Interactions Between Gaussian Processes and Bayesian Estimation

Thèse

Ya Li Wang

Doctorat en informatique
Philosophiæ doctor (Ph.D.)

Québec, Canada

Résumé

L'apprentissage (machine) de modèle et l'estimation d'état sont cruciaux pour interpréter les phénomènes sous-jacents à de nombreuses applications du monde réel. Toutefois, il est souvent difficile d'apprendre le modèle d'un système et de capturer les états latents, efficacement et avec précision, en raison du fait que la connaissance du monde est généralement incertaine. Au cours des dernières années, les approches d'estimation et de modélisation bayésiennes ont été extensivement étudiées afin que l'incertain soit réduit élégamment et de manière flexible. Dans la pratique cependant, différentes limitations au niveau de la modélisation et de l'estimation bayésiennes peuvent détériorer le pouvoir d'interprétation bayésienne. Ainsi, la performance de l'estimation est souvent limitée lorsque le modèle de système manque de souplesse ou/et est partiellement inconnu. De même, la performance de la modélisation est souvent restreinte lorsque l'estimateur Bayésien est inefficace. Inspiré par ces faits, nous proposons d'étudier dans cette thèse, les connections possibles entre modélisation bayésienne (via le processus gaussien) et l'estimation bayésienne (via le filtre de Kalman et les méthodes de Monte Carlo) et comment on pourrait améliorer l'une en utilisant l'autre.

À cet effet, nous avons d'abord vu de plus près comment utiliser les processus gaussiens pour l'estimation bayésienne. Dans ce contexte, nous avons utilisé le processus gaussien comme un prior non-paramétrique des modèles et nous avons montré comment cela permettait d'améliorer l'efficacité et la précision de l'estimation bayésienne. Ensuite, nous nous somme intéressé au fait de savoir comment utiliser l'estimation bayésienne pour le processus gaussien. Dans ce cadre, nous avons utilisé différentes estimations bayésiennes comme le filtre de Kalman et les filtres particulaires en vue d'améliorer l'inférence au niveau du processus gaussien. Ceci nous a aussi permis de capturer différentes propriétés au niveau des données d'entrée. Finalement, on s'est intéressé aux interactions dynamiques entre estimation bayésienne et processus gaussien. On s'est en particulier penché sur comment l'estimation bayésienne et le processus gaussien peuvent "travailler" de manière interactive et complémentaire de façon à améliorer à la fois le modèle et l'estimation.

L'efficacité de nos approches, qui contribuent à la fois au processus gaussien et à l'estimation bayésienne, est montrée au travers d'une analyse mathématique rigoureuse et validée au moyen de différentes expérimentations reflétant des applications réelles.

Abstract

Model learning and state estimation are crucial to interpret the underlying phenomena in many real-world applications. However, it is often challenging to learn the system model and capture the latent states accurately and efficiently due to the fact that the knowledge of the world is highly uncertain. During the past years, Bayesian modeling and estimation approaches have been significantly investigated so that the uncertainty can be elegantly reduced in a flexible probabilistic manner.

In practice, however, several drawbacks in both Bayesian modeling and estimation approaches deteriorate the power of Bayesian interpretation. On one hand, the estimation performance is often limited when the system model lacks in flexibility and/or is partially unknown. On the other hand, the modeling performance is often restricted when a Bayesian estimator is not efficient and/or accurate. Inspired by these facts, we propose *Interactions Between Gaussian Processes and Bayesian Estimation* where we investigate the novel connections between Bayesian model (Gaussian processes) and Bayesian estimator (Kalman filter and Monte Carlo methods) in different directions to address a number of potential difficulties in modeling and estimation tasks.

Concretely, we first pay our attention to *Gaussian Processes for Bayesian Estimation* where a Gaussian process (GP) is used as an expressive nonparametric prior for system models to improve the accuracy and efficiency of Bayesian estimation. Then, we work on *Bayesian Estimation for Gaussian Processes* where a number of Bayesian estimation approaches, especially Kalman filter and particle filters, are used to speed up the inference efficiency of GP and also capture the distinct input-dependent data properties. Finally, we investigate *Dynamical Interaction Between Gaussian Processes and Bayesian Estimation* where GP modeling and Bayesian estimation work in a dynamically interactive manner so that GP learner and Bayesian estimator are positively complementary to improve the performance of both modeling and estimation.

