_0)||p(\mathbf{f}_0|\mathbf{h}, \mathbf{g}_0, \mathbf{z}, \mathbf{y}_0))$, which is minimized when the combination of $\mathbf{h}$ and $\mathbf{g}_0$ gives sufficient statistics for $\mathbf{f}$. This bound can be maximized w.r.t. $\mathbf{z}$ and $\mathbf{U}_k$ for $k = 0, \ldots, T$.

It has been shown by Titsias (2009) and Hensman et al. (2013) that the implicit optimal variational distribution $q(\mathbf{g}_0)$ to maximize the right hand side of Eq. (31) is Gaussian [see Eq. (10) in Titsias (2009) and Eq. (3) in Hensman et al. (2013)]. Similarly, the optimal variational distribution $q(\mathbf{h}, \mathbf{g}_0)$ to maximize the right hand side of Eq. (30), and hence the optimal $q(\mathbf{h}_k)$, are also Gaussian. We parametrize them as follows:

$$q(\mathbf{g}_0) \triangleq \mathcal{N}(\mathbf{g}_0|\mathbf{m}_0, \mathbf{S}_0) \tag{35}$$

and

$$q(\mathbf{h}_k) \triangleq \mathcal{N}(\mathbf{h}_k|\mathbf{m}_k, \mathbf{S}_k). \tag{36}$$

Equation (34) shows the joint variational distribution of the local latent variables $\mathbf{f}_1, \ldots, \mathbf{f}_T$ of the experts. It is computationally expensive to calculate this joint distribution. However, we will see that only their marginal distributions are needed. In particular, because the likelihoods factorize as $p(\mathbf{y}_0|\mathbf{f}_0) = \prod_{n=1}^{N} p(y_n|f_0(\mathbf{x}_n))$ and $p(\mathbf{y}|\mathbf{f}) = \prod_{n=1}^{N}\prod_{k=1}^{\mathrm{T}} p(y_n|f_k(\mathbf{x}_n))^{[z_n==k]}$, and $q(\mathbf{z})$ is assumed to factorize as in (27), the bound (32) becomes

$$\begin{aligned}\ln p(\mathbf{y}, \mathbf{y}_0) \geq & \sum_{n=1}^{N}\sum_{k=1}^{\mathrm{T}} q(z_n = k)\mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))] \\ & + \sum_{n=1}^{N} \mathbb{E}_{q(f_0(\mathbf{x}_n))}[\ln p(y_n|f_0(\mathbf{x}_n))] - \mathrm{KL}(q(\mathbf{h})||p(\mathbf{h})) \\ & - \mathrm{KL}(q(\mathbf{g}_0)||p(\mathbf{g}_0)) - \mathrm{KL}(q(\mathbf{z})||p(\mathbf{z})). \end{aligned} \tag{37}$$

It can be seen from the equation above that only the marginals of $q(\mathbf{f})$ and $q(\mathbf{f}_0)$, i.e. $q(f_k(\mathbf{x}_n))$ for $k = 0, \ldots, T$ and $n = 1, \ldots, N$, are needed to compute the expectations in (37). Using the assumed distributions for $q(\mathbf{g}_0)$ and $q(\mathbf{h}_k)$ in Eqs. (35) and (36) and the conditionals in Eqs. (18) and (23), the following functional forms of the marginal distributions are obtained

$$q(f_0(\mathbf{x}_n)) = \mathcal{N}\left(f_0(\mathbf{x}_n)|\left[\mathbf{a}_n^{(0)}\right]^{\mathrm{T}}\mathbf{m}_0, \kappa_0(\mathbf{x}_n, \mathbf{x}_n) + \left[\mathbf{a}_n^{(0)}\right]^{\mathrm{T}}\left(\mathbf{S}_0 - \mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right)\mathbf{a}_n^{(0)}\right), \tag{38}$$

and

$$\begin{aligned} q(f_k(\mathbf{x}_n)) = & \mathcal{N}\left(f_k(\mathbf{x}_n)|\left[\mathbf{a}_n^{(k)}\right]^{\mathrm{T}}\mathbf{m}_k + \left[\mathbf{a}_n^{(0)}\right]^{\mathrm{T}}\mathbf{m}_0,\right. \\ & \left.\kappa_k(\mathbf{x}_n, \mathbf{x}_n) + \left[\mathbf{a}_n^{(k)}\right]^{\mathrm{T}}\left(\mathbf{S}_k - \mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}\right)\mathbf{a}_n^{(k)} + \left[\mathbf{a}_n^{(0)}\right]^{\mathrm{T}}\mathbf{S}_0\mathbf{a}_n^{(0)}\right), \end{aligned} \tag{39}$$

for $k = 1, \ldots, T$. Here $\mathbf{a}_n^{(k)}$ is a vector of the n-th column of the matrix $[\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}]^{-1}\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{X}}$ for $k = 0, \ldots, T$. The detailed derivation of Eqs. (38) and (39) are given in Appendix A. Subsequently, the expected likelihood terms from the bound (37) can be calculated as follows (see Appendix A)

$$\begin{aligned}\mathbb{E}_{q(f_0(\mathbf{x}_n))}[\ln p(y_n|f_0(\mathbf{x}_n))] = & \ln \mathcal{N}\left(y_n|\left[\mathbf{a}_n^{(0)}\right]^{\mathrm{T}}\mathbf{m}_0, \sigma_0^2\right) \\ & - \frac{1}{2\sigma_0^2}\mathrm{Tr}\left(\mathbf{S}_0\mathbf{a}_n^{(0)}\left[\mathbf{a}_n^{(0)}\right]^{\mathrm{T}}\right) - \frac{1}{2\sigma_0^2}l_{nn}^{(0)} \end{aligned} \tag{40}$$

and

$$\begin{aligned}\mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))] = & \ln \mathcal{N}\left(y_n|\left[\mathbf{a}_n^{(k)}\right]^{\mathrm{T}}\mathbf{m}_k + \left[\mathbf{a}_n^{(0)}\right]^{\mathrm{T}}\mathbf{m}_0, \sigma_k^2\right) \\ & - \frac{1}{2\sigma_k^2}\mathrm{Tr}\left(\mathbf{S}_k\mathbf{a}_n^{(k)}\left[\mathbf{a}_n^{(k)}\right]^{\mathrm{T}}\right) \\ & - \frac{1}{2\sigma_k^2}\mathrm{Tr}\left(\mathbf{S}_0\mathbf{a}_n^{(0)}\left[\mathbf{a}_n^{(0)}\right]^{\mathrm{T}}\right) - \frac{1}{2\sigma_k^2}l_{nn}^{(k)} \end{aligned} \tag{41}$$

for $k = 1, \ldots, T$. Here $l_{nn}^{(k)}$ is the n-th diagonal element of $\mathbf{K}^{(k)}_{\mathbf{X}\mathbf{X}} - \mathbf{K}^{(k)}_{\mathbf{X}\mathbf{U}_k}[\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}]^{-1}\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{X}}$ for $k = 0, \ldots, T$.

4.2 The variational inference algorithm

Inference in our model is performed by maximizing the lower bound (37) on the log marginal likelihood w.r.t. the variational distributions $q(\mathbf{z})$, $q(\mathbf{h})$ and $q(\mathbf{g}_0)$, the inducing inputs, the noise variance and the kernel hyperparameters. Notice that maximizing the lower bound w.r.t. the noise variance and the kernel hyperparameters does not necessarily make it closer to the log marginal likelihood since the latter depends on them. In fact, only the maximization of the lower bound w.r.t. the variational distributions $q(\mathbf{z})$, $q(\mathbf{h})$, $q(\mathbf{g}_0)$ and the inducing inputs brings it closer to the log marginal likelihood. Subsequently, maximizing the lower bound w.r.t. the noise variance and the kernel hyperparameters elevates the log marginal likelihood.

If we assume that the variational distribution $q(z_n)$ is a multinomial distribution, then the KullbackLeibler divergence $\mathrm{KL}(q(\mathbf{z})||p(\mathbf{z}))$ is analytically tractable and the bound can be maximized w.r.t. all the variational parameters and hyperparameters using straightforward gradient based optimization. However, this method will result in a time complexity of $O(NM^2T)$. In particular, computing the bound in Eq. (37) requires the computation of NT terms $\mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))]$ for $k = 1, \ldots, T$ and $n = 1, \ldots, N$, where the computation of each term as given in Eq. (41) has the time complexity of $O(M^2)$. The resulting linear time scaling in T is undesirable. In fact, a well-known problem with the sparse method using inducing points is that each inducing point only sculpts out the approximate posterior in a small region of the input space around it (Snelson 2008). Consequently, when the range of the inputs is large compared to this supported range, many inducing points are required to maintain the accuracy of the approximation. This means that, in many applications such as in time-series settings or in spatial datasets, the number of inducing points must grow with the number of data points, i.e. M must be scaled with N; and hence these inducing-point schemes do not reduce the computational complexity. The mixture-of-experts structures like our model provides a solution for this problem by dividing the whole input space into small regions each of which belongs to the responsibility of an expert. In this way, the M inducing points of each expert will only need to provide support for a smaller region. On the other hand, this means that instead of scaling the number of inducing points M with the number of data points N, we need to increase the number of experts T as N grows. So the linear scaling of the time complexity in T is undesirable for our model.

Here we present a more scalable inference algorithm for which the cost is independent of the number of experts T. The key for this cost reduction is to apply an approximation to the variational distribution $q(\mathbf{z})$ in order to bring some of the terms $q(z_n = k)$ in Eq. (37) to zero, thereby, reduce the total number of terms $\mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))]$ to be computed. This cost reduction approximation (described in detail in Sect. 4.3) will be combined into a two-step inference algorithm which is presented below. In particular, the optimization is performed by iteratively alternating between the two steps

1. Fix $q(\mathbf{z})$, and maximize the lower bound w.r.t. the parameters of $q(\mathbf{h})$ and $q(\mathbf{g}_0)$, the inducing inputs, the noise variance and the kernel hyperparameters, using gradient based optimization.
2. Fix $q(\mathbf{h})$, $q(\mathbf{g}_0)$, the inducing inputs, the noise variance and the kernel hyperparameters, and maximize the bound w.r.t. $q(\mathbf{z})$.

We will now discuss each of the two steps in details. For the first step, the following equation contains the relevant terms of the bound to be maximized:

$$\mathcal{L}_1(\mathcal{D}, \boldsymbol{\gamma}) = \sum_{n=1}^{N} \sum_{k=1}^{\mathrm{T}} q(z_n = k) \mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))]$$

$$+\sum_{n=1}^{N}\mathbb{E}_{q(f_0(\mathbf{x}_n))}[\ln p(y_n|f_0(\mathbf{x}_n))]-\mathrm{KL}(q(\mathbf{h})||p(\mathbf{h}))$$
$$-\mathrm{KL}(q(\mathbf{g}_0)||p(\mathbf{g}_0)). \tag{42}$$

Here, $\boldsymbol{\gamma}$ denotes the vector containing the inducing inputs, the noise variance, the kernel hyperparameters, and the parameters of $q(\mathbf{h})$ and $q(\mathbf{g}_0)$. During optimization, to maintain positive-definiteness of the covariances $\mathbf{S}_k$ for $k = 0, \ldots, T$, we represent them using a lower triangular form $\mathbf{S}_k = \mathbf{L}_k\mathbf{L}_k^{\mathrm{T}}$, as suggested in Hensman et al. (2015), and perform unconstrained optimization of the bound w.r.t. $\mathbf{L}_k$. All the terms in $\mathcal{L}_1$ are tractable, and their derivatives w.r.t. $\mathbf{m}_k$, $\mathbf{L}_k$, $\mathbf{U}_k$, σ_k and θ_k (for $k = 0, \ldots, T$) can be calculated by applying straight-forward algebra (see Appendix B).

In the second step, the lower bound on the log marginal likelihood is maximized w.r.t. $q(\mathbf{z})$. Because $q(\mathbf{h})$ and $q(\mathbf{g}_0)$ are fixed, and so are $q(\mathbf{f})$ and $q(\mathbf{f}_0)$, the bound becomes

$$\begin{aligned}\mathcal{L}_2(\mathcal{D}, q(\mathbf{z})) &= \mathbb{E}_{q(\mathbf{z})}\left\{\mathbb{E}_{q(\mathbf{f})}[\ln p(\mathbf{y}|\mathbf{f},\mathbf{z})]\right\} - \mathrm{KL}(q(\mathbf{z})||p(\mathbf{z})) + \text{const}\\ &= \mathbb{E}_{q(\mathbf{z})}\left\{\mathbb{E}_{q(\mathbf{f})}[\ln (p(\mathbf{y}|\mathbf{f},\mathbf{z})p(\mathbf{z}))]\right\} - \mathbb{E}_{q(\mathbf{z})}[\ln q(\mathbf{z})] + \text{const}\\ &= \mathbb{E}_{q(\mathbf{z})}[\ln \tilde{p}(\mathbf{y},\mathbf{z})] - \mathbb{E}_{q(\mathbf{z})}[\ln q(\mathbf{z})] + \text{const},\end{aligned} \tag{43}$$

where $\tilde{p}(\mathbf{y}, \mathbf{z}))$ is a new distribution defined by the relation

$$\begin{aligned}\ln \tilde{p}(\mathbf{y},\mathbf{z})) &= \mathbb{E}_{q(\mathbf{f})}[\ln (p(\mathbf{y}|\mathbf{f},\mathbf{z})p(\mathbf{z}))] + \text{const}\\ &= \mathbb{E}_{q(\mathbf{f})}[\ln (p(\mathbf{y},\mathbf{z}|\mathbf{f}))] + \text{const}.\end{aligned}$$

It can be recognized that (43) is the negative Kullback–Leibler divergence between $q(\mathbf{z})$ and $\tilde{p}(\mathbf{y}, \mathbf{z})$. Thus maximizing (43) is equivalent to minimizing the Kullback–Leibler divergence, which occurs when $q(\mathbf{z}) = \tilde{p}(\mathbf{y}, \mathbf{z})$, i.e.,

$$\ln q(\mathbf{z}) = \mathbb{E}_{q(\mathbf{f})}[\ln(p(\mathbf{y}|\mathbf{f},\mathbf{z})] + \ln p(\mathbf{z}), \tag{44}$$

or

$$\begin{aligned}\sum_{n=1}^{N}\ln q(z_n) = &\sum_{n=1}^{N}\sum_{k=1}^{\mathrm{T}}\mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))^{[z_n==k]}]\\ &+\sum_{n=1}^{N}\sum_{k=1}^{\mathrm{T}}\ln p(z_n=k)^{[z_n==k]}.\end{aligned}$$

Using the prior over z_n given in Eq. (24), it can be seen that z_n follows a multinomial posterior distribution, i.e., $q(z_n{=}k){=}r_{nk}$, where $r_{nk} = \rho_{nk}/\sum_{i=1}^{\mathrm{T}}\rho_{ni}$ is the *responsibility* of expert k for $\mathbf{x}_n$, and ρ_{nk} is given by

$$\ln \rho_{nk} = \ln \mathcal{N}(\mathbf{x}_n|\mathbf{m}_k, \mathbf{V}) + \mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))] + \text{const}. \tag{45}$$

4.3 Computational complexity and cost reduction approximation

We now look into the computational cost of the algorithm and describe approximation techniques to reduce it.

