An Online Bayesian Filtering Framework for Gaussian Process Regression: Application to Global Surface Temperature Analysis

Yali Wang

Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China

Email: yl.wang@siat.ac.cn

Brahim Chaib-draa

Department of Computer Science and Software Engineering, Laval University, Canada

Email: chaib@ift.ulaval.ca

Abstract

Over the past centuries, global warming has gradually become one of the most significant issues in our life. Hence, it is crucial to analyze global surface temperature with an efficient and accurate model. Gaussian process (GP) is a popular nonparametric model, due to the power of Bayesian inference framework. However, the performance of GP is often deteriorated for large-scale data sets such as global surface temperature. In this work, we propose a novel online Bayesian filtering framework for large-scale GP regression. There are three contributions. Firstly, we develop a novel GP-based state space model to efficiently process data in a sequential manner. Secondly, based on our state space model, we design a marginalized particle filter to infer the latent function values and learn the model parameters online. It can efficiently reduce the computation burden of GP while improving the estimation accuracy in a recursive Bayesian inference framework. Finally, we successfully apply our approach to a number of synthetic data sets and the large-scale global surface temperature data set. The results show that our approach outperforms related GP variants, and it is an efficient and accurate expert system for global surface temperature analysis.

Keywords: Gaussian Process Regression; Marginalized Particle Filter;

Online Parameter Learning; Global Surface Temperature Analysis.

1. Introduction

Over the past centuries, global warming has gradually become one of the most significant issues in our life. The unusual changes of global surface temperature has far-reaching effects on our planet, such as amplifying the frequency of extreme weathers, putting pressure on ecosystems, altering the ranges of infectious diseases. Hence, it is crucial to analyze global surface temperature with an efficient and accurate model, in order to help us reduce the negative influence of global warming.

Bayesian parametric models have been well-developed over the past years (Bishop, 2006; Barber, 2012). However, these models often restrict the richness of the assumed function family. As a result, Bayesian nonparametric methods have become popular recently by giving a prior probability to every possible function (Bishop, 2006; Rasmussen and Williams, 2006). In particular, Gaussian process (GP) (Rasmussen and Williams, 2006), a powerful nonparametric model with an elegant Bayesian inference framework, has been successfully used in a number of real-world applications in geosciences (Paciorek and Schervish, 2003; Hensman et al., 2013; Silversides et al., 2016). However, the performance of GP is often deteriorated for large-scale data sets such as global surface temperature (Quinonero-Candela and Rasmussen, 2005; Chalupka et al., 2013; Gal et al., 2014).

1.1. Gaussian Process Variants for Large-scale Data

The computation complexity of GP is $O(N^3)$, where N is the size of training set. For large-scale data sets, GP is often computationally intractable (Rasmussen and Williams, 2006). To reduce computation, several GP variants have been proposed and a detailed review can be found in (Quinonero-Candela and Rasmussen, 2005; Chalupka et al., 2013; Low et al., 2015; Bauer et al., 2016). In this paper, we mainly discuss several related approaches, according to different mechanisms for computation reduction.

One well-known mechanism is sparse approximation such as sparse pseudo-input Gaussian process (SPGP) (Snelson and Ghahramani, 2005) and sparse spectrum Gaussian process (SSGP) (Lázaro-Gredilla et al., 2010). In SPGP, sparsification is achieved by using a small number of learned pseudo-inputs to parameterize the covariance matrix of training set. However, its accuracy is often limited, especially when the size of the pseudo set is getting

larger and/or the dimension of the input space is getting higher (Snelson and Ghahramani, 2005). In SSGP, sparsification is achieved by learning a stationary trigonometric Bayesian model. But similar to SPGP, SSGP may trap into overfitting as optimization is implemented over a large number of model parameters. Another important mechanism is online-learning GP variants (Csató and Opper, 2002; Nguyen-Tuong et al., 2008; Vaerenbergh et al., 2012; Reece and Roberts, 2010) which process the large-scale training set in a sequential manner to maintain computation efficiency. A well-known method is the recent Kalman filter Gaussian process (KFGP) (Reece and Roberts, 2010). By taking advantage of the connection between GP and Kalman filter, KFGP reduces computation by correlating GP of different training subsets in a recursive framework. However, the model parameters in KFGP is learned off-line. As a result, KFGP may trap into a poor local optimum. Additionally, several distributed implementations of GP have been recently developed with variational inference (Gal et al., 2014; Dai et al., 2016) and product-of-experts (Deisenroth and Ng, 2015), in order to speed up computation. However, as before, these approaches are susceptible to local optima and hard to optimize (Bauer et al., 2016).

1.2. Our Contributions

Different from GP variants above, we propose an online Bayesian filtering framework for GP regression to deal with large data sets (such as global surface temperature). There are three contributions in this work. Firstly, we develop a novel GP-based state space model, which allows to efficiently process the small data collections sequentially. Secondly, we propose a marginalized particle filter to simultaneously learn the model parameters and latent function values in an online fashion. In this case, our approach can learn a more reasonable parameter vector, by using a number of weighted particles to approximate the marginal posterior (instead of the marginal likelihood in the previous fast GP approaches). Furthermore, the recursive Bayesian filtering framework can incorporate the previous estimation to improve the current performance, hence our proposed approach can alleviate computation of GP regression while improving accuracy. Finally, we show that our approach is an efficient and accurate GP regression framework on synthetic data sets and the large-scale global surface temperature data set.

The rest of this paper is organized as follows. In Section 2, we briefly review GP and how to use it for regression. In Section 3, we introduce our novel GP-based state space model. In Section 4, we design our marginalized

particle filter to learn the model parameters and latent function values in an online manner. In Section 5, we evaluate our proposed approach on synthetic data sets and the global surface temperature data set. Finally, we conclude the paper in Section 6.

2. Background

In this section, we first introduce the definition of Gaussian process (GP). Then, we review how to use GP to address a standard regression task.

2.1. Definition of Gaussian Process

Gaussian process (GP) is *a collection of random variables, any finite number of which have a joint Gaussian distribution* (Rasmussen and Williams, 2006). It has been widely used as a Bayesian prior over the latent function, where the function values at any finite number of inputs are Gaussian-distributed random variables (Rasmussen and Williams, 2006; MacKay, 1998; Bishop, 2006; Barber, 2012).

Specifically, a GP prior over the latent function $f(\mathbf{x})$ is denoted by

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{1}$$

where the mean function is

$$m(\mathbf{x}) \;=\; E[f(\mathbf{x})], \tag{2}$$

the covariance function is

$$k(\mathbf{x}, \mathbf{x}') \;=\; E[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \tag{3}$$

and $\mathbf{x}$, $\mathbf{x}' \in \mathbb{R}^{d_x}$ are any two d_x-dimension input vectors.

When $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, the prior over the latent function values $f(X) = [f(\mathbf{x}_1), \cdots, f(\mathbf{x}_N)]^T$ at any N input vectors $X = [\mathbf{x}_1, \cdots, \mathbf{x}_N]^T$ is jointly Gaussian (Rasmussen and Williams, 2006),

$$p(f(X)) = \mathcal{N}(\mathbf{m}(X), K(X, X)), \tag{4}$$

where the mean vector $\mathbf{m}(X)$ is computed from the mean function $m(\mathbf{x})$,

$$\mathbf{m}(X) = \begin{bmatrix} m(\mathbf{x}_1) \\ \cdots \\ m(\mathbf{x}_N) \end{bmatrix}, \tag{5}$$

and the covariance matrix $K(X, X)$ is computed from the covariance function $k(\mathbf{x}, \mathbf{x}')$,

$$K(X, X) = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \cdots & \cdots & \cdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}. \tag{6}$$

In this work, we follow (Rasmussen and Williams, 2006; Lawrence, 2005) to choose a widely-used GP prior,

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}) = 0, k(\mathbf{x}, \mathbf{x}')), \tag{7}$$

where the mean function is zero for simplicity [1]. Next, we illustrate how to use GP to address a standard nonlinear regression task from the Bayesian view.

2.2. Gaussian Process Regression

Suppose that we are given a training set D with N input-output data pairs, $D = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, where the input is $\mathbf{x}_n \in \mathbb{R}^{d_x}$, the output is $y_n \in \mathbb{R}$. Each output is assumed to be generated from

$$y = f(\mathbf{x}) + \epsilon_y, \tag{8}$$

where the noise is Gaussian, $\epsilon_y \sim \mathcal{N}(0, \sigma_y^2)$ with variance σ_y^2; the latent function has a GP prior, $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$. For convenience, we denote $X = [\mathbf{x}_1, ..., \mathbf{x}_N]^T$, $\mathbf{y} = [y_1, ..., y_N]^T$, and collect the parameters of $k(\mathbf{x}, \mathbf{x}')$ and the noise variance σ_y^2 into a parameter vector θ.