Through a number of mathematical analysis and experimental demonstrations, we show the effectiveness of our approaches which contribute to both GP and Bayesian estimation.

Table des matières

Liste des tableaux

Liste des figures

Remerciements

I am genuinely grateful to my supervisor, Prof. Brahim Chaib-draa, who is always there to encourage me to open my mind with his great advices, to guide me to investigate my research works with his thoughtful instructions. I am also thankful to Prof. Raquel Urtasun, Dr. Marcus A. Brubaker and Prof. David Barber for their sincere helps during my research visit in Toyota Technological Institute at Chicago and University College London. Finally and most importantly, I dedicate this thesis to my parents, Jinyong Wang and Lanyu Zhao who always support me from deep inside their heart.

Chapitre 1

Introduction

Over the past years, system modeling and state estimation have been spotlighted due to the fact that they play important roles in system analysis for a number of research domains including robotics, signal processing, bioinformatics, computer vision, geophysics, economics, etc. (Doucet et al., 2001; Thrun et al., 2005; Bishop, 2006; Rasmussen and Williams, 2006; Plagemann, 2008; Ko, 2011). However, it is often challenging to learn the model and infer the latent states as real-world data may be noisy. Bayesian approaches have been widely investigated for modeling and estimation by interpreting the uncertainty in a probabilistic manner (Doucet et al., 2001; Bishop, 2006; Barber, 2012). But still, several drawbacks of both Bayesian modeling and estimation limit the power of Bayesian approaches in practice.

On one hand, Bayesian estimation approaches are often deteriorated due to the fact that system models often lack in flexibility and/or are partially unknown. For instance, particle filter (a well-known Bayesian estimation approach based on sequential Monte Carlo sampling) often suffers from weight degeneracy because the proposal model, which is the state transition model of the system, does not contain the current observation information (Doucet et al., 2000b; Cappé et al., 2007). On the other hand, Bayesian modeling approaches are often deteriorated due to the fact that estimation frameworks are inefficient and/or not capable to handle the challenging properties in the real-world data sets. For instance, a Gaussian process (a popular Bayesian nonparametric model) is often intractable when the number of training data is beyond a few thousands because the matrix inversion of the training data has to be performed in the gradient-based optimization or Markov Chain Monte Carlo (Rasmussen and Williams, 2006).

As we can see, the performance of Bayesian modeling and estimation is often strongly connected. Inspired by this fact, we propose to investigate our research on the *Interactions Between Gaussian Processes and Bayesian Estimation* in this thesis. From the perspective of *Gaussian Processes for Bayesian Estimation*, we mainly take advantage of Gaussian processes (GPs) to learn a powerful Bayesian nonparametric model to increase the model flexibility to improve the performance of Bayesian estimation approaches. From the perspective of *Bayesian*

Estimation for Gaussian Processes, we mainly make use of a number of Bayesian estimation approaches (including Kalman filter and Monte Carlo sampling methods) to design an accurate and efficient Bayesian inference framework to interpret the distinct data properties in the real world. Finally, we work on *Dynamical Interaction Between Gaussian Processes and Bayesian Estimation* to address the potential difficulties in both GP and Bayesian estimation approaches.

In the following, we introduce our contributions and describe the outline of this thesis.

1.1 Main Contributions

1.1.1 Gaussian Processes for Bayesian Estimation

An Adaptive Nonparametric Particle Filter for State Estimation

In Wang and Chaib-draa (2012b), we introduce an adaptive nonparametric particle filter where we incorporate a Gaussian process (GP) based proposal into a KLD-sampling particle filter framework. GP based proposal contains the observation information, and thus allows us to address weight degeneracy with a number of high-quality particles which are located at the important regions of the posterior. Then this proposal is incorporated into the KLD-sampling particle filter in which the number of the particles is adaptively learned according to the complexity of the true posterior. The resulting adaptive nonparametric particle filter improves both efficiency and accuracy of state estimation.

Gaussian Processes for Bayesian Estimation in Ordinary Differential Equations

In Wang and Barber (2014), we propose a novel generative model (based on GP) for Bayesian parameter estimation in coupled ordinary differential equations (ODEs). By using GP derivatives, we directly link state derivative information with system observations to simplify the previous GP based gradient matching approaches (Calderhead et al., 2008; Dondelinger et al., 2013) and improve the estimation accuracy.