Computational complexity Assuming that the global GP and each of the local experts have the same number of inducing points, i.e. $M = P$, the cost of computing the KL divergences and their derivatives in (42) is $O(M^3T)$. Since the number of required inducing points M is expected to be much smaller than the number of training samples N, most of the cost will arise

from computing the expected likelihood terms $\mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))]$ for $k = 0, \ldots, T$ and $n = 1, \ldots, N$, and their derivatives. This computation is required for both steps and has the overall time complexity of $O(NM^2T)$.

Cost reduction approximation The cost in the first step can be reduced with the maximum a posteriori (MAP) assignment as suggested by Nguyen and Bonilla (2014). In particular, the experts are assumed to be responsible for disjoint subsets of the inputs, i.e., each data point is explained by only one expert. This is done by assigning each point to only the expert of highest responsibility:

$$z_n = \operatorname*{argmax}_k r_{nk}. \tag{46}$$

The responsibilities are then reassigned as follows:

$$r_{nk} = \begin{cases} 1, & \text{iff } z_n = k, \\ 0, & \text{otherwise.} \end{cases}$$

It can be observed that, for the computation of the bound $\mathcal{L}_1$ [given by Eq. (42)] in the first step, the term $q(f_k(\mathbf{x}_n))$ is only needed when $q(z_n = k)$ is non-zero, i.e. r_{nk} is non-zero. With the new MAP expert assignment, this only happens when the point $\mathbf{x}_n$ is assigned to expert k. As a result, the time complexity for the first step is reduced to $O(NM^2)$.

We can further observe that $\ln \rho_{nk}$ in Eq. (45) comprises two terms. The first term increases as the distance between $\mathbf{x}_n$ and the expert center $\mathbf{m}_k$ decreases. The second term, measuring the quality of prediction by expert k, increases when $\mathbf{x}_n$ is similar to the inducing inputs of the expert. This is more likely as $\mathbf{x}_n$ is getting closer to $\mathbf{m}_k$. This observation allows us to bypass the expensive computation of the second term and replace the above MAP assignment with a simplified expert assignment for the second step:

$$z_n = \operatorname*{argmax}_k \mathcal{N}(\mathbf{x}_n|\mathbf{m}_k, \mathbf{V}). \tag{47}$$

As a result, the overall time complexity of the proposed algorithm is reduced to $O(NM^2)$.

4.4 Stochastic optimization

Since the objective function (37) is given as the sum over N data points, we can optimize it in a distributed fashion by parallelizing the computation over the data points, or in a stochastic fashion by selecting a mini-batch of the data at random for each iteration. Here we discuss how to use stochastic optimization for our model in more details. First, we rewrite the objective $\mathcal{L}_1$ in Eq. (42) as follows

$$\mathcal{L}_1(\mathcal{D}, \boldsymbol{\gamma}) = \sum_{n=1}^{N} \lambda_n - \mathrm{KL}(q(\mathbf{h})||p(\mathbf{h})) - \mathrm{KL}(q(\mathbf{g}_0)||p(\mathbf{g}_0)).$$

where

$$\lambda_n = \sum_{k=1}^{\mathrm{T}} q(z_n = k)\mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))] + \mathbb{E}_{q(f_0(\mathbf{x}_n))}[\ln y_n|f_0(\mathbf{x}_n)]$$

In each iteration t, we randomly sample a set of B examples from the data. We denote the set by $S^{(t)}$. The objective $\mathcal{L}_1$ is then approximated by

$$\tilde{\mathcal{L}}_1(S^{(t)}, \boldsymbol{\gamma}) = \frac{N}{B} \sum_{x_i \in S^{(t)}} \lambda_i - \mathrm{KL}(q(\mathbf{h})||p(\mathbf{h})) - \mathrm{KL}(q(\mathbf{g}_0)||p(\mathbf{g}_0)),$$

as though $S^{(t)}$ is replicated N/B times to form the data set. The full algorithm with stochastic optimization is presented in Algorithm 1. The algorithm has the time complexity of $\max(O(BM^2), O(M^3))$ and the memory complexity of $\max(O(BM), O(M^2))$.

As previously mentioned in Sect. 4.1, in order to enable the decomposability of the lower bound $\mathcal{L}_1$ and hence stochastic optimization, the variational distributions of the inducing variables are represented explicitly using the parameters $\mathbf{m}_k$ and $\mathbf{L}_k$ for $k = 0, \ldots, T$. The drawback is that $(T+1)M(M+3)/2$ extra parameters are to be optimized, and the joint search space of these parameters is huge for a large number of experts T. However, as we will see in the experiments presented in Sect. 5, especially in the experiments with varying number of experts (up to 100 experts) presented in Sect. 5.3, the proposed method has no problem handling a moderately large number of experts.

Algorithm 1 Model inference with stochastic optimization

1: Initialize the inducing inputs, the noise variance, the kernel hyperparameters, $q(h)$, $q(g0)$ and $q(z)$.
2: Set the learning rate α and the batch size B appropriately.
3: **repeat**
4: Sample a set $S^{(t)}$ of B examples randomly.
5: Update z_n according to Eq. (47), $\forall x_n \in S^{(t)}$.
6: Calculate the gradient $\nabla_{\boldsymbol{\gamma}} \tilde{\mathcal{L}}_1(S^{(t)}, \boldsymbol{\gamma})$
7: Update the current estimate of $\boldsymbol{\gamma}$,

$$\boldsymbol{\gamma}^{(t)} = \boldsymbol{\gamma}^{(t-1)} - \alpha \nabla_{\boldsymbol{\gamma}} \tilde{\mathcal{L}}_1(S^{(t)}, \boldsymbol{\gamma})$$

8: **until** convergence.

4.5 Prediction

The predictive distribution for an unseen data point $\mathbf{x}^*$ is

$$p(y^*|\mathbf{x}^*, \mathbf{y}) = \sum_{k=1}^{\mathrm{T}} p(z^* = k|\mathbf{x}^*) p(y^*|\mathbf{x}^*, \mathbf{y}, z^* = k). \tag{48}$$

That means the final prediction at $\mathbf{x}^*$ is the weighted average of the predictions from T experts with the weights given by $p(z^* = k|\mathbf{x}^*)$. In practice, we find that the prediction by the expert with highest possibility $p(z^* = k|\mathbf{x}^*)$ is better than the weighted prediction. The predictive distribution at $\mathbf{x}^*$ by an expert k can be estimated as

$$p(y^*|\mathbf{x}^*, \mathbf{y}, z^* = k) = \int p(y^*|f^*) p(f^*|\mathbf{x}^*, \mathbf{y}, z^* = k) df^*, \tag{49}$$

where

$$\begin{aligned} p(f^*|\mathbf{x}^*, \mathbf{y}, z^* = k) &= \int p(f^*|\mathbf{f}_k, \mathbf{h}_k, \mathbf{g}_0) p(\mathbf{f}_k, \mathbf{h}_k, \mathbf{g}_0|\mathbf{y}) d\mathbf{f}_k d\mathbf{h}_k d\mathbf{g}_0 \\ &\approx \int p(f^*|\mathbf{f}_k, \mathbf{h}_k, \mathbf{g}_0) p(\mathbf{f}_k|\mathbf{h}_k, \mathbf{g}_0) q(\mathbf{h}_k) q(\mathbf{g}_0) d\mathbf{f}_k d\mathbf{h}_k d\mathbf{g}_0 \\ &= \int p(f^*|\mathbf{h}_k, \mathbf{g}_0) q(\mathbf{h}_k) q(\mathbf{g}_0) d\mathbf{h}_k d\mathbf{g}_0. \end{aligned} \tag{50}$$

The last integral in Eq. (50) results in a normal distribution similar to that in Eq. (39). The predictive distribution at $\mathbf{x}^*$ given in Eq. (49) can be computed by simply adding the noise variance σ_k^2 to the variance of the above normal distribution.

5 Experiments

In this section, we present experiments with data sets of varying size to investigate different aspects of the proposed model. The section is organized as follows. Section 5.1 discusses the experimental methods including the experimental setup and the performance measures. Section 5.2 analyzes the effects of the number of global and local inducing points on the performance of the proposed model. Section 5.3 tests the model using the varying number of experts. Section 5.4 evaluates the ability of the model to handle non-stationarity using two toy data sets. Section 5.5 presents the experiments to compare the performances of various relevant and representative GP regression methods for a number of medium-sized benchmark data sets. Finally, Sect. 5.6 evaluates the performances of the proposed method on large-scale data sets using stochastic optimization.

5.1 Experimental methods

5.1.1 Experimental setup

Each experiment is carried out on a system with Intel® Core(™) i7-4770 CPU at 3.40GHz with 8GB RAM. We use the squared exponential (SE) kernel with automatic relevance determination (ARD) for all the tested GP regression methods in all experiments.

Besides the proposed model—the *variational hierarchical mixture of GP expert* (HMGP), the following GP regression methods are repeatedly studied in our experiments: the *GP with stochastic variational inference* (SVI) (Hensman et al. 2013), the *fully independent training conditional* (FITC) method (Snelson and Ghahramani 2006), the *partially independent conditional* (PIC) method (Snelson and Ghahramani 2007) and the *fast allocated mixture of GP experts* (FGP) (Nguyen and Bonilla 2014). The information regarding the implementation of these methods are as follows. FITC, PIC and FGP are implemented in MATLAB, where optimization is carried out using the LBFGS-B optimizer (Zhu et al. 1997) from GPML package.[1] SVI and the proposed model HMGP are implemented in Python using ADADELTA optimizer (Zeiler 2012) from climin package (Bayer et al. 2015) for learning hyper-parameters and inducing inputs. ADADELTA is chosen for supporting stochastic optimization. We use the implementations of FITC and PIC from GPML package, SVI from GPy package (GPy since 2012), and FGP from the its Github repository.[2] In addition to these methods, a number of other GP regression methods are evaluated and discussed in Sect. 5.6.

We note that HMGP refers to the proposed model with the cost reduction approximation presented in Sect. 4.3. For each experiment with HMGP, the number of global inducing points is set to be the same as the number of local inducing points per expert, except for the experiments in Sect. 5.2. The prediction by HMGP for each test points $\mathbf{x}^*$ is based on the prediction of the expert k which has the highest possibility $p(z^* = k|\mathbf{x}^*)$.

[1] http://www.gaussianprocess.org/gpml/code/matlab/doc/index.html.

[2] https://github.com/trungngv/fgp.

5.1.2 Performance measures

We use the Root-Mean Square Error (RMSE), the Standardized Mean Squared Error (SMSE) and the Mean Standardized Log Loss (MSLL) to measure the quality of the predictions on the test sets. The SMSE is given as the mean squared error of the tested predictor normalized by the variance of the targets of the test cases:

$$\text{SMSE} = \left\langle (y^* - \mu^*)^2 \right\rangle / \left\langle (y^* - \bar{y})^2 \right\rangle, \tag{51}$$

where $\langle . \rangle$ averages over the test data, y^* and μ^* are the real target value and the predicted mean value at test sample $\mathbf{x}^*$, respectively; and $\bar{y}$ is the mean value of the test targets.

The MSLL is obtained by averaging $-\log p(y^*|D, x^*)$ (which is the negative log probability of the real test target value against the predictive distribution) over the test set and then subtracting the same score of a trivial model (which predicts using a Gaussian with the mean and variance of the training data). MSLL takes into account the predictive variances while RMSE and SMSE do not. The lower are these measurements, the better is the predictor.

5.2 Experiments with varying number of inducing points

First, we analyze the effects of the number of global and local inducing points on the performance of the proposed HMGP method. In the experiments, HMGP is tested on three datasets *kin40k* (8 dimensions, 10,000 training, 30,000 test),[3] *pole-telecom* (26 dimensions, 10,000 training, 5000 test),[4] and *sarcos* (21 dimensions, 44484 training, 4449 test) (see footnote 4) using varying numbers of global and local inducing points: 50, 100 and 200. The number of local experts is fixed to 3. The performance in terms of SMSE and MSLL is reported in Table 1.