The goal of regression is to learn the model parameters θ and predict the test outputs $\mathbf{y}_\star = [y_\star^1, ..., y_\star^M]^T$ at the test inputs $X_\star = [\mathbf{x}_\star^1, ..., \mathbf{x}_\star^M]^T$, by using the training set $D = (X, \mathbf{y})$. Within GP, this problem can be elegantly interpreted to learn a predictive distribution over θ and $\mathbf{y}_\star$,

$$p(\theta, \mathbf{y}_\star | X_\star, X, \mathbf{y}) = p(\mathbf{y}_\star | X_\star, X, \mathbf{y}, \theta) p(\theta | X, \mathbf{y}). \tag{9}$$

As $p(\theta|X, \mathbf{y}) \propto p(\mathbf{y}|X, \theta)$, a popular approach to learn θ is to minimize the negative log likelihood with the gradient optimization (Rasmussen and

[1] Note that it is straightforward to choose other mean functions to do mathematical derivations without difficulties.

Williams, 2006)

$$
\begin{aligned}
&-\log p(\mathbf{y}|X,\theta) \\
=&\frac{1}{2}\mathbf{y}^T[K(X,X)+\sigma_y^2 I]^{-1}\mathbf{y}+\frac{1}{2}\log|K(X,X)+\sigma_y^2 I|+\frac{n}{2}\log 2\pi, \qquad (10)
\end{aligned}
$$

where $K(X,X)$ is constructed by using Eq. (6).

After learning θ, one can make prediction by $p(\mathbf{y}_\star|X_\star,X,\mathbf{y},\theta)$. **Firstly**, since the latent function has a GP prior, the joint distribution over the latent function values at the training inputs $f(X)=[f(\mathbf{x}^1),...,f(\mathbf{x}^N)]^T$ and the latent function values at the test inputs $f(X_\star)=[f(\mathbf{x}_\star^1),...,f(\mathbf{x}_\star^M)]^T$ is the following Gaussian distribution (Rasmussen and Williams, 2006),

$$
p(f(X),f(X_\star)|X,X_\star,\theta)=\mathcal{N}(\mathbf{0},\begin{bmatrix}K(X,X) & K(X,X_\star)\\ K(X_\star,X) & K(X_\star,X_\star)\end{bmatrix}), \qquad (11)
$$

where $K(X_\star,X)$, $K(X_\star,X_\star)$ are constructed by using $X_\star$ and X in Eq. (6), and $K(X,X_\star)=K(X_\star,X)^T$.

Secondly, based on the Gaussian distribution in Eq. (11) and the Gaussian noise in Eq. (8), one can obtain that the joint distribution over the training outputs $\mathbf{y}$ and latent function values at test inputs $f(X_\star)$ is Gaussian (Rasmussen and Williams, 2006),

$$
p(\mathbf{y},f(X_\star)|X,X_\star,\theta)=\mathcal{N}(\mathbf{0},\begin{bmatrix}K(X,X)+\sigma_y^2 I & K(X,X_\star)\\ K(X_\star,X) & K(X_\star,X_\star)\end{bmatrix}). \qquad (12)
$$

Thirdly, since the joint distribution $p(\mathbf{y},f(X_\star)|X,X_\star,\theta)$ in Eq. (12) is Gaussian, its corresponding conditional distribution $p(f(X_\star)|X_\star,X,\mathbf{y},\theta)$ is also Gaussian (Rasmussen and Williams, 2006),

$$
p(f(X_\star)|X_\star,X,\mathbf{y},\theta)=\mathcal{N}(\mu_\star,\Sigma_\star), \qquad (13)
$$

where the mean vector $\mu_\star$ and the covariance matrix $\Sigma_\star$ are computed from Eq. (12),

$$
\mu_\star=K(X_\star,X)[K(X,X)+\sigma_y^2 I]^{-1}\mathbf{y}, \qquad (14)
$$

$$
\Sigma_\star=K(X_\star,X_\star)-K(X_\star,X)[K(X,X)+\sigma_y^2 I]^{-1}K(X_\star,X)^T. \qquad (15)
$$

Finally, based on the Gaussian distribution in Eq. (13) and the Gaussian noise in Eq. (8), the target distribution $p(\mathbf{y}_\star|X_\star,X,\mathbf{y},\theta)$ is Gaussian,

$$
p(\mathbf{y}_\star|X_\star,X,\mathbf{y},\theta)=\mathcal{N}(\mu_\star,\Sigma_\star+\sigma_y^2 I). \qquad (16)
$$

Note that, the computation complexity of GP regression is mainly governed by the covariance matrix inversion of N training pairs in Eq. (10), (14), (15). This results in $\mathcal{O}(N^3)$ which is computationally expensive when N is beyond a few thousand (Rasmussen and Williams, 2006). In the following, we propose a novel online Bayesian filtering framework for GP regression in order to derive a computationally tractable GP model while preserving the estimation accuracy.

3. State Space Model for GP Regression

In practice, the whole training set is often constructed by gathering small subsets several times. We assume that the training subset at the t-th collection consists of n_t input-output pairs, i.e., $\{(\mathbf{x}_t^1, y_t^1), \cdots (\mathbf{x}_t^{n_t}, y_t^{n_t})\}$. For simplicity, we denote $X_t = [\mathbf{x}_t^1, \cdots \mathbf{x}_t^{n_t}]^T$, $\mathbf{y}_t = [y_t^1, \cdots y_t^{n_t}]^T$, and denote the whole training set with T collections as $(X_{1:T}, \mathbf{y}_{1:T})$. Inspired by this data collection procedure, we propose a novel GP-based state space model (SSM) to efficiently process data in an online manner.

3.1. Prediction Model

In general, SSM consists of a prediction model (also called transition model or motion model) and an observation model. We first construct the prediction model which reflects the evolution of latent states. In the context of GP, the latent states are the model parameter vector θ and the latent function values.

For the model parameter vector θ, we propose the following dynamics as its prediction model,

$$\theta_t = b\theta_{t-1} + (1-b)\bar{\theta}_{t-1} + s_{t-1}, \tag{17}$$

where $b = (3\delta - 1)/(2\delta)$ and δ is a discount factor (typically 0.95-0.99), $\bar{\theta}_{t-1}$ is the Monte Carlo mean of θ at $t-1$, $s_{t-1} \sim \mathcal{N}(0, r^2\Sigma_{t-1})$ is the noise of rand walk in which $r^2 = 1 - b^2$ and Σ_{t-1} is the Monte Carlo variance matrix of θ at $t-1$. Due to the fact that Eq. (17) takes advantage of kernel smoothing in the Bayesian filtering, it can guarantee the estimation convergence (Liu and West, 2001; Li et al., 2004; Kantas et al., 2009).

For the latent function values, we use GP to explore a novel correlation between the $(t-1)_{th}$ and t_{th} data subsets to construct the corresponding prediction model. For simplicity, we denote $X_t^c = \{X_t, X_\star\}$ and $f_t^c = f(X_t^c)$.

Since the latent function has a GP prior, i.e., $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$, the joint distribution $p(f_t^c, f_{t-1}^c | X_{t-1}^c, X_t^c, \theta_t)$ is Gaussian,

$$p(f_t^c, f_{t-1}^c | X_{t-1}^c, X_t^c, \theta_t) = \mathcal{N}(\mathbf{0}, \begin{bmatrix} K(X_t^c, X_t^c) & K(X_t^c, X_{t-1}^c) \\ K(X_t^c, X_{t-1}^c)^T & K(X_{t-1}^c, X_{t-1}^c) \end{bmatrix}). \quad (18)$$

Furthermore, because the joint distribution in Eq. (18) is Gaussian, its conditional distribution $p(f_t^c | f_{t-1}^c, X_{t-1}^c, X_t^c, \theta_t)$ is also Gaussian,

$$p(f_t^c | f_{t-1}^c, X_{t-1}^c, X_t^c, \theta_t) = \mathcal{N}(G(\theta_t) f_{t-1}^c, Q(\theta_t)), \quad (19)$$

where

$$G(\theta_t) = K(X_t^c, X_{t-1}^c) K^{-1}(X_{t-1}^c, X_{t-1}^c), \quad (20)$$

$$Q(\theta_t) = K(X_t^c, X_t^c) - K(X_t^c, X_{t-1}^c) K^{-1}(X_{t-1}^c, X_{t-1}^c) K(X_t^c, X_{t-1}^c)^T. \quad (21)$$