1.1.2 Bayesian Estimation for Gaussian Processes

A Marginalized Particle Gaussian Process Regression

In Wang and Chaib-draa (2012a), we present a novel marginalized particle Gaussian process (MPGP) regression which is a fast and accurate online Bayesian filtering framework to model the latent function. Based on a state space model that is constructed according to sequential data collections, our MPGP filters out the hidden function values efficiently and accurately with a Gaussian mixture. Additionally, it provides us with an effective online method for training GP hyperparameters by a number of weighted particles.

A KNN Based Kalman Filter Gaussian Process Regression

In Wang and Chaib-draa (2013), we design a novel K nearest neighbor based Kalman

filter Gaussian process (KNN-KFGP) regression approach to model nonstationarity. Based on a state space model established by a KNN driven data grouping, our KNN-KFGP recursively filters out the latent function values in a computationally efficient and accurate Kalman filtering framework. Since KNN allows each test point to find its strongly correlated local training subset, our KNN-KFGP is a suitable way to deal with nonstationarity.

Particle Based Gaussian Processes for Time-Varying Applications

In Wang and Chaib-draa (2015a), we propose two particle based GP approaches where time-varying GP models are learned in a sequential Monte Carlo manner. By doing so, we address the difficulties in time-varying applications where the data points are sequentially ordered and often exhibit temporal-related non-stationarity and heteroscedasticity.

1.1.3 Dynamical Interaction Between Gaussian Process and Bayesian Estimation

Bayesian Filtering with Online Gaussian Process Latent Variable Models

In Wang et al. (2014), we present a novel online GP particle filter where the prediction and observation models are learned by performing dynamical interaction between GP modeling and particle filtering in an online fashion. This resulting framework is able to flexibly handle multi-modality due to the fact that we represent the prediction and observation models as GP based mixture models in which two online GP variants, sparse online GP (Csató and Opper, 2002) and local GP (Urtasun and Darrell, 2008), are explored for each component of the mixture to manage computation efficiency.

Efficient Sequential Inference for Heteroscedastic Deep Gaussian Processes Regression

In Wang and Chaib-draa (2015b), we propose a heteroscedastic deep GP (HDGP) to express input-dependent noise correlations by sharing a deep GP structure between signal and noise in the observation layer. It can be seen as a generalized version of both heteroscedastic GP and deep GP. Inspired by dynamical interaction between GP and Bayesian estimation, we then design an efficient sequential inference mechanism for our HDGP to infer the latent variables and update the model in a recursive manner. Finally, the weighting mechanism of our inference framework allows us to straightforwardly handle missing data for multi-output regression.

1.2 Thesis Outline

This thesis is organized as follows. Firstly, we introduce a background on two fundamental perspectives of this thesis, Gaussian process (GP) and Bayesian estimation in Chapter 2. Secondly, we present our contributions to *Gaussian Processes for Bayesian Estimation* in

Chapter 3. Thirdly, we present our contributions to *Bayesian Estimation for Gaussian Processes* in Chapter 4. Then, we present our contribution to *Dynamical Interaction Between Gaussian Processes and Bayesian Estimation* in Chapter 5. Finally, we conclude this thesis in Chapter 6.

Chapitre 2

Background

In this chapter, we review two fundamental parts in this thesis, Gaussian processes (GPs) and Bayesian estimation, in order to establish the interactions between them later on. For GP, we start from its definition and then illustrate how to use it to address a nonlinear regression task which is mainly investigated in the thesis. For Bayesian estimation, we begin with a brief introduction of state space models and then review a number of popular techniques that are used for Bayesian estimation including Kalman filter and Monte Carlo sampling methods.

2.1 Gaussian Processes

2.1.1 Definition

Formally, *a Gaussian process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution* (Rasmussen and Williams, 2006). According to this definition, a GP can be flexibly interpreted as a distribution over functions where the *random variables* in this case are the function values of a latent function $f(\mathbf{x})$ (where $\mathbf{x}$ is the input vector). That is why GPs have been widely used as a Bayesian nonparametric prior for $f(\mathbf{x})$ (MacKay, 1998; Rasmussen and Williams, 2006; Bishop, 2006; Barber, 2012).