It can be observed that increasing the number of global or local inducing points generally improves the performance of HMGP in terms of both SMSE and MSLL. However, the impact of increasing the number of global or local inducing points on the performance varies across different data sets. For the *kin40k* data set, a moderately simple data set, increasing the number of global inducing points improves the performance more significantly than increasing the number of local inducing points. The opposite trend is observed for the *sarcos* data set, a highly complex nonlinear one. In this data set, an increase of local inducing points has more impact on the performance than an increase of global inducing points.

The *pole-telecom* data set is somewhere in between *kin40k* and *sarcos* in complexity. The experimental results for this data set in Table 1 exhibit that increasing the number of global inducing points gives a higher improvement in terms of SMSE and a lower improvement in terms of MSLL than those achieved by increasing the same number of local inducing points. In such a data set, choosing a balanced number of global and local inducing points is generally a good choice.

5.3 Experiments with varying number of experts

Here we test the proposed HMGP model with and without the cost reduction approximation (presented in Sect. 4.3) using the varying number of experts T on a spatial dataset. As previously mentioned, when the range of the inputs is large compared to the supporting range of an inducing point such as in time-series or spatial datasets, the number of experts T needs

[3] Available from http://www.cs.toronto.edu/~delve/data/datasets.html.

[4] Available from http://homepages.inf.ed.ac.uk/ckiw/code/gpr_approx.html.

Table 1 Performance of HMGP in terms of SMSE and MSLL using different numbers of global and local inducing points

(a) *kin40k* dataset

Local \ Global	SMSE			MSLL		
	50	100	200	50	100	200
50	0.0706	0.0555	0.0364	-1.456	-1.568	-1.823
100	0.0706	0.0555	0.0364	-1.497	-1.624	-1.836
200	0.0548	0.0500	0.0362	-1.645	-1.682	-1.838

(b) *pole-telecom* dataset

Local \ Global	SMSE			MSLL		
	50	100	200	50	100	200
50	0.0134	0.0094	0.0077	-2.234	-2.294	-2.351
100	0.0121	0.0094	0.0076	-2.365	-2.369	-2.416
200	0.0115	0.0093	0.0075	-2.448	-2.450	-2.452

(c) *sarcos* dataset

Local \ Global	SMSE			MSLL		
	50	100	200	50	100	200
50	0.0210	0.0195	0.0193	-2.162	-2.193	-2.203
100	0.0168	0.0164	0.0151	-2.249	-2.265	-2.303
200	0.0142	0.0127	0.0125	-2.320	-2.358	-2.373

to increase as the number of data points grows. So the linear scaling of the time complexity in T is undesirable for HMGP. The cost reduction approximation was introduced in order to reduce the time complexity of HMGP from $O(NM^2T)$ to $O(NM^2)$. This experiment aims to evaluate the effectiveness of this cost reduction approximation.

We use the monthly price paid data[5] in England and Wales for the period of February to October 2016, and filter for apartments resulting in a dataset with 76,919 entries. Each entry contains a postcode of the apartment for which we cross-reference against a postcode database to get the geographical coordinates: the latitude and longitude. The normalized logarithmic apartment prices are then regressed on the geographical coordinates. We randomly select 10,000 data points as a test set and use the remaining for training. This dataset is similar to that used in Hensman et al. (2013), where the data was for year 2012.

We compare the models before and after applying the cost reduction approximation, which are denoted as HMGP-b and HMGP, respectively. We also use SVI (Hensman et al. 2013) as baseline for comparison. SVI can be considered as a special case of HMGP where the lower layer comprising local experts is removed. Different numbers of local experts T are used for HMGP and HMGP-b. For each method and each T value, the number of inducing points is varied to trace-out speed-accuracy frontiers. The resulted SMSE and MSLL measurements are plotted against the training time in Fig. 2. Note that for HMGP and HMGP-b, the number of global inducing points is set to be equal to the number of local inducing points per expert, i.e., $P = M$. Learning proceeds until convergence or until 1000 iterations are reached. As seen in Fig. 2, HMGP-b and HMGP achieve the similar SMSE and MSLL measurements

[5] The data are available from http://data.gov.uk/dataset/land-registry-monthly-price-paid-data/ (accessed 02-Octorber-2017).

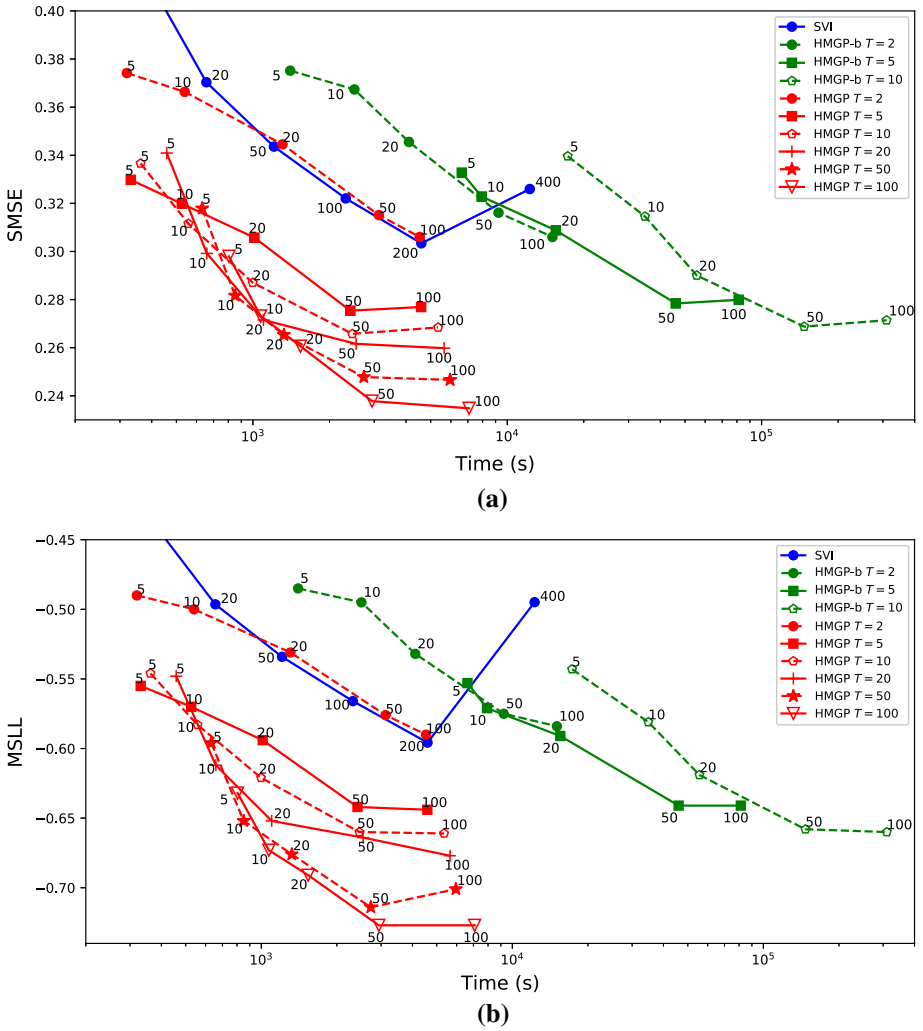


Fig. 2 SMSE and MSLL as functions of training time for the *apartment price* dataset for different GP approximation methods and different T values. Points on each line are annotated with the number of inducing points. Faster and more accurate approximation methods are located towards the bottom left corner of the plots. **a** SMSE versus training time, **b** MSLL versus training time

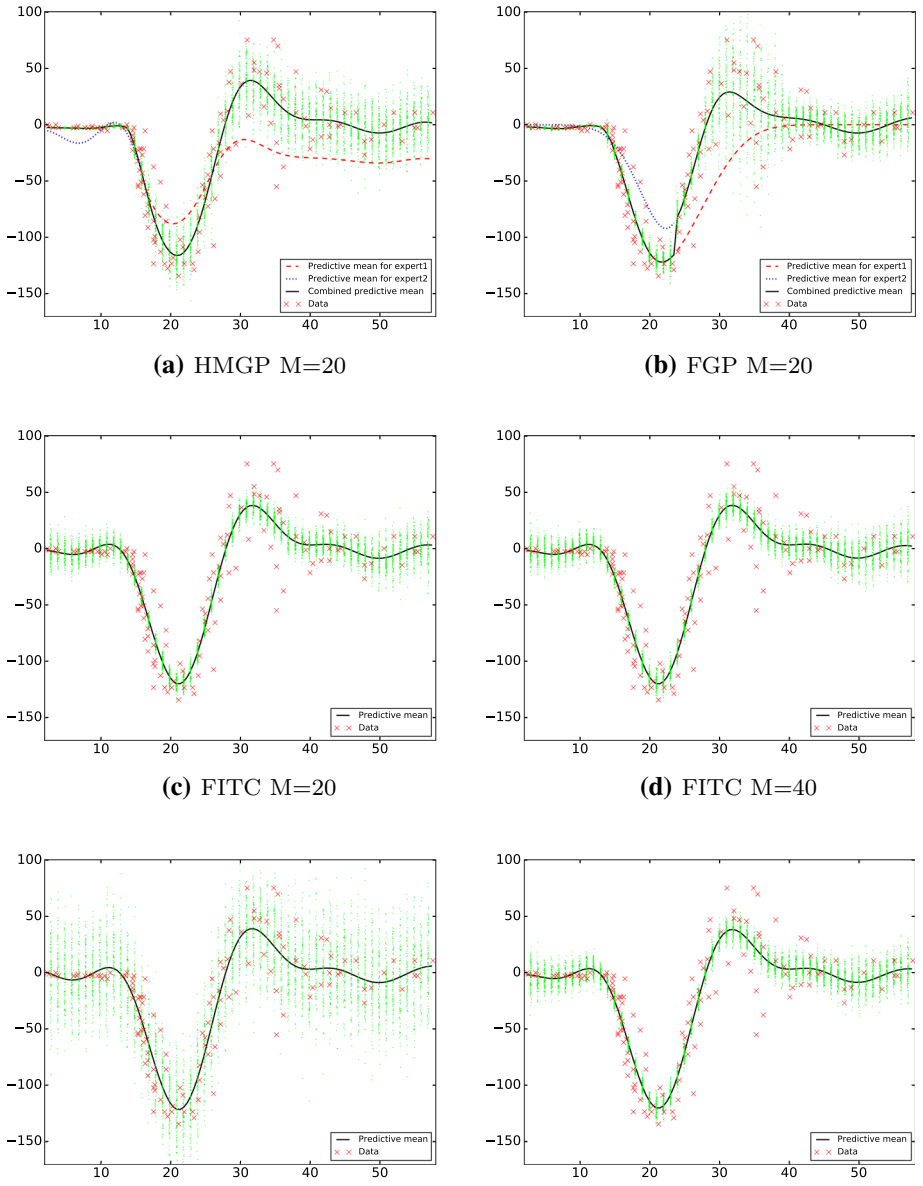
given the same number of experts and inducing points. However, the training time of HMGP-b is much longer. This is due to the time complexity of $O(NM^2T)$ and $O(NM^2)$ for HMGP-b and HMGP, respectively. The prohibitive training time prevents us from testing HMGP-b for $T > 10$. For HMGP, using 2 experts it has similar speed-accuracy performance as SVI. As the number of experts increases from 2 to up to 100, the speed-accuracy performance of HMGP keeps improving; it achieves lower SMSE and MSLL with only slightly increased training time given the same number of inducing points, thanks to the cost reduction approximation technique.

5.4 Experiments on small-sized data sets

Next, we evaluate the ability of the model to deal with non-stationarity using a small-sized dataset where regression result can be easily visualized. Here we use the *motorcycle* data set (Silverman 1985), which contains 133 data points with input-dependent noise as shown in Fig. 3. Four other GP regression methods are used in this experiments: FGP (Nguyen and Bonilla 2014), FITC (Snelson and Ghahramani 2006), PIC (Snelson and Ghahramani 2007) and the full GP. FGP is a mixture of GP experts in which each expert uses only local information. All the tested approximation methods except PIC have the time and memory complexity of $O(NM^2)$ and $O(NM)$, respectively, where M is the total number of inducing points for FITC, and the number of inducing points per experts for FGP and HMGP. PIC has the time and memory complexity of $O(NP^2)$ and $O(NP)$, where $P = \max(M, C)$: M is the number of inducing points and C is the size of clusters, assuming that all the clusters have the same size. In this experiment, we use 2 experts and 20 inducing points per expert in FGP and HMGP ($M = 20$). In PIC, a similar setting is used: 20 inducing points ($M = 20$) and 2 clusters which are formed using k-mean algorithm ($C > 20$ though). FITC is tested with two different settings: $M = 20$ and $M = 40$. We train each method 5 times and select the trained model with the smallest objective value at convergence.

Figure 3 presents the results obtained by the six predictors. For HMGP, the first expert models the beginning part of the data where the noise level is low, and the second expert depicts the remaining part where the noise level is high. This result shows the ability of the proposed method to handle non-stationarity in the data. In this respect, FGP obtains similar result. However, since the proposed method uses both global information and local data, it gives a smoother transition between the two experts while FPG generates a big gap at the boundary. Finally, FITC, PIC and the full GP are not able to model the varying noise level across the data set, which results in a poor predictive distribution (notice the high predictive variances at the beginning part of the data).