According to Bayesian filtering (Liu and West, 2001; Li et al., 2004; Kantas et al., 2009), the conditional distribution in Eq. (19) can be transformed to the following prediction model for latent function values, which is a linear transition with an additive Gaussian noise $v_t^f \sim \mathcal{N}(\mathbf{0}, Q(\theta_t))$,

$$f_t^c = G(\theta_t) f_{t-1}^c + v_t^f. \quad (22)$$

3.2. Observation Model

The observation model reflects the relation between the latent states and the corresponding observations. In our case, it can be straightforwardly obtained from Eq. (8) with the t-th data collection $(X_t, \mathbf{y}_t)$,

$$\mathbf{y}_t = H_t f_t^c + v_t^y, \quad (23)$$

where $H_t = [I_{n_t} \ \ \mathbf{0}]$ is an index matrix to make $H_t f_t^c = f(X_t)$, due to the fact that the t-th training outputs $\mathbf{y}_t$ are only corresponding to the t-th training inputs X_t. Additionally, the noise $v_t^y \sim \mathcal{N}(0, R(\theta_t))$ is from the noise in Eq. (8), where the noise covariance is $R(\theta_t) = \sigma_{y,t}^2 I$. Note that σ_y is a static, unknown parameter. We use $\sigma_{y,t}$ in order to keep consistency with the artificial evolution of θ in Eq. (17).

To sum up, our GP-based SSM is fully defined by the prediction model in Eq. (17) and (22), the observation model in Eq. (23). Based on this novel SSM, we propose an online Bayesian filtering framework in the following section to learn latent function values and unknown model parameters efficiently in a recursive fashion.

4. Bayesian Inference by Marginalized Particle Filter

In contrast to the standard GP regression with an off-line inference procedure, we propose a Bayesian filtering framework to simultaneously estimate hidden function values and learn the model parameters in an online manner. According to our SSM, the GP regression task can be transformed to infer the following posterior recursively,

$$p(f_t^c, \theta_{1:t} | X_{1:t}, X_\star, \mathbf{y}_{1:t}). \tag{24}$$

However, this posterior is analytically intractable. One of the most popular approximation techniques is Monte Carlo sampling (Doucet et al., 2001; Cappé et al., 2007). Since our SSM is constructed on a sequential data collection procedure, we investigate sequential Monte Carlo sampling approaches (i.e., particle filtering) for Bayesian inference.

Furthermore, Eq. (22) in our SSM is a linear structure given θ_t. The traditional sampling importance resampling particle filter can introduce the unnecessary computational burden during sampling (Doucet et al., 2001; Cappé et al., 2007). This fact inspires us to apply the marginalized particle filter (i.e., Rao-Blackwellised particle filter) for inference, where the latent states in the conditionally-linear structure can be efficiently estimated by Kalman filter (Li et al., 2004; de Freitas, 2002; Schön et al., 2005).

Specifically, we use Bayes rule to factorize the posterior in Eq. (24) as follows,

$$p(f_t^c, \theta_{1:t} | X_{1:t}, X_\star, \mathbf{y}_{1:t}) = p(\theta_{1:t} | X_{1:t}, X_\star, \mathbf{y}_{1:t}) p(f_t^c | \theta_{1:t}, X_{1:t}, X_\star, \mathbf{y}_{1:t}). \tag{25}$$

As the first term $p(\theta_{1:t} | X_{1:t}, X_\star, \mathbf{y}_{1:t})$ is analytically intractable, we approximate it by using a number of particles. After the particle approximation of $\theta_{1:t}$ is obtained, we can perform Kalman filter to compute the second term $p(f_t^c | \theta_{1:t}, X_{1:t}, X_\star, \mathbf{y}_{1:t})$. This is because f_t^c is the latent state in the conditionally-linear structure (Eq. (22)) of our GP-based SSM.

4.1. How to Compute $p(\theta_{1:t} | X_{1:t}, X_\star, \mathbf{y}_{1:t})$

Since $p(\theta_{1:t} | X_{1:t}, X_\star, \mathbf{y}_{1:t})$ is not analytically tractable, we factorize it in the following recursive form and approximate it by a sequential importance sampling mechanism,

$$\begin{aligned} &p(\theta_{1:t} | X_{1:t}, X_\star, \mathbf{y}_{1:t}) \\ \propto &p(\mathbf{y}_t | \mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star) p(\theta_t | \theta_{t-1}) p(\theta_{1:t-1} | X_{1:t-1}, X_\star, \mathbf{y}_{1:t-1}). \end{aligned} \tag{26}$$

At the t-th sampling iteration, the particles for θ_t are firstly drawn from the proposal $p(\theta_t|\theta_{t-1})$ which is easily obtained from Eq. (17). Then the importance weight for each particle at t can be computed by $p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)$. This distribution can be solved analytically,

$$\begin{aligned}
&p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star) \\
=&\int p(\mathbf{y}_t|f_t^c, \theta_t, X_t, X_\star) p(f_t^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star) df_t^c \\
=&\int \mathcal{N}(H_t f_t^c, R(\theta_t)) \mathcal{N}(f_{t|t-1}^c, P_{t|t-1}^c) df_t^c \\
=&\mathcal{N}(H_t f_{t|t-1}^c, H_t P_{t|t-1}^c H_t^T + R(\theta_t)).
\end{aligned} \tag{27}$$

where $p(\mathbf{y}_t|f_t^c, \theta_t, X_t, X_\star) = \mathcal{N}(H_t f_t^c, R(\theta_t))$ is a Gaussian distribution due to our Gaussian observation model in Eq. (23). $p(f_t^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star) = \mathcal{N}(f_{t|t-1}^c, P_{t|t-1}^c)$ is also a Gaussian distribution with the predictive mean $f_{t|t-1}^c$ and covariance $P_{t|t-1}^c$, because this distribution is the prediction step of Kalman filter for f_t^c.

4.2. How to Compute $p(f_t^c|\theta_{1:t}, X_{1:t}, X_\star, \mathbf{y}_{1:t})$

This posterior can be recursively obtained by using Kalman filter, since Eq. (22) and (23) are linear and Gaussian given $\theta_{1:t}$. Specifically, we use Bayes rule to factorize it as follows,

$$p(f_t^c|\theta_{1:t}, X_{1:t}, X_\star, \mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|f_t^c, \theta_t, X_t, X_\star) p(f_t^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)}. \tag{28}$$

The prediction distribution $p(f_t^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)$ can be computed by

$$\begin{aligned}
&p(f_t^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star) \\
=&\int p(f_t^c|f_{t-1}^c, \theta_t, X_{t-1:t}, X_\star) p(f_{t-1}^c|\mathbf{y}_{1:t-1}, \theta_{1:t-1}, X_{1:t-1}, X_\star) df_{t-1}^c \\
=&\int \mathcal{N}(G(\theta_t) f_{t-1}^c, Q(\theta_t)) \mathcal{N}(f_{t-1|t-1}^c, P_{t-1|t-1}^c) df_{t-1}^c \\
=&\mathcal{N}(G(\theta_t) f_{t-1|t-1}^c, \quad G(\theta_t) P_{t-1|t-1}^c G(\theta_t)^T + Q(\theta_t)),
\end{aligned} \tag{29}$$

where $p(f_t^c|f_{t-1}^c, \theta_t, X_{t-1:t}, X_\star)$ is the prediction model of our GP-based SSM in Eq. (19), and $p(f_{t-1}^c|\mathbf{y}_{1:t-1}, \theta_{1:t-1}, X_{1:t-1}, X_\star) = \mathcal{N}(f_{t-1|t-1}^c, P_{t-1|t-1}^c)$ is the posterior estimation for f_{t-1}^c. Note that, $p(f_t^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)$ is

$\mathcal{N}(f^c_{t|t-1}, P^c_{t|t-1})$, as mentioned in Eq. (27). Hence, the prediction step of Kalman filter is summarized by

$$f^c_{t|t-1} = G(\theta_t) f^c_{t-1|t-1}, \tag{30}$$

$$P^c_{t|t-1} = G(\theta_t) P^c_{t-1|t-1} G(\theta_t)^T + Q(\theta_t). \tag{31}$$

Next, we put Eq. (23), (27) and (29) into Eq. (28). We can obtain that the posterior $p(f^c_t|\theta_{1:t}, X_{1:t}, X_\star, \mathbf{y}_{1:t})$ is a Gaussian distribution, where the update step of Kalman filter is summarized with Kalman Gain Γ_t,

$$\Gamma_t = P^c_{t|t-1} H_t^T (H_t P^c_{t|t-1} H_t^T + R(\theta_t))^{-1}, \tag{32}$$

$$f^c_{t|t} = f^c_{t|t-1} + \Gamma_t(\mathbf{y}_t - H_t f^c_{t|t-1}), \tag{33}$$

$$P^c_{t|t} = P^c_{t|t-1} - \Gamma_t H_t P^c_{t|t-1}. \tag{34}$$

To sum up, we show the whole sampling procedure of our Marginalized Particle filter for GP regression (MPGP) in Alg. 1. At each iteration, we first draw N_p particles for the model parameter vector (Alg. 1, Line 4), and we use these particles to calculate the computation quantities of our SSM (Alg. 1, Line 5). Then, we perform Kalman filter for latent function values (Alg. 1, Line 6-7). Next, we compute the weights of particles and make the weighted estimation for the model parameters and latent function values (Alg. 1, Line 8-11). Finally, we resample the particles according to the normalized weights for the next step (Alg. 1, Line 12).