Specifically, a GP is fully specified as

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

with a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$,

$$\begin{aligned} m(\mathbf{x}) &= E[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= E[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \end{aligned}$$

According to the definition of a GP, the prior $p(f(X))$ over n latent function values $f(X) = \{f(\mathbf{x}_1), \cdots, f(\mathbf{x}_n)\}$ of the input vectors $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ is Gaussian distributed, i.e.,

$\mathcal{N}(\mathbf{m}, K)$ with the mean vector,

$$\mathbf{m} = \begin{bmatrix} m(\mathbf{x}_1) \\ \cdots \\ m(\mathbf{x}_n) \end{bmatrix},$$

and the covariance matrix,

$$K = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \cdots & \cdots & \cdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}.$$

In this section, we follow Rasmussen and Williams (2006) and use a zero-mean GP,

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

for simplicity [1]. In this case, the prior $p(f(X))$ becomes $\mathcal{N}(\mathbf{0}, K)$.

Covariance Functions

Covariance functions play a key role in GPs since they encode the similarity of the function values at different input locations. Many functions of $\mathbf{x}$ and $\mathbf{x}'$ can be used as the covariance functions of GP as long as the corresponding covariance matrices K are symmetric and positive semidefinite ($\mathbf{v}^T K \mathbf{v} \geq 0$, $\forall \mathbf{v}$) (Rasmussen and Williams, 2006). In the following, we mainly introduce three popular covariance functions which are used to build up the covariance functions in this thesis. A detailed review of covariance functions can be found in Rasmussen and Williams (2006).

The **squared exponential (SE)** covariance function is one of the most popular covariance functions in the machine learning community. It is a stationary covariance function (as it is a function of $\mathbf{x} - \mathbf{x}'$) and more specifically it is an isotropic covariance function (as it is a function of $\| \mathbf{x} - \mathbf{x}' \|$) (Rasmussen and Williams, 2006). In general, it is expressed as follows :

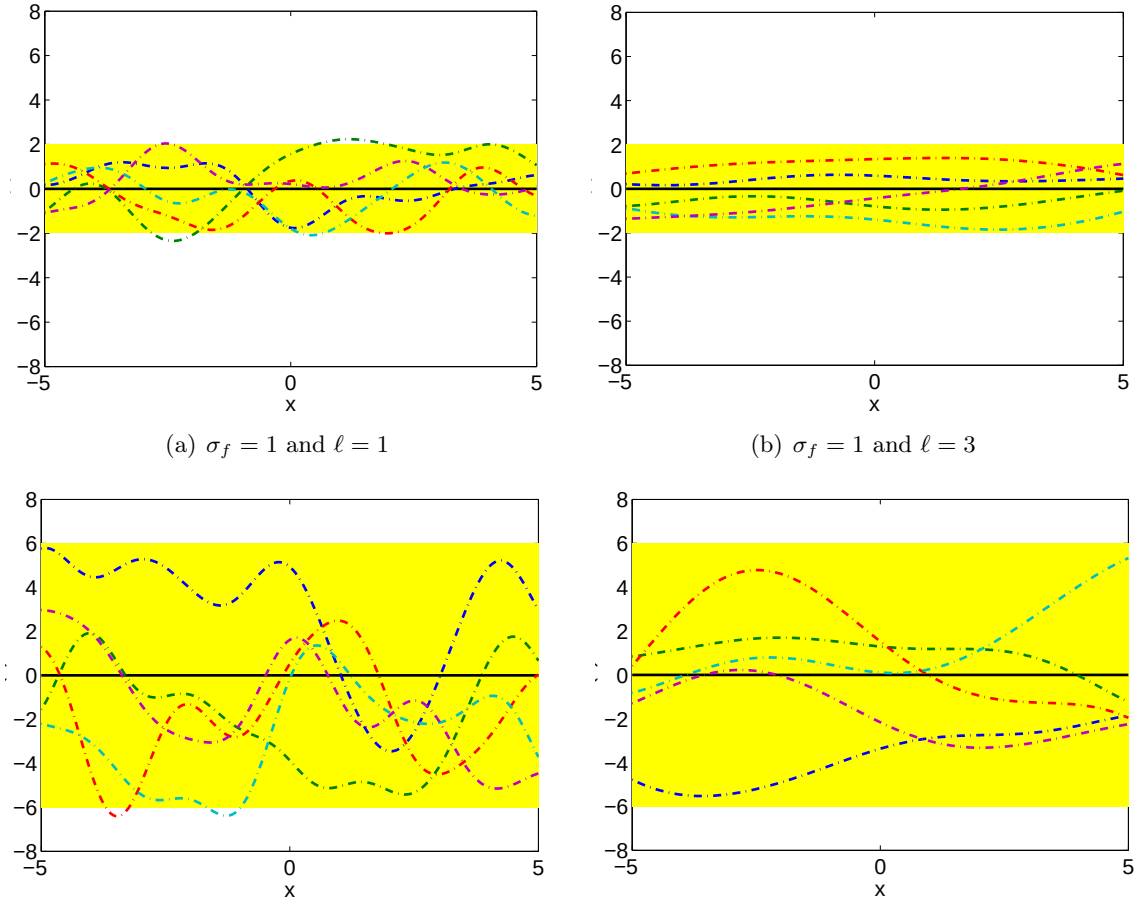
$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^T (\mathbf{x} - \mathbf{x}')}{2\ell^2}\right),$$