Next, we consider a more realistic situation in practical regression problems, where a common trend is observed across the entire data set. To simulate this situation, we modify the *motorcycle* data set by adding a sine function to its outputs: $y = x + 30\sin(x)$. This yields a highly non-linear data set. The results of the six tested methods are shown in Fig. 4. For FPG, the first expert depicts the first part of the data pretty well. However, the second expert is confused by the high noise level in the later part of data, and hence it is not able to model the non-linearity in the data. FITC using 20 inducing points detects common trend of the data but cannot account for such high non-linearity with a small number of global inducing points. FITC with an increased number of inducing points ($M = 40$) results in better predictive means but it still over-estimates the predictive variances at the beginning part of the data. PIC and the full GP give good predictive mean, but they cannot model the different noise levels in the data set for a good predictive distribution. In contrast, the proposed method using both global and local information with separate hyper-parameter sets for the experts performs well on this data set. In addition, in PIC, k-mean algorithm does not take into account the varying noise levels in finding a suitable cluster boundary for the inference, which results in a big gap between two clusters (at around data point 32). In HMGP, the inference and gating network learning are done simultaneously giving a smoother transition at the expert boundaries.

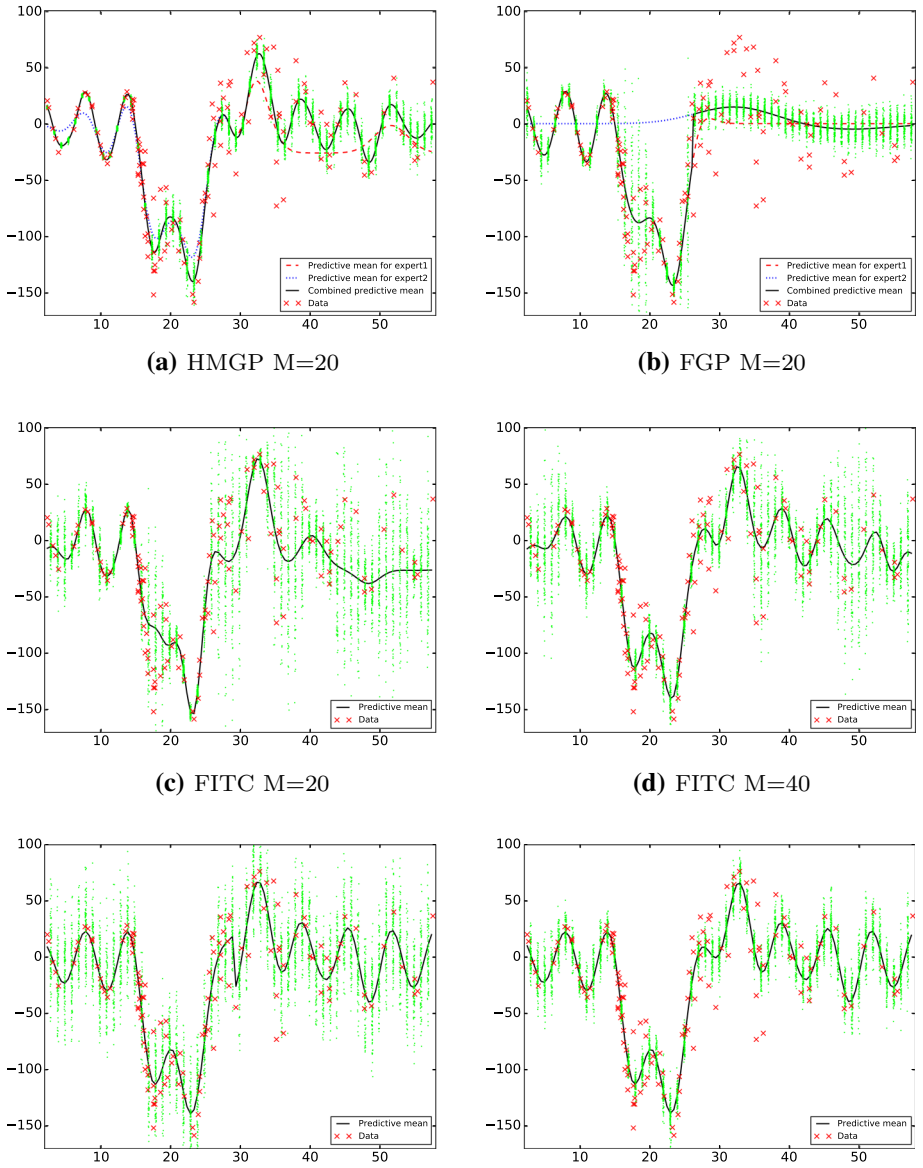


(a) HMGP M=20 **(b)** FGP M=20

(c) FITC M=20 **(d)** FITC M=40

(e) PIC M=20 **(f)** GP

Fig. 3 Test results for *motorcycle* data using HMGP, FGP, FITC, PIC and full GP. Training data are marked with red crosses. Green dots are samples drawn from the predictive distribution evaluated at evenly spaced points (100 samples per point). Solid black line represents the predictive mean. In the top two figures, the predictive means by the two experts, which are represented by red dashed and blue dotted lines, are overlaid by the final combined predictive means (solid black line) (Color figure online)



(a) HMGP M=20 **(b)** FGP M=20

(c) FITC M=20 **(d)** FITC M=40

(e) PIC M=20 **(f)** GP

Fig. 4 Test results for *motorcycle + sine* data using HMGP, FGP, FITC, PIC and full GP. Training data are marked with red crosses. Green dots are samples drawn from the predictive distribution evaluated at evenly spaced points (100 samples per point). Solid black line represents the predictive mean. In the top two figures, the predictive means by the two experts, which are represented by red dashed and blue dotted lines, are overlaid by the final combined predictive means (solid black line) (Color figure online)

5.5 Experiments on medium-sized data sets

In this section, we evaluate the predictive performance of our model on 5 medium-sized benchmark data sets: *kin40k* (8 dimensions, 10,000 training, 30,000 test),[6] *pumadyn32nm* (32 dimensions, 7168 training, 1024 test),[7] *pole-telecom* (26 dimensions, 10,000 training, 5000 test) (see footnote 7), *chem* (15 dimensions, 31,535 training, 31,536 test) (see footnote 7), and *sarcos* (21 dimensions, 44,484 training, 4449 test) (see footnote 7). We use the same training /test splits as in Williams and Rasmussen (2006b), Nguyen and Bonilla (2014) and Chalupka et al. (2013).

We compare the performance of the proposed method (HMGP) to a number of the state-of-the-art GP regression methods: FITC (Snelson and Ghahramani 2006), PIC (Snelson and Ghahramani 2007), FGP (Nguyen and Bonilla 2014), SVI (Hensman et al. 2013), and the local mixture of GP experts (LMGP). LMGP is actually a special case of HMGP when the global sparse GP in the upper layer is removed. In this special case, the predictor is left with a set of local GP experts, and therefore makes use of only local information. FITC and SVI use only global information where the entire data set is summarized by a set of inducing points, while FGP makes use of only local information. Finally, PIC uses both global and local information. The experiment thus allows us to examine whether combining global and local information in the proposed method provides any performance improvement over using either one alone.

No stochastic optimization is used in this experiment, i.e., all the training data points are used in each iteration. All the tested methods have the time and memory complexity of $O(NM^2)$ and $O(NM)$, respectively, where M is the total number of inducing points for FITC and SVI, the number of inducing points per experts for FGP, LMGP and HMGP, and the size of clusters in PIC (assuming that all the clusters have the same size). We choose $M = 500$ for all the methods. In PIC, since we cannot guarantee that all the clusters have the same size, we choose the number of clusters such that the average cluster size is 500, and use k-means algorithm for clustering. We also fix the number of clusters in FPG, LMGP and in the lower-layer of HMGP to 3 in all of the experiments. Each method is run 5 times, and each run is started with different random seeds. The performance is evaluated in terms of the standardized mean squared error (SMSE) and mean standardized log loss (MSLL), which measure the accuracy and confidence of the prediction, respectively.

The optimization process of all the methods is run until convergence or for 1, 000 iterations, whichever is the earlier. Figs. 5, 6, 7, 8 and 9 illustrate the SMSE and MSLL as functions of training time for all the tested methods. In addition, the average performance measures and training times of the tested methods over 5 runs together with their standard deviations are reported in Table 2. First, from Table 2, we observe that HMGP requires a longer training time than SVI and LMGP. In fact, the training time of HMGP is approximately equal to the sum of training times of SVI and LMGP. However, it can be seen from Figs. 5, 6, 7, 8 and 9 that of all the tested models, HMGP is the most efficient one in terms of the time-accuracy trade-off, except for the time-MSLL trade-off in the *pole-telecom* data set. Second, the results in Table 2 show that HMGP also gives the best performance in terms of both accuracy (SMSE) and prediction confidence (MSLL). This applies to all data sets, except for the *pole-telecom* data set, where FGP has a lower MSLL, but a much higher SMSE. HMGP provides significant gains in terms of SMSE compared to the second best predictor in 4 out of 5 data sets (35.3% in *kin40k*, 40% in *pole-telecom*, 58% in *chem* and 25% in *sarcos*).

[6] Available from http://www.cs.toronto.edu/~delve/data/datasets.html.

[7] Available from http://homepages.inf.ed.ac.uk/ckiw/code/gpr_approx.html.

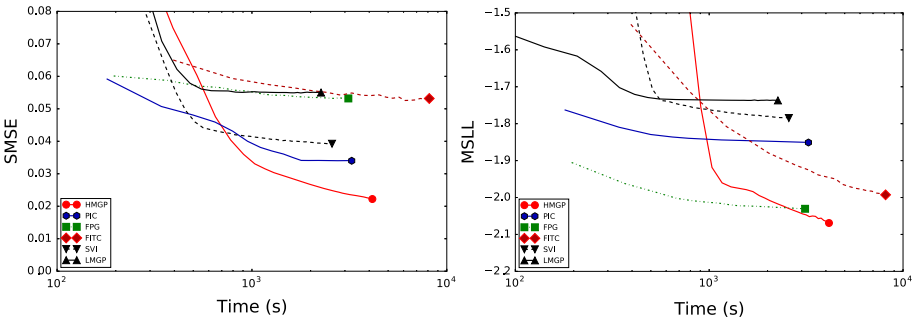


Fig. 5 SMSE and MSLL as functions of training time for the *kin40k* data set

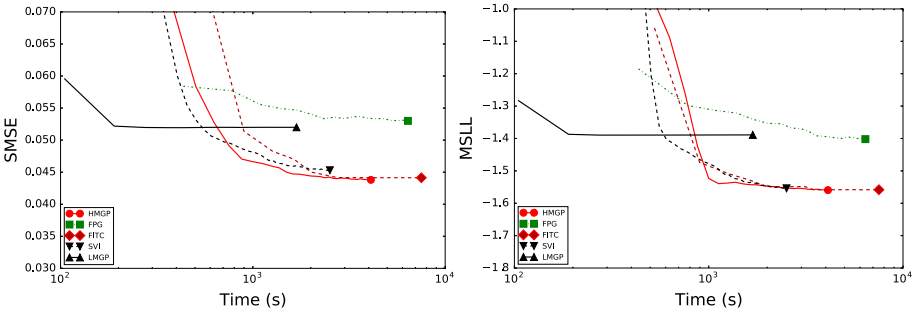


Fig. 6 SMSE and MSLL as functions of training time for the *pumadyn32nm* data set. Since the performance of PIC is very poor on this data set, it has been removed from the plots to increase their resolutions

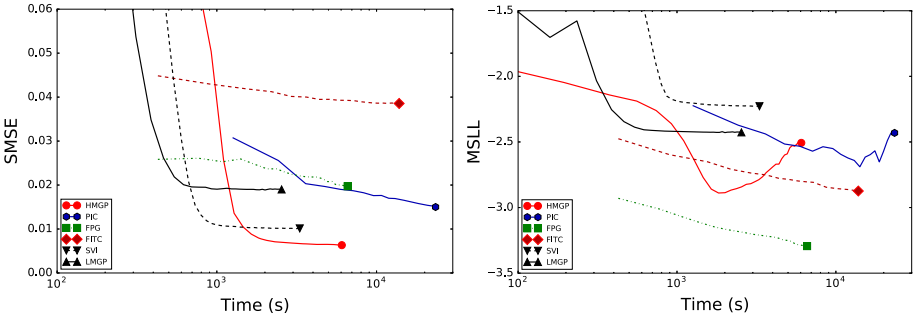


Fig. 7 SMSE and MSLL as functions of training time for the *pole-telecom* data set

Note that the results for HMGP, SVI and LMGP on the *kin40k*, *pole-telecom*, and *sarcos* data sets in Table 2 are consistent with the analysis in Sect. 5.2. In particular, whenever SVI performs better than LMGP, then an increase in the number of global inducing points for HMGP will improve the performance more significantly than an increase in the number of local inducing points. The opposite is also true.