Note that, the computational cost of our MPGP is mainly governed by $\mathrm{O}(N_p T S^3)$ (Kantas et al., 2009), where N_p is the number of the particles, T is the number of data collections, $S = \max(n_t)$, $(t = 1, 2, ...T)$ is the max size among all the data subsets. Compared to the standard GP regression, our MPGP naturally cut down the computational load by sequentially processing a number of small data subsets. Moreover, Bayesian filtering framework allows us to propagate the previous estimation efficiently to improve the current accuracy. Therefore, our MPGP can accelerate computation while preserving accuracy for large data sets.

Additionally, it is worth mentioning that, Kalman filter GP (KFGP) (Reece and Roberts, 2010) can be treated as a special case of our MPGP. In KFGP, the model parameter vector is firstly trained off-line. Given the learned parameter vector, the posterior $p(f^c_t|\theta_{1:t}, X_{1:t}, X_\star, \mathbf{y}_{1:t})$ is then estimated by Kalman filter. However, this off-line parameter learning procedure in KFGP will either take a long time using a large extra training set or fall

Algorithm 1 Our marginalized particle filter for GP regression (MPGP).

for $t = 1$ to T **do**

——————— Importance Sampling ———————

for $i = 1$ to N_p **do**

Drawing $\theta_t^i \sim p(\theta_t|\tilde{\theta}_{t-1}^i)$ by using the resampled particle $\tilde{\theta}_{t-1}^i$ in (17)

Using θ_t^i to construct $G(\theta_t^i)$, $Q(\theta_t^i)$, $R(\theta_t^i)$ in (20), (21), (23)

Kalman Predict: Using $\tilde{f}_{t-1|t-1}^{c,i}$, $\tilde{P}_{t-1|t-1}^{c,i}$ into (30-31) to compute $f_{t|t-1}^{c,i}$, $P_{t|t-1}^{c,i}$

Kalman Update: Using $f_{t|t-1}^{c,i}$ and $P_{t|t-1}^{c,i}$ into (32-34) to compute $f_{t|t}^{c,i}$ and $P_{t|t}^{c,i}$

Putting $f_{t|t-1}^{c,i}$, $P_{t|t-1}^{c,i}$, $R(\theta_t^i)$ into (27) to compute the importance weight $\bar{w}_t^i$

end for

Normalizing the weight: $w_t^i = \bar{w}_t^i / (\sum_{i=1}^{N_p} \bar{w}_t^i) \quad (i = 1, ..., N_p)$

——————— Weighted Estimation ———————

$$\hat{\theta}_t = \sum_{i=1}^{N_p} w_t^i \theta_t^i$$

$$\hat{f}_{t|t}^c = \sum_{i=1}^{N_p} w_t^i f_{t|t}^{c,i} \Rightarrow \hat{f}_{t|t}^\star = H_t^\star \hat{f}_{t|t}^c$$

$$\hat{P}_{t|t}^c = \sum_{i=1}^{N_p} w_t^i (P_{t|t}^{c,i} + (f_{t|t}^{c,i} - \hat{f}_{t|t}^c)(f_{t|t}^{c,i} - \hat{f}_{t|t}^c)^T) \Rightarrow \hat{P}_{t|t}^\star = H_t^\star \hat{P}_{t|t}^c (H_t^\star)^T$$

where $H_t^\star = [\mathbf{0} \quad I_m]$ is an index matrix to get the estimation of the latent function values $f(X_\star)$ at the test inputs $X_\star$

——————— Resampling ———————

Resampling $\theta_t^i, f_{t|t}^{c,i}, P_{t|t}^{c,i}$ with respect to the importance weight w_t^i, in order to obtain $\tilde{\theta}_t^i, \tilde{f}_{t|t}^{c,i}, \tilde{P}_{t|t}^{c,i}$ for the next step

end for

into an unsatisfactory local optimum using a small extra training set. On the contrary, the model parameter vector and latent function values in our MPGP are learned online by using the marginalized particle filter, which can improve estimation accuracy within its Bayesian filtering framework.

5. Experiments

In this section, we evaluate the performance of our MPGP on two synthetic data sets and a real-world global surface temperature data set. Additionally, we apply a popular zero-mean GP prior, $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$, where we choose a compounded covariance function $k_{SENN}(\mathbf{x}, \mathbf{x}')$ to deal with the non-stationary phenomena which often exist in the global surface

temperature data set,

$$k_{SENN}(\mathbf{x}, \mathbf{x}') = k_{SE}(\mathbf{x}, \mathbf{x}') + k_{NN}(\mathbf{x}, \mathbf{x}'). \tag{35}$$

$k_{SE}(\mathbf{x}, \mathbf{x}')$ is a stationary squared exponential covariance function,

$$k_{SE}(\mathbf{x}, \mathbf{x}') = a_1^2 \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^T(\mathbf{x} - \mathbf{x}')}{2a_2^2}\right). \tag{36}$$

$k_{NN}(\mathbf{x}, \mathbf{x}')$ is a non-stationary neural network covariance function,

$$k_{NN}(\mathbf{x}, \mathbf{x}') = a_3^2 \sin^{-1}\left(\frac{\tilde{\mathbf{x}}^T\tilde{\mathbf{x}}'}{a_4^2\sqrt{(1 + \frac{1}{a_4^2}\tilde{\mathbf{x}}^T\tilde{\mathbf{x}})(1 + \frac{1}{a_4^2}\tilde{\mathbf{x}}'^T\tilde{\mathbf{x}}')}}\right), \tag{37}$$

where the augmented input $\tilde{\mathbf{x}}$ in $k_{NN}(\mathbf{x}, \mathbf{x}')$ is $[1; \mathbf{x}]$. As before, we collect the parameters of $k_{SENN}(\mathbf{x}, \mathbf{x}')$ and the noise variance σ_y^2 of Eq. (8) into a parameter vector θ,

$$\theta = [\sigma_y \quad a_1 \quad a_2 \quad a_3 \quad a_4]^T. \tag{38}$$

5.1. Two Synthetic Data Sets

We first evaluate our MPGP on two benchmark synthetic data sets. One is a sharp-peak function which contains spatially-inhomogeneous smoothness (DiMatteo et al., 2001),

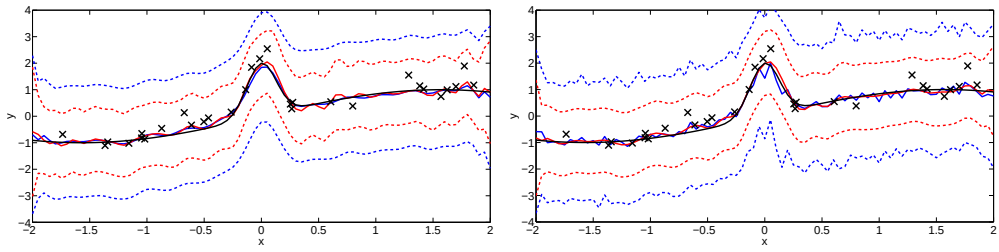
$$f_1(x) = \sin(x) + 2\exp(-30x^2). \tag{39}$$

For $f_1(x)$, we gather the entire training data set with 100 collections. For each collection, we randomly select 30 inputs from [-2, 2] and then calculate their outputs by adding a Gaussian noise $\mathcal{N}(0, 0.3^2)$ to their function values. The test input is from -2 to 2 with 0.05 interval.