where the hyper-parameter σ_f controls the amplitude and ℓ controls the length-scale. An illustration of the influence of σ_f and ℓ is shown in Figure 2.1 where the sampled functions become more flat when ℓ becomes larger, and the amplitude of the sampled functions become larger when σ_f becomes larger.

In this thesis we also use two nonstationary covariance functions, the **dot product** covariance function,

$$k(\mathbf{x}, \mathbf{x}') = \ell^2 \mathbf{x}^T \mathbf{x}',$$

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



(a) $\sigma_f = 1$ and $\ell = 1$

(b) $\sigma_f = 1$ and $\ell = 3$

(c) $\sigma_f = 3$ and $\ell = 1$

(d) $\sigma_f = 3$ and $\ell = 3$

FIGURE 2.1: The influence of σ_f and ℓ for a GP prior with the SE covariance function. In all the plots, the dashed lines are 5 functions drawn from the GP prior. The black line is the mean of the GP prior which is zero. The yellow interval is the 95% confidence interval. The sampled functions become more flat when ℓ becomes larger, and the amplitude of the sampled functions become larger when σ_f becomes larger.

and the **neural network** covariance function with an augmented input $\tilde{\mathbf{x}} = [1; \mathbf{x}]$,

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \sin^{-1}\left(\frac{\tilde{\mathbf{x}}^T \tilde{\mathbf{x}}'}{\ell^2 \sqrt{(1 + \frac{1}{\ell^2}\tilde{\mathbf{x}}^T\tilde{\mathbf{x}})(1 + \frac{1}{\ell^2}\tilde{\mathbf{x}}'^T\tilde{\mathbf{x}}')}}\right).$$

In Figure 2.2, it is clearly shown that the nonstationary covariance functions are not only related to $\mathbf{x} - \mathbf{x}'$, but they are also related to the input locations $\mathbf{x}$ and $\mathbf{x}'$ themselves. That is why these covariance functions are often used to capture the nonstationary correlations.

2.1.2 Gaussian Process Regression

Suppose that the data generation procedure is based on the following observation model,

$$y = f(\mathbf{x}) + \epsilon, \tag{2.1}$$

where the input vector is $\mathbf{x} \in R^d$, the scalar output is $y \in R$ and the observation noise is $\epsilon \sim N(0, \sigma_y^2)$. In general, there are two tasks in a nonlinear regression problem

1. Learning the model parameters given a training set $(X, \mathbf{y}) = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$.
2. Predicting $f(X_\star)$ for a given test input set $X_\star = \{\mathbf{x}_\star^i\}_{i=1}^M$.

Since Bayesian parametric approaches restrict the richness of the assumed function family when solving the regression problem, Bayesian nonparametric methods have been significantly investigated by giving a prior probability to every possible function (Bishop, 2006; Rasmussen and Williams, 2006).