5.6 Experiments on large data sets

In this section, we evaluate the performance of the proposed method using stochastic optimization on two large-scale data sets: the *Million Song* data set (Bertin-Mahieux et al. 2011)

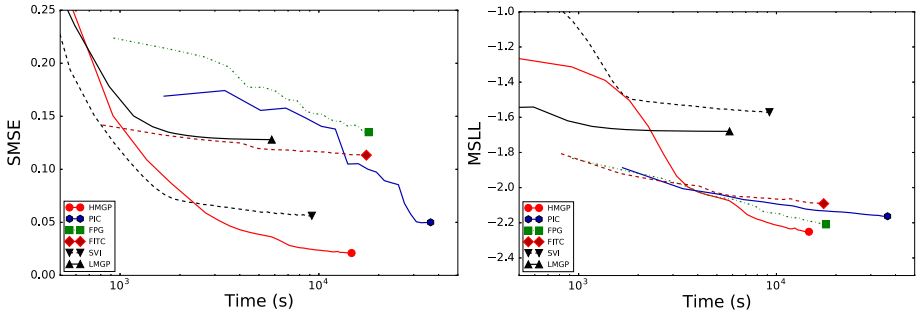


Fig. 8 SMSE and MSLL as functions of training time for the *chem* data set

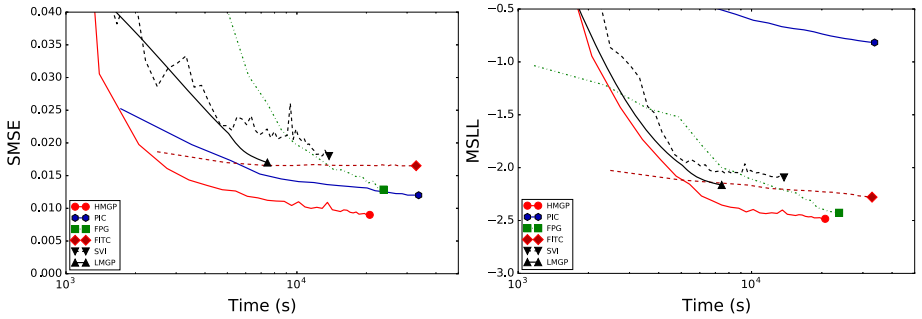


Fig. 9 SMSE and MSLL as functions of training time for the *sarcos* data set

and the *US flight* data set (Hensman et al. 2013). Here we briefly discuss about the GP regression methods that are used for comparison in this section, before giving details regarding the experiments on each of the two aforementioned datasets.

5.6.1 GP regression methods for comparison

The following relevant and representative GP regression methods are studied in this section:

- HGMP with and without stochastic optimization (denoted as HMGP-S and HMGP, respectively)
- SVI with and without stochastic optimization (denoted as SVI-S and SVI, respectively) (Hensman et al. 2013)
- FGP (Nguyen and Bonilla 2014)
- Local-FITC (Nguyen and Bonilla 2014)
- Variational sparse spectrum approximation to GP (VSSGP) (Gal and Turner 2015)
- Variational PIC (Hoang et al. 2015)
- Distributed low-rank-cum-Markov approximation (Distributed LMA) (Hoang et al. 2016)
- Subset of data points (SOD) (Quiñonero-Candela and Rasmussen 2005)

In Local FITC, the training data is randomly divided into small clusters, and a FITC model is separately trained on each cluster. In SOD, a subset of data points from the training set is used for training a full GP. In VSSGP, the GP's stationary covariance function is decomposed into an infinite Fourier series, which is then approximated by a finite one. In variational PIC, Hoang et al. (2015) apply a reverse variational inference procedure to

Table 2 Test results, which include SMSE and MSLL, and training time (along with their respective standard deviations in brackets), for five different benchmark data sets: *kin40k*, *pumadyn32nm*, *pole-telecom*, *chem* and *sarcos*

		kin40k	*pumadyn32nm*	*pole-telecom*	*chem*	*sarcos*
SMSE	HMGP	**0.022 (± 0.000)**	**0.044 (± 0.001)**	**0.006 (± 0.000)**	**0.021 (± 0.004)**	**0.009 (± 0.001)**
	SVI	0.039 (± 0.000)	0.045 (± 0.001)	0.010 (± 0.000)	0.056 (± 0.002)	0.018 (± 0.005)
	LMGP	0.055 (± 0.000)	0.052 (± 0.001)	0.019 (± 0.000)	0.128 (± 0.001)	0.017 (± 0.003)
	FITC	0.053 (± 0.001)	**0.044 (± 0.001)**	0.039 (± 0.001)	0.113(± 0.009)	0.017 (± 0.001)
	FGP	0.053 (± 0.003)	0.053 (± 0.002)	0.020 (± 0.001)	0.135 (± 0.052)	0.013 (± 0.001)
	PIC	0.034 (± 0.002)	10.200 (± 0.053)	0.015 (± 0.000)	0.050 (± 0.002)	0.012 (± 0.002)
MSLL	HMGP	**− 2.069 (± 0.061)**	**− 1.559 (± 0.024)**	− 2.507 (± 0.115)	**− 2.251 (± 0.052)**	**− 2.484 (± 0.065)**
	SVI	− 1.785 (± 0.083)	− 1.557 (± 0.023)	− 2.228 (± 0.016)	− 1.571 (± 0.006)	− 2.093 (± 0.119)
	LMGP	− 1.737 (± 0.031)	− 1.389 (± 0.015)	− 2.427 (± 0.101)	− 1.680 (± 0.020)	− 2.164 (± 0.064)
	FITC	− 1.992 (± 0.025)	− 1.559 (± 0.036)	− 2.847 (± 0.012)	− 2.091 (± 0.041)	− 2.279 (± 0.025)
	FGP	− 2.031 (± 0.023)	− 1.402 (± 0.021)	**− 3.294 (± 0.176)**	− 2.209 (± 0.245)	− 2.427 (± 0.045)
	PIC	− 1.851 (± 0.017)	0.712 (± 0.047)	− 2.431 (± 0.024)	− 2.163 (± 0.250)	− 0.817 (± 0.045)
Training time (s)	HMGP	4147 (± 74)	4109 (± 154)	6059 (± 226)	14,603 (± 317)	20,679 (± 1782)
	SVI	2576 (± 52)	2514 (± 112)	3304 ± 203)	9217 (± 212)	13,778 (± 1088)
	LMGP	2262 (± 101)	1686 (± 247)	2544 ± 350)	5793 (± 363)	7449 (± 542)
	FITC	8128 (± 235)	7519 (± 738)	13,844(± 1325)	17,334 (± 312)	32,832 (± 1842)
	FGP	3124 (± 137)	6402 (± 349)	6588 (± 146)	17,804 (± 1069)	23,715 (± 362)
	PIC	3251 (± 295)	2835 (± 272)	23,412 (± 5770)	36,472 (± 1328)	33,671 (± 2174)

Results are the averages over 5 trials, along with the standard deviation. The best performances are given in bold

derive a stochastic natural gradient ascent method that can achieve asymptotic convergence to the predictive distribution of PIC. Both VSSGP and variational PIC are amendable to stochastic optimization. In LMA (Low et al. 2015), the low-rank approximation of a GP, which is based on inducing points and block structure of input space, is complemented with a Markov approximation of the resulting residual process. Distributed LMA is a distributed implementation of LMA. It allows the observation noise variance to vary across the input space by representing it as a finite realization of a Gaussian Markov random process. Table 3 gives the time and memory complexity of the methods evaluated in this section. It is unclear how much memory is required for distributed LMA from Hoang et al. (2016), hence, this information is not given in Table 3.

The implementation of FGP, HMGP(-S) and SVI(-S) have been discussed in the experimental setup (Sect. 5.1.1). Local FITC and SOD are implemented based on GPML package which is written in MATLAB. They use the LBFGS-B algorithm for optimization. For VSSGP, we use its Github source code,[8] which is written in Python. Like for HMGP(-S) and SVI(-S), ADADELTA optimizer is used for VSSGP to support stochastic optimization. The implementations for variational PIC and distributed LMA are also from their Github repositories.[9,10] Both variational PIC and distributed LMA are written in C++. Their authors implement their own optimizers. We leave this intact since it is non-trivial to implement and integrate LBFGS-B or ADADELTA optimizer to their C++ code base. In addition, incorporating LBFGS-B does not seem to improve the performance (Quang Minh Hoang, personal communication, Oct 12, 2017). To make sure that the optimizers that come with variational PIC and distributed LMA are used efficiently, for each experiment on these methods, we try different configurations for the optimizers (i.e., different learning rates and decay parameters), and report the best results among those configurations. K-mean algorithm is used to partition training sets into blocks for variational PIC and distributed LMA.

5.6.2 Experiments on the Million Song data set

First, we use the *Million Song* data set (Bertin-Mahieux et al. 2011). We use the exact split as in Nguyen and Bonilla (2014), in which 100,000 and 51,630 songs are used for training and testing, respectively. Each song has 90 acoustic features based on which its year of release is to be predicted.

We test the proposed method with and without stochastic optimization (HMGP-S and HMGP), and compare it with 8 other GP predictors: SVI, SVI-S, FGP, Local-FITC, VSSGP, variational PIC, distributed LMA and SOD. As suggested in Nguyen and Bonilla (2014), when using a single machine to train such a large data set, the choice regarding the sparsity and complexity of the predictors should be directed by the memory limit of the computer. In this experiment, subject to the memory limit of the computer which is 8 GB, we set $M = 300$ for FGP, local-FITC, SVI, HMGP and distributed LMA. For SVI-S, HMGP-S, VSSGP and variational PIC, the use of stochastic optimization allows more inducing points to be used given the same time and memory complexity. For these methods, we set $B = M = 2000$, which results in a similar time complexity as the other compared methods and a lower memory complexity. In SOD, 2000 data points is randomly sampled for training and the rest is discarded. We use 3 experts (or clusters) for HMGP, FGP and local-FITC. Optimization is run until convergence or until 1000 iterations is reached, whichever is earlier.

[8] Code for VSSGP is available from https://github.com/yaringal/VSSGP.

[9] Code for variational PIC is available from https://github.com/qminh93/RVGP.

[10] Code for distributed LMA is available from https://github.com/qminh93/dSGP_ICML16.

Table 3 The time and memory complexity of several GP regression methods studied in Sect. 5.6

Method	S?	Time	Memory	Notes
HMGP-S	√	$\max(O(BM^2), O(M^3))$	$\max(O(BM), O(M^2))$	M: the number of inducing points per expert
HMGP		$O(NM^2)$	$O(NM)$	M: the number of inducing points per expert
SVI-S	√	$\max(O(BM^2), O(M^3))$	$\max(O(BM), O(M^2))$	M: the number of inducing points
SVI		$O(NM^2)$	$O(NM)$	M: the number of inducing points
FGP		$O(NM^2)$	$O(NM)$	M: the number of inducing points per expert
Local-FITC		$O(NM^2)$	$O(NM)$	M: the number of inducing points per clusters
SOD		$O(M^3)$	$O(M^2)$	M: the size of the subset
VSSGP	√	$\max(O(BM^2), O(M^3))$	$\max(O(BM), O(M^2))$	M: the number of inducing frequencies
Variational PIC	√	$\max(O(BM^2), O(M^3))$	$\max(O(BM), O(M^2))$	M: the larger between the number of inducing points and the cluster size
Distributed LMA		$O(NM^2)$	–	M: the larger between the number of inducing points and the block size Assume that a single-core machine is used

The second column indicates whether the method is amendable to stochastic optimization. N is the size of the training set. B is the batch size when using stochastic optimization

Table 4 Performance (the average SMSE, MSLL and training time) of different methods, on the Million Song data set; the best performance is indicated in bold typeface

Method	SMSE	MSLL	Training time (h)
HMGP-S	0.690 (± 0.003)	− **0.217** (± 0.015)	6.5
HMGP	**0.678** (± 0.002)	− 0.216 (± 0.014)	18.8
SVI-S	0.705 (± 0.003)	− 0.177 (± 0.011)	3.1
SVI	0.701 (± 0.005)	− 0.185 (± 0.008)	11.8
FGP	0.712 (± 0.004)	− 0.214 (± 0.017)	16.2
Local-FITC	0.732 (± 0.007)	− 0.207 (± 0.024)	15.4
SOD	0.787 (± 0.013)	− 0.114 (± 0.024)	3.7
VSSGP	0.700 (± 0.004)	− 0.098 (± 0.011)	–
Variational PIC	0.981 (± 0.001)	6.730 (± 0.024)	18.8
Distributed LMA	0.969 (± 0.001)	–	46.9

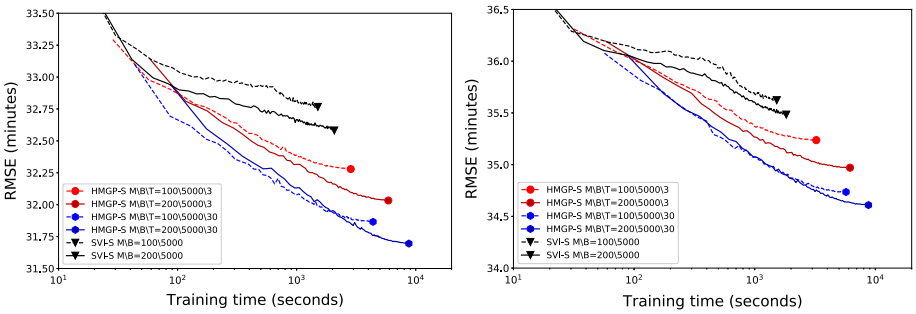
The numbers in brackets indicate the standard deviation over 5 runs

The performances in terms of SMSE and MSLL of all the methods are given in Table 4. The results are averaged over 5 runs. The training time of VSSGP is not available since its Github code runs on GPU. The MSLL measure for the distributed LMA is also not available since predictive variances are not generated from its source code. It can be seen from Table 4 that HMGP and HMGP-S give the best performances among the compared methods. With stochastic optimization, HMGP-S requires a shorter training time while achieving a comparable predictive performance to HMGP.

We notice that variational PIC and distributed LMA perform poorly on this dataset. One factor that might contribute to these poor performances is that even though the partitioning of the input space into blocks and especially the selection of inducing inputs have a great impact on the performance of these methods, they are carried out in advance and are independent of the inference of the models. In particular, the partitioning of the input space into blocks is done using an external clustering algorithm (k-mean algorithm in this case). The inducing inputs are then calculated for each block as the weighted averages of the randomly selected data points from the block and the block center. On high-dimensional datasets like this, the partitioning result from such an external clustering algorithm is noisy, unreliable and non-optimal for the inference of the GP approximation methods. In HMGP, the clustering and the inference including the learning of inducing inputs are done simutaneously, the results from the inference can improve the clustering and vice versa, hence it does not have the same pitfall as the above methods.