The other function is with a discontinuity (Wood, 2002),

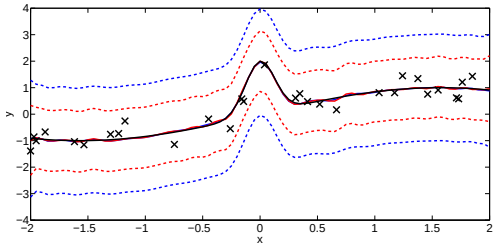
$$f_2(x) = \begin{cases} \mathcal{N}(x; 0.6, 0.2^2) + \mathcal{N}(x; 0.15, 0.05^2) & (0 \le x \le 0.3), \\ \mathcal{N}(x; 0.6, 0.2^2) + \mathcal{N}(x; 0.15, 0.05^2) + 4 & (0.3 < x \le 1). \end{cases} \tag{40}$$

For $f_2(x)$, we gather the entire training data set with 50 collections. For each collection, we randomly select 60 inputs from [0, 1] and then calculate their outputs by adding a Gaussian noise $\mathcal{N}(0, 0.8^2)$ to their function values. The test input is from 0 to 1 with a 0.02 interval.

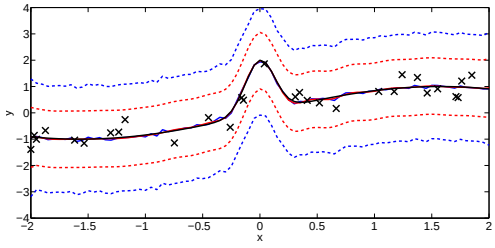


(a) Estimation (SE) of f_1 at t=10

(b) Estimation (SENN) of f_1 at t=10



(c) Estimation (SE) of f_1 at t=100



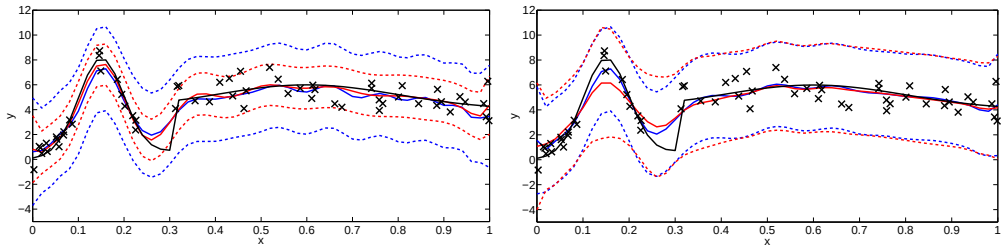
(d) Estimation (SENN) of f_1 at t=100

Figure 1: Estimation comparison for f_1. For all plots, the estimation of KFGP/our MPGP are represented by blue/red line (mean) with blue/red dashed interval (mean $\pm$ $2\times$ standard deviation), the observations are represented by black crosses, the true $f_1(X_\star)$ is represented by black line. The 1st/2nd row show the estimation at the $t = 10/100$ data collection. The 1st/2nd column show the estimation with SE/SENN kernel.

In the first experiment, we evaluate our MPGP in comparison with the closely-related KFGP (Reece and Roberts, 2010). For simplicity, we denote SE-KFGP and SENN-KFGP as KFGP with the squared exponential covariance function k_{SE} and KFGP with the compounded covariance function $k_{SE} + k_{NN}$ respectively. Similarly, SE-MPGP and SENN-MPGP are our MPGP with k_{SE} and MPGP with $k_{SE} + k_{NN}$. The number of particles in our MPGP is set to 10.

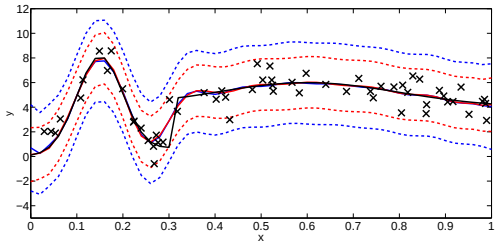
First, we show the estimation comparison in Fig. 1-2. We can see that, both KFGP and our MPGP are getting better and tend to converge when the number of data collections increases. This is mainly credited to the fact that Bayesian filtering allows to incorporate the previous estimation into the current estimation to improve performance. However, compared to KFGP, our MPGP reduces the estimation uncertainty (i.e., a more compact confidence interval than KFGP) because of our online parameter learning.

Second, we evaluate the prediction accuracy when the number of data

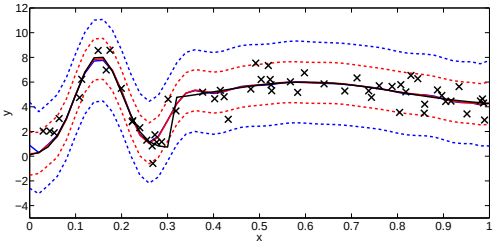


(a) Estimation (SE) of f_2 at t=10

(b) Estimation (SENN) of f_2 at t=10



(c) Estimation (SE) of f_2 at t=50

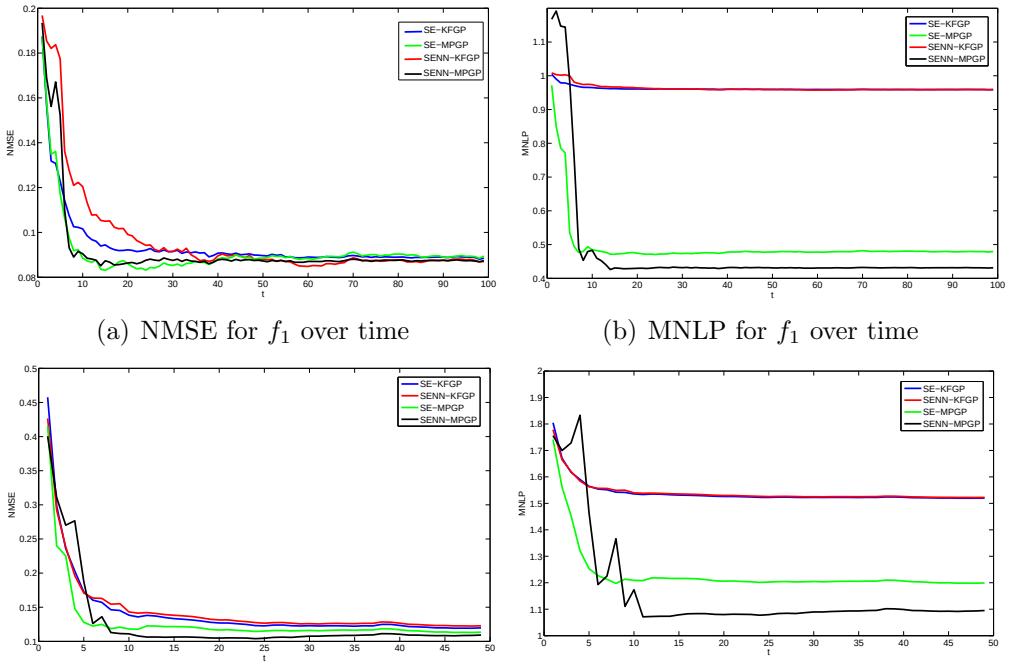


(d) Estimation (SENN) of f_2 at t=50

Figure 2: Estimation comparison for f_2. For all plots, the estimation of KFGP/our MPGP are represented by blue/red line (mean) with blue/red dashed interval (mean $\pm$ 2$\times$ standard deviation), the observations are represented by black crosses, the true $f_2(X_\star)$ is represented by black line. The 1st/2nd row show the estimation at the $t = 10/50$ data collection. The 1st/2nd column show the estimation with SE/SENN kernel.

collections increases. The evaluation criterion is the test Normalized Mean Square Error (NMSE) and the test Mean Negative Log Probability (MNLP), as suggested in (Lázaro-Gredilla et al., 2010). In Fig. 3, the accuracy of MPGP for both f_1 and f_2 are generally better than KFGP when the number of data collections increases, especially for MNLP which penalizes both uncertainty and inconsistency of estimation. This is mainly due to the online parameter learning of our MPGP. Furthermore, our SENN-MPGP outperforms our SE-MPGP, as SENN-MPGP takes the spatial non-stationary phenomenon into account.