Gaussian process (GP) is a popular nonparametric prior as it allows us to make an analytically tractable Bayesian inference to obtain a predictive distribution over $f(X_\star)$. Concretely, let the GP prior be $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$ where for simplicity of notation we put all the hyperparameters of $k(\mathbf{x}, \mathbf{x}')$ and the variance of the noise σ_y^2 into a vector θ. In the following, we first use the training set $(X, \mathbf{y})$ to infer θ, which is the model parameter vector.

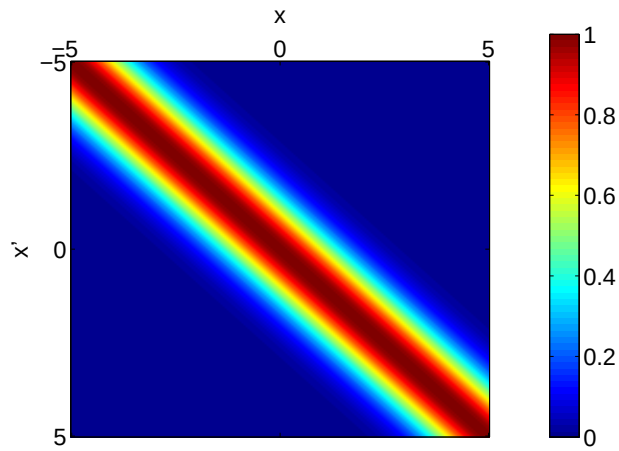
Hyper-parameter Learning

Several methods have been applied for hyperparameter learning, such as gradient based optimization, Markov Chain Monte Carlo (MCMC), etc. A detailed review can be found in Rasmussen and Williams (2006). Here we briefly introduce a popular gradient based optimization approach.

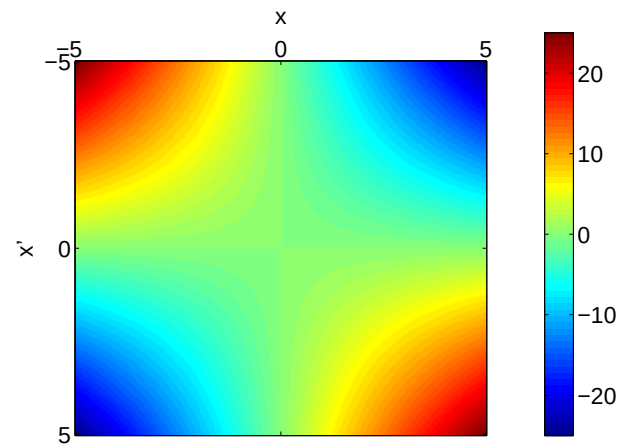
Due to the fact [2] that $p(f(X)|X, \theta) \sim \mathcal{N}(\mathbf{0}, K(X, X))$ and $p(y|f(X), \theta) \sim \mathcal{N}(f(X), \sigma_y^2 I)$, the log marginal likelihood is analytically tractable,

$$\begin{aligned} \log p(\mathbf{y}|X, \theta) &= \log \int p(y|f(X), \theta) p(f(X)|X, \theta) df(X) \\ &= -\frac{1}{2}\mathbf{y}^T (K(X, X) + \sigma_y^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log|K(X, X) + \sigma_y^2 I| - \frac{n}{2}\log 2\pi. \end{aligned} \tag{2.2}$$

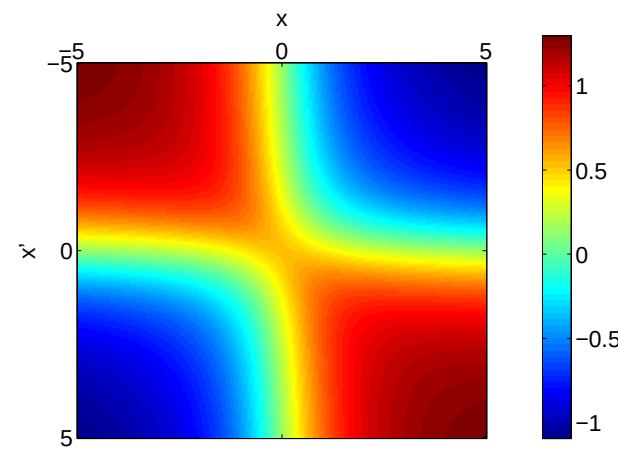
2. $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$ and the observation model (Equation 2.1) is Gaussian



(a) SE Covariance Function ($\sigma_f = 1$, $\ell = 1$)



(b) Dot Product Covariance Function ($\ell = 1$)



(c) Neural Network Covariance Function ($\sigma_f = 1$, $\ell = 1$)

FIGURE 2.2: The comparison of the covariance matrix using different covariance functions for 201 equally-spaced-ordered input points (from -5 to 5).