5.6.3 Experiments on the US flight data set

The second large-scale data set we consider is the *US flight* data set (Hensman et al. 2013). This data set contains information about all commercial flights in the USA from January 2008 to April 2008. The aim is to predict the delay in reaching the destination based on 8 attributes: age of the aircraft, distance to be covered, airtime, departure time, arrival time, day of the week, day of the month and month. The data set consists of more than 2 million points after removing instances with missing values. For this experiment, we use two different setups for sampling the training/test sets from the *US flight* dataset. The first setup is similar to that used in Hensman et al. (2013) in which the first 800K points from the data set are selected and then



(a) *Flight-700K* split **(b)** *Flight-All* split

Fig. 10 RMSE as a function of training time for the *US flight* dataset using the *Flight-700K* and *Flight-All* splits for HMGP and SVI. Faster and more accurate approximation methods are located towards the bottom left corner of the plots

split randomly into 700K and 100K points for training and testing. In the second setup, 100K points are randomly picked from the entire original data set for testing; and the remaining points are used for training. Hereinafter, we refer to these two setups as the *Flight-700K* and *Flight-All* splits. To perform GP regression on such a large-scale dataset as in the second setup using a single computer, stochastic optimization is required. To be compatible with the experiments in Hensman et al. (2013), RMSE is used as the performance measure for this dataset.

Compare in terms of the performance-time trade-off We first compare HMGP-S and SVI-S in terms of the performance-time trade-off using the two aforementioned training/test splits. This is done by plotting the accuracy measurement RMSE as functions of training time. The result is shown in Fig. 10. Each of the two models HMGP-S and SVI-S is tested with two different numbers of inducing points of 100 and 200, and the same batch size of 5000. HMGP-S is tested with 3 and 30 experts. All the methods are run until convergence or until 10,000 optimization iterations are reached, whichever is earlier. Even though SVI-S takes less time than HMGP-S to complete one optimization iteration, it can be observed from the plot that HMGP-S using either 3 or 30 clusters is more efficient than SVI-S in terms of the accuracy-time trade-off.

Compare in terms of the final RMSE at convergence Finally, it is interesting to compare the performance of HMGP-S in the *US flight* dataset to those of VSSGP and variational PIC, which have been reported to give very good performances in this dataset. Since VSSGP code runs on GPU, and variational PIC uses its own optimizers, it is infeasible to compare their performance-time trade-offs as we previously did with HMGP-S and SVI-S. Therefore, we compare the performances of HMGP-S, SVI-S, VSSGP and variational PIC in the *US flight* dataset in terms of the final RMSE achieved when their optimization processes arrive at convergence or when a maximum number of optimization iterations is reached, whichever is earlier.

The settings for the above methods are as follows. Stochastic optimization is used in all the four methods. We set the batch size to 5000, the number of inducing points/frequencies to 200, and the maximum number of optimization iterations to 10,000. The number of experts for HMGP-S is fixed to 30 in both the *Flight-700K* and *Flight-All* splits. To set the number of blocks for variational PIC, we test it with different numbers of blocks within the available memory and report the best result. This corresponds to 3500 blocks in the *Flight-700K* split and 2000 blocks in the *Flight-All* split.

Table 5 Performances in terms of RMSE of different methods, on the *US flight* data set; the best performances are indicated in bold typeface

	Flight-700K		*Flight-All*	
	RMSE	Training time (h)	RMSE	Training time (h)
HMGP-S	**31.69** (± 0.01)	3.1	**34.61** (± 0.01)	3.4
SVI-S	32.59 (± 0.01)	0.6	35.46 (± 0.02)	0.6
VSSGP	31.88 (± 0.08)	–	34.81 (± 0.04)	–
Variational PIC	32.59 (± 0.09)	3.8	35.06 (± 0.04)	5.2

The numbers in brackets indicate the standard deviation over 5 runs. Training time for each method includes time spent on clustering and model initialization

Table 5 summarizes the performances of HMGP-S, SVI-S, VSSGP and variational PIC on the *US flight* dataset. Again the training time of VSSGP is not available since its Github code runs on GPU. HMGP gives the best performances in terms of RMSE among the four tested methods in both the *Flight-700K* and *Flight-All* splits.

Note that since distributed LMA is not amendable to stochastic optimization, both the *Flight-700K* and *Flight-All* splits are too big for it to fit in the memory of the test computer. However, distributed LMA has been reported in Hoang et al. (2016) to give an impressively low RMSE of 16.5 in a similar setup to the *Flight-700K* split. In that experiment, a distributed system with 32 computing cores and 96 GB memory is used. It is clearly that the targets of distributed LMA and the proposed HMGP method are different. Distributed LMA is designed to handle big data when abundant computational resources (processors and memory) are available. On the other hand, HMGP aims to do GP regression on big datasets using limited computational resources by applying stochastic optimization.

6 Conclusion

In this article, a scalable GP regression method was presented. The proposed method exploits both global and local information from the training data through a two-layer hierarchical model with a sparse global GP in the upper layer and a mixture of sparse GPs in the lower layer. A two-step iterative algorithm was proposed to minimize a variational lower bound of the log marginal likelihood leading to rapid learning of all the hyperparameters of the local and global GPs, the inducing points, and the gating network.

Experimental results were presented which demonstrate the ability of the proposed model to handle non-stationarity and locality in the data as well as to model common trends across the entire data set. Experiments on a wide range of benchmark data sets showed that the proposed model outperformed many state-of-the-art sparse GP regression methods. The method also worked well on large-scale problems using stochastic optimization.

It would be beneficial to allow an unlimited number of layers to be added into the hierarchy to model a more sophisticated structure of data. Although it seems like a natural extension to this research, we leave the detailed implementation and exploration to future studies.

Appendix A The expected likelihood terms

Here we will present the derivation of the expected likelihood terms of the bound (37) in detail. First, we derive the functional forms of the marginal distributions $q(f_k(\mathbf{x}_n))$ for

$k = 0, \ldots, T$. For this, we apply the general properties of the Gaussian distributions which are presented in Equation 2.115 of Bishop (2006) and re-stated below.

Marginal and Conditional Gaussians *Given a marginal Gaussian distribution for* $\mathbf{x}$ *and a conditional Gaussian distribution for* $\mathbf{y}$ *given* $\mathbf{x}$ *in the form*

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$$
$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{1})$$

the marginal distribution of $\mathbf{y}$ *is given by*

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x} = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{1} + \mathbf{A}\boldsymbol{\Lambda}^{1}\mathbf{A}^{\mathrm{T}}). \tag{52}$$

With $q(\mathbf{g}_0)$ and $p(\mathbf{f}_0|\mathbf{g}_0)$ given by Eqs. (35) and (18), we can derive the formula for $q(\mathbf{f}_0) \triangleq \int p(\mathbf{f}_0|\mathbf{g}_0)q(\mathbf{g}_0)d\mathbf{g}_0$ by applying the above property

$$q(\mathbf{f}_0) = \mathcal{N}\left(\mathbf{f}_0|[\mathbf{A}^{(0)}]^{\mathrm{T}}\mathbf{m}_0, \mathbf{K}^{(0)}_{\mathbf{XX}} + [\mathbf{A}^{(0)}]^{\mathrm{T}}\left(\mathbf{S}_0 - \mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right)\mathbf{A}^{(0)}\right), \tag{53}$$

where $\mathbf{A}^{(0)} \equiv [\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}]^{-1}\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{X}}$. As a result, we have

$$\begin{aligned} q(f_0(\mathbf{x}_n)) &= \mathcal{N}\left(f_0(\mathbf{x}_n)|\left[\mathbf{a}^{(0)}_n\right]^{\mathrm{T}}\mathbf{m}_0, \kappa_0(\mathbf{x}_n, \mathbf{x}_n) + \left[\mathbf{a}^{(0)}_n\right]^{\mathrm{T}}\left(\mathbf{S}_0 - \mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right)\mathbf{a}^{(0)}_n\right), \\ &= \mathcal{N}\left(f_0(\mathbf{x}_n)|\left[\mathbf{a}^{(0)}_n\right]^{\mathrm{T}}\mathbf{m}_0, \mathrm{Tr}\left(\mathbf{S}_0\mathbf{a}^{(0)}_n\left[\mathbf{a}^{(0)}_n\right]^{\mathrm{T}}\right) + l^{(0)}_{nn}\right), \end{aligned} \tag{54}$$

where $\mathbf{a}^{(0)}_n$ is a vector of the n-th column of the matrix $\mathbf{A}^{(0)}$ and $l^{(0)}_{nn}$ is the nth diagonal element of the matrix $\mathbf{K}^{(0)}_{\mathbf{XX}} - \mathbf{K}^{(0)}_{\mathbf{XU}_0}[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}]^{-1}\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{X}}$. Similarly, we can derive $q(\mathbf{f}_k) \triangleq \int p(\mathbf{f}_k|\mathbf{h}_k, \mathbf{g}_0)q(\mathbf{h}_k)q(\mathbf{g}_0)d\mathbf{h}_k d\mathbf{g}_0$ for $k = 1, \ldots, T$, with $q(\mathbf{g}_0), q(\mathbf{h}_k)$ and $p(\mathbf{f}_k|\mathbf{h}_k, \mathbf{g}_0)$ given by Eqs. (35), (36) and (23) as follows.

$$\begin{aligned} q(\mathbf{f}_k) &\triangleq \int p(\mathbf{f}_k|\mathbf{h}_k, \mathbf{g}_0)q(\mathbf{h}_k)q(\mathbf{g}_0)d\mathbf{h}_k d\mathbf{g} \\ &= \int \left\{\int p(\mathbf{f}_k|\mathbf{h}_k, \mathbf{g}_0)q(\mathbf{h}_k)d\mathbf{h}_k\right\} q(\mathbf{g}_0)d\mathbf{g} \\ &= \int \left\{\mathcal{N}\left(\mathbf{f}_k|[\mathbf{A}^{(k)}]^{\mathrm{T}}\mathbf{m}_k + [\mathbf{A}^{(0)}]^{\mathrm{T}}\mathbf{g}_0, \mathbf{K}^{(k)}_{\mathbf{XX}} + [\mathbf{A}^{(k)}]^{\mathrm{T}}\left(\mathbf{S}_k - \mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}\right)\mathbf{A}^{(k)}\right)\right\} q(\mathbf{g}_0)d\mathbf{g} \\ &= \mathcal{N}\left(\mathbf{f}_k|[\mathbf{A}^{(k)}]^{\mathrm{T}}\mathbf{m}_k + [\mathbf{A}^{(0)}]^{\mathrm{T}}\mathbf{m}_0, \mathbf{K}^{(k)}_{\mathbf{XX}} + [\mathbf{A}^{(k)}]^{\mathrm{T}}\left(\mathbf{S}_k - \mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}\right)\mathbf{A}^{(k)} + [\mathbf{A}^{(0)}]^{\mathrm{T}}\mathbf{S}_0\mathbf{A}^{(0)}\right), \end{aligned}$$

where $\mathbf{A}^{(k)} \equiv [\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}]^{-1}\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{X}}$. Its marginal is then given by

$$\begin{aligned} q(f_k(\mathbf{x}_n)) = \mathcal{N}\Bigg(&f_k(\mathbf{x}_n)|\left[\mathbf{a}^{(k)}_n\right]^{\mathrm{T}}\mathbf{m}_k + \left[\mathbf{a}^{(0)}_n\right]^{\mathrm{T}}\mathbf{m}_0, \\ &\kappa_k(\mathbf{x}_n, \mathbf{x}_n) + \left[\mathbf{a}^{(k)}_n\right]^{\mathrm{T}}\left(\mathbf{S}_k - \mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}\right)\mathbf{a}^{(k)}_n + \left[\mathbf{a}^{(0)}_n\right]^{\mathrm{T}}\mathbf{S}_0\mathbf{a}^{(0)}_n\Bigg) \\ = \mathcal{N}\Bigg(&f_k(\mathbf{x}_n)|\left[\mathbf{a}^{(k)}_n\right]^{\mathrm{T}}\mathbf{m}_k + \left[\mathbf{a}^{(0)}_n\right]^{\mathrm{T}}\mathbf{m}_0, \mathrm{Tr}\left(\mathbf{S}_k\mathbf{a}^{(k)}_n\left[\mathbf{a}^{(k)}_n\right]^{\mathrm{T}}\right) \\ &+\mathrm{Tr}\left(\mathbf{S}_0\mathbf{a}^{(0)}_n\left[\mathbf{a}^{(0)}_n\right]^{\mathrm{T}}\right) + l^{(k)}_{nn}\Bigg) \end{aligned} \tag{55}$$

where $\mathbf{a}_n^{(k)}$ is a vector of the n-th column of the matrix $\mathbf{A}^{(k)}$ and $l_{nn}^{(k)}$ is the nth diagonal element of the matrix $\mathbf{K}_{\mathbf{XX}}^{(k)} - \mathbf{K}_{\mathbf{XU}_k}^{(k)}[\mathbf{K}_{\mathbf{U}_k\mathbf{U}_k}^{(k)}]^{-1}\mathbf{K}_{\mathbf{U}_k\mathbf{X}}^{(k)}$.