In the second experiment, we illustrate the average accuracy and efficiency of our SE-MPGP and SENN-MPGP when the number of particles increases. For each value of the number of particles, we run our SE-MPGP and SENN-MPGP five times and show the average NMSE, MNLP and running time in Fig. 4. As expected, NMSE and MNLP of SE-MPGP and SENN-MPGP decrease while the running time increases, when we increase the number of



(a) NMSE for f_1 over time

(b) MNLP for f_1 over time

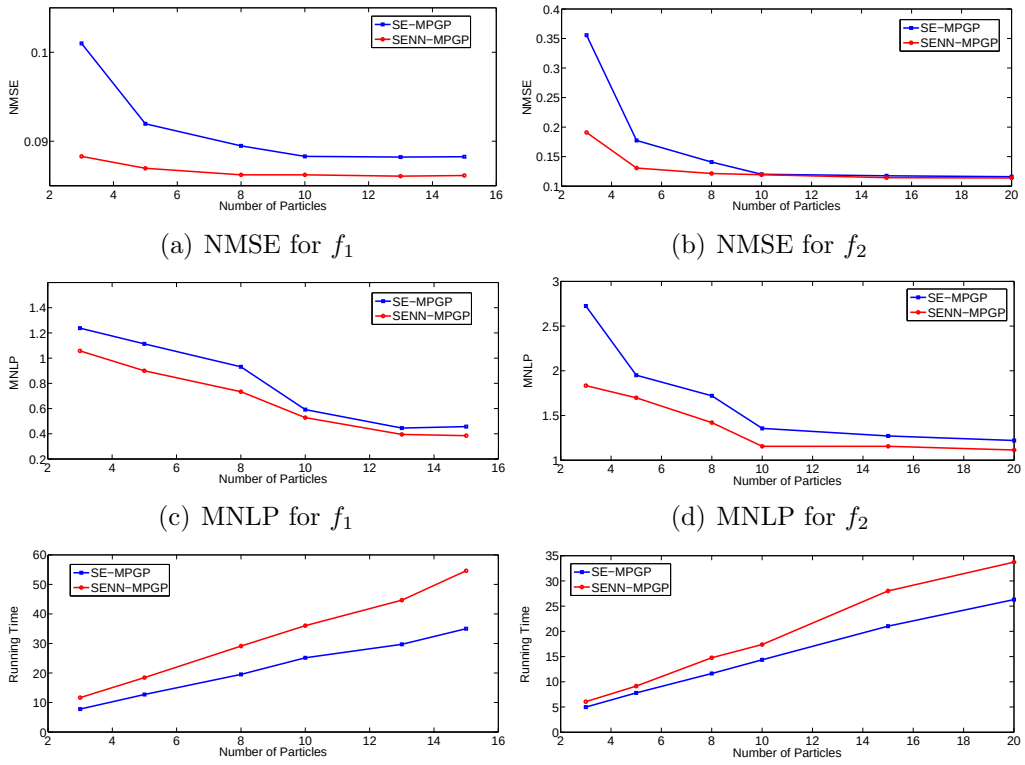
(c) NMSE for f_2 over time

(d) MNLP for f_2 over time

Figure 3: Accuracy evaluation for f_1/f_2 over time.

particles. Furthermore, the average accuracy of SENN-MPGP is better than SE-MPGP, since it can capture the spatial non-stationarity with its SENN covariance function. But SENN-MPGP needs more computation than SE-MPGP, because the dimension of the parameter vector in SENN-MPGP is larger than SE-MPGP.

In the third experiment, we compare our MPGP with other state-of-the-art fast GP approaches. The benchmark methods we chose for comparison are sparse pseudo-input Gaussian process (SPGP) (Snelson and Ghahramani, 2005) and sparse spectrum Gaussian process (SSGP) (Lázaro-Gredilla et al., 2010). To take the tradeoff between accuracy and computation efficiency into account, we implemented SPGP with five pseudo inputs, SSGP with ten basis functions, SE-MPGP with five particles, SENN-MPGP with five particles. Moreover, due to the fact that the chosen training data subsets are sequentially ordered, we would like to examine the robustness of our MPGP with regards to the order of the training data subsets, i.e., we should clarify



(a) NMSE for f_1 (b) NMSE for f_2

(c) MNLP for f_1 (d) MNLP for f_2

(e) Running Time for f_1 (f) Running Time for f_2

Figure 4: Accuracy and efficiency for f_1/f_2 as a function of the number of particles.

whether the good estimation of our MPGP heavily depends on the order of training data collections. Hence we randomly permute the order of training subsets (we used before) in this experiment. In Table 1, our SE-MPGP mainly outperforms SPGP except that its MNLP1 is worse than the one of SPGP. The reason is the synthetic functions are non-stationary but SE-MPGP uses a stationary SE kernel. Hence, we perform our non-stationary SENN-MPGP to show that our MPGP is competitive with SSGP, and much more accurate and computationally efficient than SPGP.

Table 1: Benchmarks comparison for synthetic data sets. NMSEi, MNLPi, RTi represent NMSE, MNLP and the running time for f_i $(i = 1, 2)$.

Methods	NMSE1	MNLP1	RT1	NMSE2	MNLP2	RT2
SPGP	0.224	0.54	28.6s	0.54	1.60	30.6s
SSGP	0.089	0.16	18.9s	0.11	1.12	10.2s
SE-MPGP	0.088	1.63	12.6s	0.17	1.35	12.5s
SENN-MPGP	0.088	0.18	18.8s	0.13	1.18	11.6s

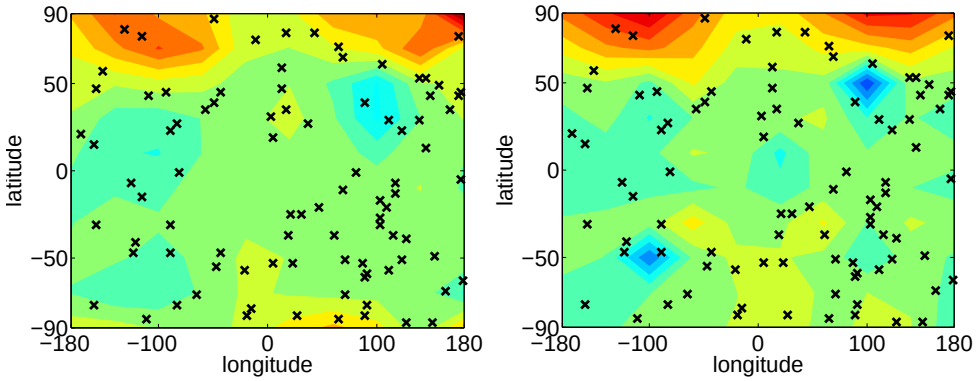
5.2. Global Surface Temperature Data Set

We now perform our MPGP on the large-scale and spatially non-stationary global surface temperature data set[2]. In this experiment, we first gather the training data set with 100 collections. For each collection, we randomly select 90 data points, where the input vector is the longitude and latitude location, the output is the temperature (^oC). There are two test data sets for different goals of our evaluation. The first test set is a gridded input set (Longitude: -180:40:180, Latitude: -90:20:90). It is used to show the estimated global surface temperature. The second test set (100 data points) is randomly selected from the data website after we obtain all the training data collections. This test set is used to evaluate accuracy and efficiency of our MPGP.

In the first experiment, we show the predicted temperature surface at the gridded test inputs. We choose the number of particles in our SE-MPGP and SENN-MPGP as 20. From Fig. 5, we can see that two KFGP methods stuck in the local optimum, i.e., SE-KFGP seems to be underfitting since it does not model the cold region around the location (100, 50), SENN-KFGP seems to be overfitting since it unexpectedly models the cold region around (-100, -50). On the contrary, SE-MPGP and SENN-MPGP fit the temperature data set suitably, due to online parameter learning (Fig. 6).

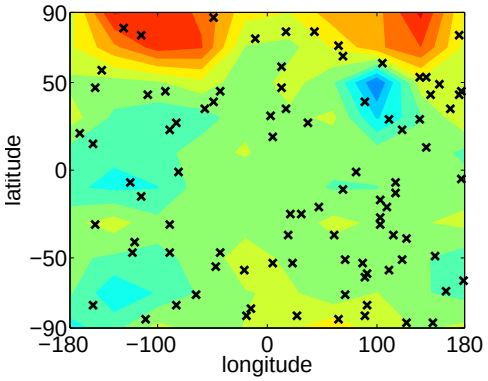
In the second experiment, we evaluate accuracy and efficiency of our MPGP using the second test set. Firstly, we run SE-KFGP, SENN-KFGP, our SE-MPGP and our SENN-MPGP, as the number of training data collections increase. In Fig. 7, NMSE and MNLP for all the methods decrease when the number of the training data collections increases. This is because that

[2]The data set is available at http://data.giss.nasa.gov/gistemp/.

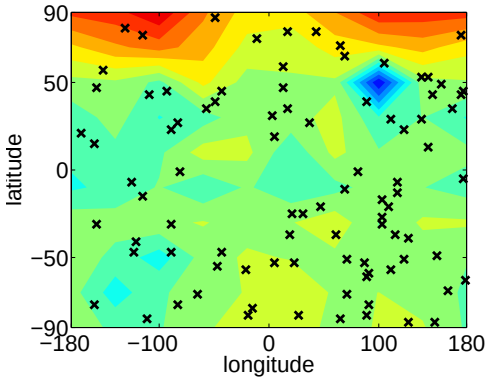


(a) SE-KFGP

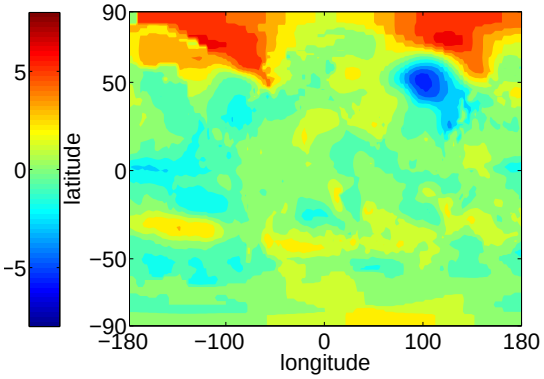
(b) SENN-KFGP



(c) Our SE-MPGP

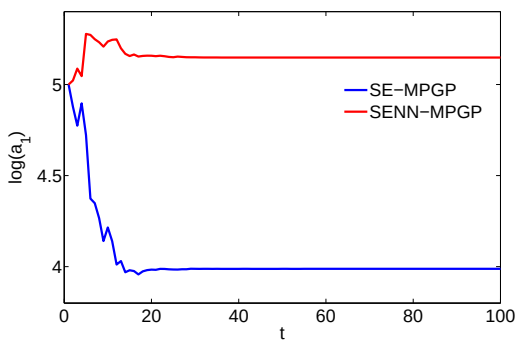


(d) Our SENN-MPGP

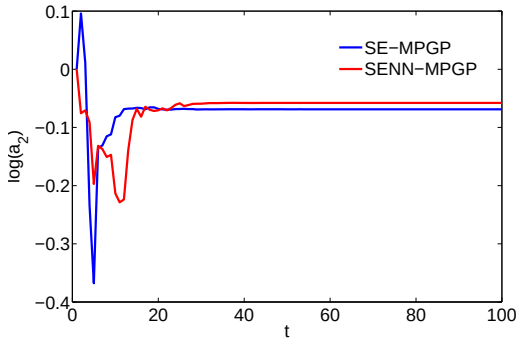


(e) Ground Truth

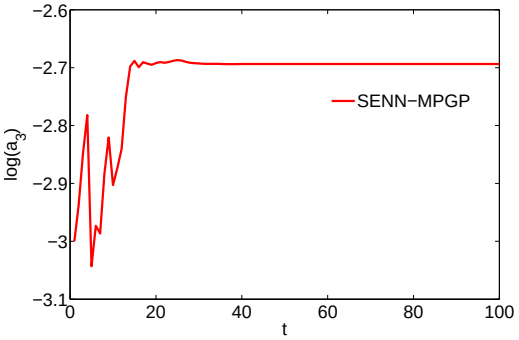
Figure 5: The temperature estimation at $t = 100$. The black crosses are the 100^{th} collection of the training data points.



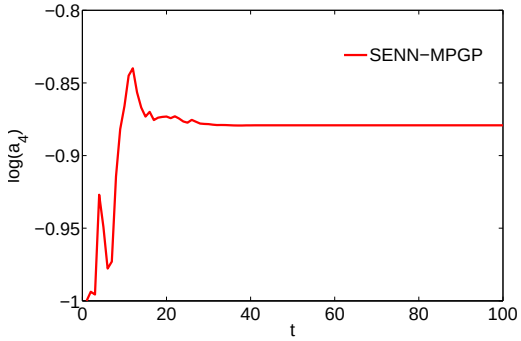
(a) Estimation of $\log(a_1)$ over time



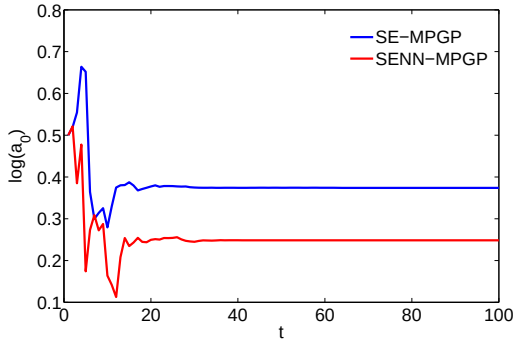
(b) Estimation of $\log(a_2)$ over time



(c) Estimation of $\log(a_3)$ over time



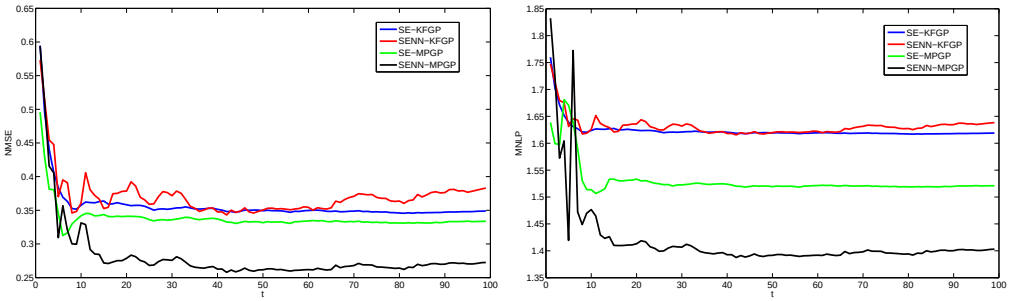
(d) Estimation of $\log(a_4)$ over time



(e) Estimation of $\log(\sigma_y)$ over time

Figure 6: Online parameter learning for the temperature data set. Note that, the covariance function in SE-MPGP is $k_{SE}(\mathbf{x}, \mathbf{x}')$. Hence there are no a_3 and a_4 for SE-MPGP.

Bayesian filtering is a recursive mechanism to improve performance. Furthermore, NMSE and MNLP of our MPGP are much lower than KFGP, due



(a) NMSE over time (temperature data) (b) MNLP over time (temperature data)

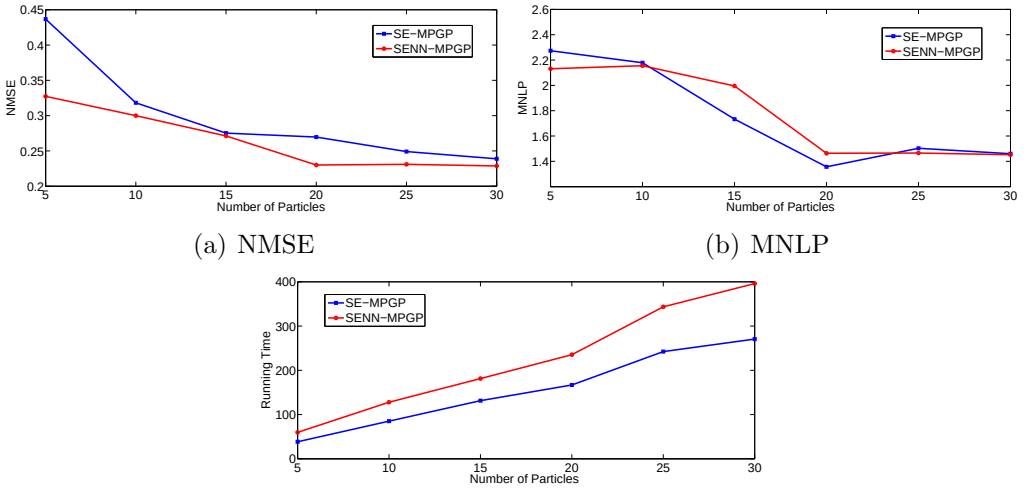
Figure 7: Accuracy evaluation over time (temperature data).

to online parameter learning of our MPGP. Additionally, Our SENN-MPGP is more accurate than SE-MPGP, and this shows that our SENN-MPGP successfully models non-stationarity of this temperature data by using the compounded non-stationary covariance function. Finally, we evaluate the performance of our MPGP as a function of the number of particles. For each value of the number of particles, we run SE-MPGP and SENN-MPGP three times to evaluate the average NMSE, MNLP and running time. In Figure 8, our SENN-MPGP fits the data better than SE-MPGP but the trade-off is the longer running time. This is mainly credited to the fact that the non-stationary covariance function of our SENN-MPGP can capture the spatial non-stationarity to improve accuracy, while this covariance contains more parameters than the SE covariance function in our SE-MPGP. Hence the running time of our SENN-GP is slightly longer than our SE-MPGP.