Let μ_{kn} and v_{kn}, respectively, denote the mean and variance of the marginal distribution $q(f_k(\mathbf{x}_n))$ given by Eqs. (54) and (55). The expected likelihood terms from the bound (37) can be calculated as follows

$$\begin{aligned}
&\mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))] \\
&= \mathbb{E}_{q(f_k(\mathbf{x}_n))}\left\{-\frac{1}{2}\ln(2\pi\sigma_k^2) - \frac{1}{2\sigma_k^2}\left[y_n^2 - 2y_n f_k(\mathbf{x}_n) + f_k^2(\mathbf{x}_n)\right]\right\} \\
&= -\frac{1}{2}\ln(2\pi\sigma_k^2) - \frac{1}{2\sigma_k^2}\left\{y_n^2 - 2y_n\mathbb{E}_{q(f_k(\mathbf{x}_n))}\left[f_k(\mathbf{x}_n)\right] + \mathbb{E}_{q(f_k(\mathbf{x}_n))}\left[f_k^2(\mathbf{x}_n)\right]\right\} \\
&= -\frac{1}{2}\ln(2\pi\sigma_k^2) - \frac{1}{2\sigma_k^2}\left[y_n^2 - 2y_n\mu_{kn} + \mu_{kn}^2 + v_{kn}\right] \\
&= \ln\mathcal{N}(y_n|\mu_{kn},\sigma_k^2) - \frac{1}{2\sigma_k^2}v_{kn},
\end{aligned} \tag{56}$$

for $k = 0,\ldots,T$. This results in the following formulations for the expected likelihood terms:

$$\begin{aligned}
\mathbb{E}_{q(f_0(\mathbf{x}_n))}[\ln p(y_n|f_0(\mathbf{x}_n))] = \ln\mathcal{N}\Big(y_n|\Big[\mathbf{a}_n^{(0)}\Big]^{\mathrm{T}}\mathbf{m}_0,\sigma_0^2\Big) \\
-\frac{1}{2\sigma_0^2}\mathrm{Tr}\Big(\mathbf{S}_0\mathbf{a}_n^{(0)}\Big[\mathbf{a}_n^{(0)}\Big]^{\mathrm{T}}\Big) - \frac{1}{2\sigma_0^2}l_{nn}^{(0)}
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))] = \ln\mathcal{N}\Big(y_n|\Big[\mathbf{a}_n^{(k)}\Big]^{\mathrm{T}}\mathbf{m}_k + \Big[\mathbf{a}_n^{(0)}\Big]^{\mathrm{T}}\mathbf{m}_0,\sigma_k^2\Big) \\
-\frac{1}{2\sigma_k^2}\mathrm{Tr}\Big(\mathbf{S}_k\mathbf{a}_n^{(k)}\Big[\mathbf{a}_n^{(k)}\Big]^{\mathrm{T}}\Big) \\
-\frac{1}{2\sigma_k^2}\mathrm{Tr}\Big(\mathbf{S}_0\mathbf{a}_n^{(0)}\Big[\mathbf{a}_n^{(0)}\Big]^{\mathrm{T}}\Big) - \frac{1}{2\sigma_k^2}l_{nn}^{(k)}
\end{aligned}$$

for $k = 1,\ldots,T$.

Appendix B The evidence lower-bound $\mathcal{L}_1$ and its derivatives

We perform optimization over the bound $\mathcal{L}_1$ (given by Eq. (42)) w.r.t. the noise variance $\alpha_k \equiv \sigma_k^2$, the kernel hyper-parameters θ_k, the parameters m_k and $\mathbf{L}_k$ of the variational distributions, and the locations of the inducing points $\mathbf{U}_k$. For that, we need to obtain the partial derivatives of the bound w.r.t. each of the variables to be optimized.

$$\mathcal{L}_1(\mathcal{D}, \boldsymbol{\gamma}) = \sum_{n=1}^{N}\sum_{k=1}^{\mathrm{T}} q(z_n = k)\mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))] + \sum_{n=1}^{N}\mathbb{E}_{q(f_0(\mathbf{x}_n))}[\ln p(y_n|f_0(\mathbf{x}_n))] - \sum_{k=1}^{\mathrm{T}}\mathrm{KL}(q(\mathbf{h}_k)||p(\mathbf{h}_k)) - \mathrm{KL}(q(\mathbf{g}_0)||p(\mathbf{g}_0)),$$

where $\mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))]$ given in Eq. (56) for $k = 0, \ldots, T$,

$$\mathrm{KL}(q(\mathbf{g}_0)||p(\mathbf{g}_0)) = 0.5\left\{\ln\frac{|\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}|}{|\mathbf{S}_0|} - M + \mathrm{Tr}\left(\left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1}\mathbf{S}_0\right) + \mathbf{m}_0^{\mathrm{T}}\left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1}\mathbf{m}_0\right\}$$

and

$$\mathrm{KL}(q(\mathbf{h}_k)||p(\mathbf{h}_k)) = 0.5\left\{\ln\frac{|\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}|}{|\mathbf{S}_k|} - M + \mathrm{Tr}\left(\left[\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}\right]^{-1}\mathbf{S}_k\right) + \mathbf{m}_k^{\mathrm{T}}\left[\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}\right]^{-1}\mathbf{m}_k\right\}.$$

B. 1 Partial derivatives with respect to noise variances

The partial derivatives w.r.t. noise variances are given by

$$\frac{\partial\mathcal{L}_1(\mathcal{D}, \boldsymbol{\gamma})}{\partial\alpha_0} = \sum_{n=1}^{N}\frac{\partial\mathbb{E}_{q(f_0(\mathbf{x}_n))}[\ln p(y_n|f_0(\mathbf{x}_n))]}{\partial\alpha_0} = \sum_{n=1}^{N}\left\{-\frac{1}{2\alpha_0} + \frac{1}{2\alpha_0^2}\left(y_n^2 - 2y_n\mu_{0n} + \mu_{0n}^2 + v_{0n}\right)\right\}$$

and

$$\frac{\partial\mathcal{L}_1(\mathcal{D}, \boldsymbol{\gamma})}{\partial\alpha_k} = \sum_{n=1}^{N}\sum_{k=1}^{\mathrm{T}} q(z_n = k)\frac{\partial\mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))]}{\partial\alpha_k} = \sum_{n=1}^{N}\sum_{k=1}^{\mathrm{T}} q(z_n = k)\left\{-\frac{1}{2\alpha_k} + \frac{1}{2\alpha_k^2}\left(y_n^2 - 2y_n\mu_{kn} + \mu_{kn}^2 + v_{kn}\right)\right\}.$$

B.2 Partial derivatives with respect to other variables

Let r represent a variable to be optimized. The derivative of $\mathcal{L}_1$ w.r.t. r is given by

$$\frac{\partial\mathcal{L}_1(\mathcal{D}, \boldsymbol{\gamma})}{\partial r} = \sum_{n=1}^{N}\sum_{k=1}^{\mathrm{T}} q(z_n = k)\frac{\partial\mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))]}{\partial r} + \sum_{n=1}^{N}\frac{\partial\mathbb{E}_{q(f_0(\mathbf{x}_n))}[\ln p(y_n|f_0(\mathbf{x}_n))]}{\partial r} - \sum_{k=1}^{\mathrm{T}}\frac{\partial\mathrm{KL}(q(\mathbf{h}_k)||p(\mathbf{h}_k))}{\partial r} - \frac{\partial\mathrm{KL}(q(\mathbf{g}_0)||p(\mathbf{g}_0))}{\partial r}. \quad (57)$$

Following the chain rule for multivariate functions, we get:

$$\begin{aligned}\frac{\partial \mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))]}{\partial r} &= \frac{\partial \mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))]}{\partial \mu_{kn}}\frac{\partial \mu_{kn}}{\partial r} \\ &+ \frac{\partial \mathbb{E}_{q(f_k(\mathbf{x}_n))}[\ln p(y_n|f_k(\mathbf{x}_n))]}{\partial v_{kn}}\frac{\partial v_{kn}}{\partial r} \\ &= \frac{1}{\sigma_k^2}(y_n - \mu_{kn})\frac{\partial \mu_{kn}}{\partial r} - \frac{1}{2\sigma_k^2}\frac{\partial v_{kn}}{\partial r},\end{aligned}$$

for $k = 0, \ldots, T$. Hence, to find the derivative in Eq. (57), we just need to find the following partial derivatives: $\frac{\partial \mu_{kn}}{\partial r}$, $\frac{\partial v_{kn}}{\partial r}$, $\frac{\partial \mathrm{KL}(q(\mathbf{h}_k)||p(\mathbf{h}_k))}{\partial r}$ and $\frac{\partial \mathrm{KL}(q(\mathbf{g}_0)||p(\mathbf{g}_0))}{\partial r}$.

B.2.1 Partial derivatives with respect to parameters of the variational distributions

First, we obtain the partial derivatives w.r.t. the variational means $\mathbf{m}_k$ for $k = 0, \ldots, T$. We only present the terms that are non-zero.

The partial derivatives w.r.t. $\mathbf{m}_0$ are given by

$$\begin{aligned}\frac{\partial \mu_{in}}{\partial \mathbf{m}_0} &= \mathbf{a}_n^{(0)}, \text{ for } i = 0, \ldots, T; \\ \frac{\partial \mathrm{KL}(q(\mathbf{g}_0)||p(\mathbf{g}_0))}{\partial \mathbf{m}_0} &= \left[\mathbf{K}_{\mathbf{U}_0\mathbf{U}_0}^{(0)}\right]^{-1}\mathbf{m}_0.\end{aligned}$$

The partial derivatives w.r.t. $\mathbf{m}_k$, for $k = 1, \ldots, T$, are given by

$$\begin{aligned}\frac{\partial \mu_{kn}}{\partial \mathbf{m}_k} &= \mathbf{a}_n^{(k)}; \\ \frac{\partial \mathrm{KL}(q(\mathbf{h}_k)||p(\mathbf{h}_k))}{\partial \mathbf{m}_k} &= \left[\mathbf{K}_{\mathbf{U}_k\mathbf{U}_k}^{(k)}\right]^{-1}\mathbf{m}_k.\end{aligned}$$

Next, we obtain the partial derivatives w.r.t. the variational covariances $\mathbf{S}_k$ for $k = 0, \ldots, T$, which are

$$\begin{aligned}\frac{\partial v_{kn}}{\partial \mathbf{S}_k} &= \mathbf{a}_n^{(k)}\left[\mathbf{a}_n^{(k)}\right]^{\mathrm{T}}; \\ \frac{\partial \mathrm{KL}(q(\mathbf{h}_k)||p(\mathbf{h}_k))}{\partial \mathbf{S}_k} &= \frac{1}{2}\left(-\mathbf{S}_k^{-1} + \left[\mathbf{K}_{\mathbf{U}_k\mathbf{U}_k}^{(k)}\right]^{-1}\right).\end{aligned}$$

The partial derivatives w.r.t. $\mathbf{L}_k$, for $k = 0, \ldots, T$, are then calculated as

$$\frac{\partial \mathcal{L}_1(\mathcal{D}, \boldsymbol{\gamma})}{\partial \mathbf{L}_k} = \frac{\partial \mathcal{L}_1(\mathcal{D}, \boldsymbol{\gamma})}{\partial \mathbf{S}_k}\frac{\partial \mathbf{S}_k}{\partial \mathbf{L}_k} = 2\frac{\partial \mathcal{L}_1(\mathcal{D}, \boldsymbol{\gamma})}{\partial \mathbf{S}_k}\mathbf{L}_k.$$

B.2.2 Partial derivatives with respect to the kernel hyper-parameters and the locations of the inducing points

The bound $\mathcal{L}_1$ depends on the kernel hyper-parameters θ_k and the locations of the inducing points U_k through the three covariance matrices/vectors $\mathbf{K}_{\mathbf{U}_k\mathbf{U}_k}^{(k)}$, $\mathbf{K}_{\mathbf{U}_k\mathbf{X}}^{(k)}$ and $\mathrm{diag}\left[\mathbf{K}_{\mathbf{X}\mathbf{X}}^{(k)}\right]$. We first need to find the partial derivatives of $\mathcal{L}_1$ w.r.t. these matrices.