In the third experiment, we compare our MPGP with fast GP approaches. All denotations are the same as the third experiment of synthetic data. We also randomly interrupt the order of training subsets for the robustness consideration. In Table 2, our MPGP outperforms SPGP and SSGP with a shorter running time and a better accuracy. This illustrates that our MPGP is an efficient and accurate GP regression approach for large data sets such as global surface temperature.

6. Conclusion and Future Work

In this work, we have proposed a Bayesian filtering framework for large-scale GP regression and successfully applied it to global surface temperature



(a) NMSE

(b) MNLP

(c) Running Time

Figure 8: Accuracy and efficiency (temperature data) as a function of the number of particles.

analysis. From the theoretical perspective, our approach provides an efficient and accurate learning mechanism for large-scale GP regression. It outperforms other related GP approaches by using the powerful marginalized particle filter. Furthermore, different from the previous GP approaches, online learning in our method is a novel venue for parameter optimization in GP. From the practical perspective, our approach can predict global surface temperature with high accuracy and efficiency, which illustrates that it can be used as a reasonable and effective expert system for climate analysis.

In the future, it would be interesting to take outlier modeling into our approach, due to the fact that the Gaussian noise may not be theoretically robust to the outlier observations. The student-t noise is a popular choice for outlier modeling (Vanhatalo et al., 2009, 2013; Solin and Särkkä, 2015), which can be applied in our sampling mechanism. From the practical perspective, we would like to use our proposed framework as the expert system for other real-life applications such as robotics (Plagemann, 2008), computer vision (Yao et al., 2011; Wang et al., 2016) and finance (Park et al., 2014; Wang et al., 2016).

Table 2: Benchmarks comparison for the temperature data set.

Method	NMSE	MNLP	RTime
SPGP	0.48	1.62	181.3s
SSGP	0.27	1.33	97.2s
SE-MPGP	0.11	1.05	51.0s
SENN-MPGP	0.10	1.16	59.3s

Acknowledgments

This work has been supported by Natural Sciences and Engineering Research Council of Canada (NSERC), National Natural Science Foundation of China (61502470), Shenzhen Research Program (JCYJ20160229193541167).

Reference

Barber, D., 2012. Bayesian Reasoning and Machine Learning. Cambridge University Press.

Bauer, M. S., van der Wilk, M., Rasmussen, C. E., 2016. Understanding probabilistic sparse gaussian process approximations. In: NIPS.

Bishop, C. M., 2006. Pattern Recognition and Machine Learning. Springer.

Cappé, O., Godsill, S. J., Moulines, E., 2007. An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo. Proceedings of the IEEE 95, 899–924.

Chalupka, K., Williams, C. K. I., Murray, I., 2013. A Framework for Evaluating Approximation Methods for Gaussian Process Regression. Journal of Machine Learning Research 14, 333–350.

Csató, L., Opper, M., 2002. Sparse Online Gaussian Processes. Neural Computation 14 (3), 641–668.

Dai, Z., Damianou, A., González, J., Lawrence, N., 2016. Variational auto-encoded deep gaussian processes. In: ICLR.

de Freitas, N., 2002. Rao-Blackwellised particle filtering for fault diagnosis. In: IEEE Aerospace Conference Proceedings. pp. 1767–1772.

Deisenroth, M. P., Ng, J. W., 2015. Distributed gaussian processes. In: ICML.

DiMatteo, I., Genovese, C. R., Kass, R. E., 2001. Bayesian Curve Fitting with Free-Knot Splines. Biometrika 88, 1055–1071.

Doucet, A., de Freitas, N., Gordon, N., 2001. Sequential Monte Carlo Methods in Practice. Springer.

Gal, Y., van der Wilk, M., Rasmussen, C., 2014. Distributed variational inference in sparse gaussian process regression and latent variable models. In: Neural Information Processing Systems (NIPS'14) Conference on. pp. 3257–3265.

Hensman, J., Fusi, N., Lawrence, N. D., 2013. Gaussian processes for big data. arXiv Preprint cs.LG.
URL `http://arxiv.org/abs/1309.6835`

Kantas, N., Doucet, A., Singh, S. S., Maciejowski, J. M., 2009. An overview of squential Monte Carlo methods for parameter estimation in general state space models. In: 15 th IFAC Symp. on System Identification. pp. 774–785.

Lawrence, N., 2005. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. Journal of Machine Learning Research 6, 1783–1816.

Lázaro-Gredilla, M., Quinonero-Candela, J., Rasmussen, C. E., Figueiras-Vidal, A. R., 2010. Sparse Spectrum Gaussian Process Regression. Journal of Machine Learning Research 11, 1865–1881.

Li, P., Goodall, R., Kadirkamanathan, V., 2004. Estimation of parameters in a linear state space model using a Rao-Blackwellised particle filter. IEEE Proceedings on Control Theory and Applications 151, 727–738.

Liu, J., West, M., 2001. Combined Parameter and State Estimation in Simulation-Based Filtering. In: Sequential Monte Carlo Methods in Practice. pp. 197–223.

Low, K. H., Chen, J., Hoang, T. N., Xu, N., Jaillet, P., 2015. Recent advances in scaling up gaussian process predictive models for large spatiotemporal data. In: Dynamic Data-Driven Environmental Systems Science. Springer, pp. 167–181.

MacKay, D. J. C., 1998. Introduction to Gaussian Processes. In: Neural Networks and Machine Learning. pp. 133–165.

Nguyen-Tuong, D., Peters, J., Seeger, M., 2008. Local Gaussian Process Regression for Real Time Online Model Learning and Control. In: Neural Information Processing Systems (NIPS). pp. 1193–1200.

Paciorek, C. J., Schervish, M. J., 2003. Nonstationary Covariance Functions for Gaussian Process Regression. In: Neural Information Processing Systems (NIPS). pp. 273–280.

Park, H., Kim, N., Lee, J., 2014. Parametric models and non-parametric machine learning models for predicting option prices: Empirical comparison study over kospi 200 index options. Expert Systems with Applications 41, 52275237.

Plagemann, C., 2008. Gaussian Processes for Flexible Robot Learning. Ph.D. thesis, Albert-Ludwigs-Universität Freiburg.

Quinonero-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., Williams, C. K. I., 2006. Gaussian Processes for Machine Learning. The MIT Press.

Reece, S., Roberts, S., 2010. An Introduction to Gaussian Processes for the Kalman Filter Expert. In: International Conference on Information Fusion (FUSION). pp. 1–9.

Schön, T., Gustafsson, F., Nordlund, P.-J., 2005. Marginalized Particle Filters for Mixed Linear/Nonlinear State-Space Models. IEEE Transactions on Signal Processing 53, 2279–2289.

Silversides, K. L., Melkumyan, A., Wyman, D., 2016. Fusing gaussian processes and dynamic time warping for improved natural gamma signal classification. Mathematical Geosciences 48 (2), 187–210.

Snelson, E., Ghahramani, Z., 2005. Sparse Gaussian Processes using Pseudo-inputs. In: Neural Information Processing Systems (NIPS). pp. 1257–1264.

Solin, A., Särkkä, S., 2015. State space methods for efficient inference in student-t process regression. In: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics. pp. 885–893.

Vaerenbergh, S. V., Lázaro-Gredilla, M., Santamaría, I., 2012. Kernel Recursive Least-Squares Tracker for Time-Varying Regression. IEEE Transactions on Neural Networks and Learning Systems 23 (8), 1313–1326.

Vanhatalo, J., Jylänki, P., Vehtari, A., 2009. Gaussian Process Regression with Student-t Likelihood. In: Neural Information Processing Systems (NIPS). pp. 1910–1918.

Vanhatalo, J., Riihimäki, J., Hartikainen, J., Jylänki, P., Tolvanen, V., Vehtari, A., 2013. Gpstuff: Bayesian modeling with gaussian processes. The Journal of Machine Learning Research 14 (1), 1175–1179.

Wang, Y., Brubaker, M., Chaib-Draa, B., Urtasun, R., 2016. Sequential Inference for Deep Gaussian Process. In: AISTATS.

Wood, S. A., 2002. Bayesian Mixture of Splines for Spatially Adaptive Nonparametric Regression. Biometrika 89, 513–528.

Yao, A., Gall, J., Gool, L. V., Urtasun, R., 2011. Learning Probabilistic Non-Linear Latent Variable Models for Tracking Complex Activities. In: Neural Information Processing Systems (NIPS). pp. 1359–1367.