The partial derivatives w.r.t. entries of $\mathrm{diag}\left[\mathbf{K}_{\mathbf{X}\mathbf{X}}^{(k)}\right]$ for $k = 0, \ldots, T$ are

$$\frac{\partial v_{kn}}{\partial \kappa_k(x_n, x_n)} = 1.$$

The partial derivatives w.r.t. entries of $\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{X}}$ are

$$\frac{\partial \mu_{in}}{\partial \mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{x}_n}} = \left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1}\mathbf{m}_0 \text{ for } i = 0, \ldots, T;$$
$$\frac{\partial v_{0n}}{\partial \mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{x}_n}} = \left(\left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1}\mathbf{S}_0 - \mathbf{I}\right)\left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1}\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{x}_n};$$
$$\frac{\partial v_{in}}{\partial \mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{x}_n}} = \left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1}\mathbf{S}_0\left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1}\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{x}_n} \text{ for } i = 1, \ldots, T.$$

The partial derivatives w.r.t. entries of $\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{X}}$ for $k = 1, \ldots, T$ are

$$\frac{\partial \mu_{kn}}{\partial \mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{x}_n}} = \left[\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}\right]^{-1}\mathbf{m}_k;$$
$$\frac{\partial v_{kn}}{\partial \mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{x}_n}} = \left(\left[\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}\right]^{-1}\mathbf{S}_k - \mathbf{I}\right)\left[\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}\right]^{-1}\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{x}_n}.$$

The partial derivatives w.r.t. entries of $\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}$ are

$$\frac{\partial \mu_{in}}{\partial \mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}} = -\left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1}\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{x}_n}\mathbf{m}_0^{\mathrm{T}}\left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1} \text{ for } i = 0, \ldots, T;$$
$$\frac{\partial v_{0n}}{\partial \mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}} = \mathbf{a}_n^{(0)}\left[\mathbf{a}_n^{(0)}\right]^{\mathrm{T}} - \left(\mathbf{a}_n^{(0)}\left[\mathbf{a}_n^{(0)}\right]^{\mathrm{T}}\mathbf{S}_0\left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1} + \left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1}\mathbf{S}_0\mathbf{a}_n^{(0)}\left[\mathbf{a}_n^{(0)}\right]^{\mathrm{T}}\right);$$
$$\frac{\partial v_{in}}{\partial \mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}} = -\left(\mathbf{a}_n^{(0)}\left[\mathbf{a}_n^{(0)}\right]^{\mathrm{T}}\mathbf{S}_0\left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1} + \left[\mathbf{K}^{(0)}_{\mathbf{U}_0\mathbf{U}_0}\right]^{-1}\mathbf{S}_0\mathbf{a}_n^{(0)}\left[\mathbf{a}_n^{(0)}\right]^{\mathrm{T}}\right) \text{ for } i = 1, \ldots, T.$$

The partial derivatives w.r.t. entries of $\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}$ for $k = 1, \ldots, T$ are

$$\frac{\partial \mu_{kn}}{\partial \mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}} = -\left[\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}\right]^{-1}\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{x}_n}\mathbf{m}_k^{\mathrm{T}}\left[\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}\right]^{-1};$$
$$\frac{\partial v_{kn}}{\partial \mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}} = \mathbf{a}_n^{(k)}\left[\mathbf{a}_n^{(k)}\right]^{\mathrm{T}} - \left(\mathbf{a}_n^{(k)}\left[\mathbf{a}_n^{(k)}\right]^{\mathrm{T}}\mathbf{S}_k\left[\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}\right]^{-1} + \left[\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}\right]^{-1}\mathbf{S}_k\mathbf{a}_n^{(k)}\left[\mathbf{a}_n^{(k)}\right]^{\mathrm{T}}\right).$$

Next, we assume that the squared exponential (SE) kernel with automatic relevance determination (ARD) is used, i.e., the kernel function $\kappa_k(\mathbf{x}, \mathbf{x}')$ is given by

$$\kappa_k(\mathbf{x}, \mathbf{x}') = \beta_k \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^{\mathrm{T}}\mathbf{M}_k(\mathbf{x} - \mathbf{x}')\right), \tag{58}$$

where β_k is the signal variance and $\mathbf{M}_k = \mathrm{diag}([\phi_1^{(k)}, \ldots, \phi_D^{(k)}])$ with $\phi_d^{(k)} = [\ell_1^{(k)}]^{-2}$ and $\ell_d^{(k)}$ being the characteristic length-scale for input dimension d, for $d = 1, \ldots, D$. The partial derivative w.r.t to any variable r among the kernel hyper-parameters $\theta_k = (\beta_k, \phi_1^{(k)}, \ldots, \phi_D^{(k)})$ or locations of the inducing points $\mathbf{U}_k$ can then be calculated as

$$\frac{\partial \mathcal{L}_1(\mathcal{D}, \boldsymbol{\gamma})}{r} = \frac{\partial \mathcal{L}_1(\mathcal{D}, \boldsymbol{\gamma})}{\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}}\frac{\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}}{r} + \frac{\partial \mathcal{L}_1(\mathcal{D}, \boldsymbol{\gamma})}{\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{X}}}\frac{\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{X}}}{r} + \frac{\partial \mathcal{L}_1(\mathcal{D}, \boldsymbol{\gamma})}{\mathrm{diag}\left[\mathbf{K}^{(k)}_{\mathbf{X}\mathbf{X}}\right]}\frac{\mathrm{diag}\left[\mathbf{K}^{(k)}_{\mathbf{X}\mathbf{X}}\right]}{r},$$

for $k = 0, \ldots, T$. We now find the partial derivatives of the covariance matrices $\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{U}_k}$, $\mathbf{K}^{(k)}_{\mathbf{U}_k\mathbf{X}}$ and $\text{diag}\left[\mathbf{K}^{(k)}_{\mathbf{XX}}\right]$ w.r.t. the kernel hyper-parameters and locations of the inducing points.

Since $\frac{\partial \kappa_k(\mathbf{x},\mathbf{x}')}{\partial \beta_k} = \frac{\kappa_k(\mathbf{x},\mathbf{x}')}{\beta_k}$, the partial derivatives of any matrix $\mathbf{K}^{(k)}$ w.r.t. the signal variances β_k is

$$\frac{\partial \mathbf{K}^{(k)}}{\partial \beta_k} = \frac{\mathbf{K}^{(k)}}{\beta_k}.$$

The partial derivative of any matrix entry $\kappa_k(\mathbf{x}, \mathbf{x}')$ w.r.t. $\phi^{(k)}_d$ is

$$\frac{\partial \kappa_k(\mathbf{x}, \mathbf{x}')}{\partial \phi^{(k)}_d} = \kappa_k(\mathbf{x}, \mathbf{x}')(\mathbf{x}_d - \mathbf{x}'_d)^2.$$

Let $\mathbf{U}^{(k)}_i$ denote the i-th inducing input of $\mathbf{U}_k$ for $k = 0, \ldots, T$, and $\mathbf{U}^{(k)}_{id}$ denote its d-th dimension. The partial derivative $\frac{\partial \mathbf{K}_{\mathbf{U}_k\mathbf{U}_k}}{\partial \mathbf{U}^{(k)}_{id}}$ is given by

$$\left(\frac{\partial \mathbf{K}_{\mathbf{U}_k\mathbf{U}_k}}{\partial \mathbf{U}^{(k)}_{id}}\right)_{mm'} = \frac{\partial \kappa(\mathbf{U}^{(k)}_m, \mathbf{U}^{(k)}_{m'})}{\partial \mathbf{U}^{(k)}_{id}} = (\delta_{mi} - \delta_{m'i})\kappa(\mathbf{U}^{(k)}_m, \mathbf{U}^{(k)}_{m'})(-\phi^{(k)}_d)(\mathbf{U}^{(k)}_{md} - \mathbf{U}^{(k)}_{m'd}).$$

The partial derivative $\frac{\partial \mathbf{K}_{\mathbf{U}_k\mathbf{X}}}{\partial \mathbf{U}^{(k)}_{id}}$ is given by

$$\left(\frac{\partial \mathbf{K}_{\mathbf{U}_k\mathbf{X}}}{\partial \mathbf{U}^{(k)}_{id}}\right)_{mn} = \frac{\partial \kappa(\mathbf{U}^{(k)}_m, \mathbf{x}_n)}{\partial \mathbf{U}^{(k)}_{id}} = \delta_{mi}\kappa(\mathbf{U}^{(k)}_m, \mathbf{x}_n)(-\phi^{(k)}_d)(\mathbf{U}^{(k)}_{md} - \mathbf{x}_{nd}).$$

References

Bayer, J., Osendorfer, C., Diot-Girard, S., Rueckstiess, T., & Urban, S. (2015). *Climin-a pythonic framework for gradient-based function optimization*. Technical report of TUM.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., & Lamere, P. (2011). The million song dataset. In *Proceedings of 12th international conference on music information retrieval*, pp. 591–596.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.

Boyle, P., & Frean, M. (2005). Dependent Gaussian processes. *Advances in Neural Information Processing Systems*, *17*, 217–224.

Chalupka, K., Williams, C. K., & Murray, I. (2013). A framework for evaluating approximation methods for Gaussian process regression. *Journal of Machine Learning Research*, *14*(1), 333–350.

Csató, L. (2002). Gaussian processes: Iterative sparse approximations. PhD thesis, Aston University.

Gal, Y., & Turner, R. (2015). Improving the Gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. *Advances in Neural Information Processing Systems*, *27*, 3257–3265.

Gal, Y., van der Wilk, M., & Rasmussen, C. E. (2014). Distributed variational inference in sparse Gaussian process regression and latent variable models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 27* (pp. 3257–3265). Curran Associates, Inc.

Goldberg, P. W., Williams, C. K., & Bishop, C. M. (1998). Regression with input-dependent noise: A Gaussian process treatment. *Advances in Neural Information Processing Systems*, *10*, 493–499.

GPy. (since 2012). *GPy: A Gaussian process framework in python*. http://github.com/SheffieldML/GPy. Accessed 08 July 2016.

Gramacy, R. B., Lee, H. K., & Macready, W. G. (2004). Parameter space exploration with Gaussian process trees. In *Proceedings of 21st international conference on machine learning*, pp. 353–360.

Hensman, J., Fusi, N., Lawrence, N. D. (2013). Gaussian processes for big data. In *Proceedings of 29th conference on uncertainty in artificial intelligence*, pp. 282–290.

Hensman, J., Matthews, A., & Ghahramani, Z. (2015). Scalable variational Gaussian process classification. In *Proceedings of 8th international conference on artificial intelligence and statistics*, pp. 351–360.
Hoang, T. N., Hoang, Q. M., Low, B. K. H. (2015). A unifying framework of anytime sparse Gaussian process regression models with stochastic variational inference for big data. In *Proceedings of 32nd international conference on machine learning*, pp. 569–578.
Hoang, T. N., Hoang, Q. M., & Low, B. K. H. (2016). A distributed variational inference framework for unifying parallel sparse Gaussian process regression models. In *Proceedings of 33rd international conference on machine learning*, pp. 382–391.
Hoffman, M. D., Blei, D. M., Wang, C., & Paisley, J. W. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, *14*(1), 1303–1347.
Lawrence, N., Seeger, M., & Herbrich, R. (2003). Fast sparse Gaussian process methods: The informative vector machine. *Advances in Neural Information Processing Systems*, *15*, 625–632.
Low, K. H., Yu, J., Chen, J., & Jaillet, P. (2015). Parallel Gaussian process regression for big data: Low-rank representation meets Markov approximation. In *Proceedings of 29th AAAI conference on artificial intelligence*, pp. 2821–2827.
Meeds, E., & Osindero, S. (2006). An alternative infinite mixture of Gaussian process experts. *Advances in Neural Information Processing Systems*, *18*, 883–890.
Nguyen, T., & Bonilla, E. (2014). Fast allocation of Gaussian process experts. In *Proceedings of 31st international conference on machine learning*, pp. 145–153.
Nguyen, T., Bouzerdoum, A., & Phung, S. (2016). Variational inference for infinite mixtures of sparse Gaussian processes through KL-correction. In *Proceedings of 41st IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 2579–2583.
O'Hagan, A., & Kingman, J. (1978). Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society Series B (Methodological)*, *40*(1), 1–42.
Park, S., & Choi, S. (2010). Hierarchical Gaussian process regression. In *Proceedings 2nd Asian conference on machine learning*, pp. 95–110.
Quiñonero-Candela, J., & Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, *6*, 1939–1959.
Rasmussen, C. E., & Ghahramani, Z. (2002). Infinite mixtures of Gaussian process experts. *Advances in Neural Information Processing Systems*, *14*, 881–888.
Seeger, M. (2003). *Bayesian Gaussian process models: Pac-Bayesian generalisation error bounds and sparse approximations*. PhD thesis, University of Edinburgh.
Seeger, M., Williams, C., & Lawrence, N. (2003). Fast forward selection to speed up sparse Gaussian process regression. In *Proceedings 9th international workshop on artificial intelligence and statistics*, pp. 3–6.
Shi, J. Q., Murray-Smith, R., & Titterington, D. (2003). Bayesian regression and classification using mixtures of Gaussian processes. *Intern Journal of Adaptive Control and Signal Processing*, *17*(2), 149–161.
Shi, J. Q., Murray-Smith, R., & Titterington, D. (2005). Hierarchical Gaussian process mixtures for regression. *Statistics and Computing*, *15*(1), 31–41.
Silverman, B. W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical Society Series B (Methodological)*, *47*(1), 1–52.
Smola, A. J., & Bartlett, P. L. (2001). Sparse greedy Gaussian process regression. *Advances in Neural Information Processing Systems*, *13*, 619–625.
Snelson, E., & Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems*, *18*, 1257–1264.
Snelson, E., & Ghahramani, Z. (2007). Local and global sparse Gaussian process approximations. In *Proceedings of 11th international conference on artificial intelligence and statistics*, pp. 524–531.
Snelson, E. L. (2008). *Flexible and efficient Gaussian process models for machine learning*. PhD thesis, University of London, University College London, UK.
Sollich, P., & Williams, C. (2005). Using the equivalent kernel to understand Gaussian process regression. *Advances in Neural Information Processing Systems*, *17*, 1313–1320.
Sun, S., & Xu, X. (2011). Variational inference for infinite mixtures of Gaussian processes with applications to traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, *12*(2), 466–475.
Titsias, M. K. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of 12th international conference on artificial intelligence and statistics*, pp. 567–574.
Tresp, V. (2000a). A Bayesian committee machine. *Neural Computation*, *12*(11), 2719–2741.
Tresp, V. (2000b). Mixtures of Gaussian processes. *Advances in Neural Information Processing Systems*, *13*, 654–660.
Williams, C. K., & Rasmussen, C. E. (2006a). *Gaussian processes for machine learning* (Vol. 2, No. 3, p. 4). Cambridge: The MIT Press.

Williams, C. K., & Rasmussen, C. E. (2006b). *Gaussian processes for machine learning*. Cambridge, MA: The MIT Press.

Williams, C. K. I., & Rasmussen, C. E. (1996). Gaussian processes for regression. *Advances in Neural Information Processing Systems*, *8*, 514–520.

Yuan, C., & Neubauer, C. (2009). Variational mixture of Gaussian process experts. *Advances in Neural Information Processing Systems*, *21*, 1897–1904.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.

Zhu, C., Byrd, R. H., Lu, P., & Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, *23*(4), 550–560.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.