Then the partial derivatives of Equation 2.2 (with regard to the hyperparameters),

$$\frac{\partial}{\partial \theta_j} \log p(\mathbf{y}|X,\theta) = \frac{1}{2} tr[((K(X,X)^{-1}\mathbf{y})(K(X,X)^{-1}\mathbf{y})^T - K(X,X)^{-1})\frac{\partial K(X,X)}{\partial \theta_j}], \quad (2.3)$$

are used in the gradient-based optimization to learn the hyperparameters which maximize the marginal likelihood in Equation 2.2 (Rasmussen and Williams, 2006). Additionally, it is worth mentioning that there is no guarantee of achieving a global optimum since this optimization problem is non-convex (Rasmussen and Williams, 2006). In practice, however, this gradient based optimization approach tends to work well.

Prediction

After θ is learned, we now make the prediction of $f(X_\star)$ for the test input set $X_\star$. Rather than only the point prediction of $f(X_\star)$, a GP allows us to obtain a full Bayesian predictive distribution over $f(X_\star)$ due to the fact that a GP is a nonparametric prior.

Concretely, we can first take advantage of $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$ and the Gaussian observation model in Equation 2.1 to obtain that the joint distribution $p(\mathbf{y}, f(X_\star)|X, X_\star, \theta)$ is Gaussian,

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

Then, based on $p(\mathbf{y}, f(X_\star)|X, X_\star, \theta)$ and the conditional property of Gaussian distribution, we can obtain that the predictive distribution over $f(X_\star)$, i.e., $p(f(X_\star)|X, \mathbf{y}, X_\star, \theta)$, is also Gaussian with the mean $\bar{f}(X_\star)$ and the covariance $P(X_\star, X_\star)$ (Rasmussen and Williams, 2006) :

$$\begin{aligned} \bar{f}(X_\star) &= K(X_\star, X)[K(X,X) + \sigma_y^2 I]^{-1}\mathbf{y}, \\ P(X_\star, X_\star) &= K(X_\star, X_\star) - K(X_\star, X)[K(X,X) + \sigma_y^2 I]^{-1} K(X_\star, X)^T, \end{aligned}$$

where $K(X_\star, X)$ denotes an $M \times N$ covariance matrix in which each entry is computed by the covariance function $k(\mathbf{x}, \mathbf{x}')$ with the learned θ. The $K(X,X)$ and $K(X_\star, X_\star)$ are constructed in a similar way.

An illustration of the predictive posterior is shown in Figure 2.3. The 20 training input-output pairs are obtained from $y = f(\mathbf{x}) + \epsilon$ where the latent function is $f(\mathbf{x}) = \sin(x)$, the observation noise ϵ is $\mathcal{N}(0,1)$, and the training inputs are randomly selected from [-5, 5]. Then we choose that the GP prior is $\mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$ where $k(\mathbf{x}, \mathbf{x}')$ is the SE covariance function with the initial hyperparameters $\sigma_f = 2$, $\ell = 1$ (introduced in Subsection 2.1.1), and the initial standard derivation of the noise $\sigma_y = 2$. Finally the test inputs are 201 equally-spaced-ordered input points from -5 to 5.

From Figure 2.3, the performance of the posterior (without hyperparameter learning) is better than the performance of the prior. The reason is that, compared to the prior, the information

of the training data is incorporated into the posterior. Moreover, the posterior (with hyperparameter learning) performs better than the one without hyperparameter learning. This is because hyperparameter learning is based on the maximization of the marginal likelihood, which helps the GP fit the data more correctly.

Computation Complexity

The computation complexity of GP is mainly governed by the training phase where the inversion of the covariance matrix of the N training points in Equation 2.3 has to be operated in the optimization, and this results in the complexity $O(N^3)$ (Rasmussen and Williams, 2006; Ko, 2011). Hence, a standard GP is often intractable for large data sets.

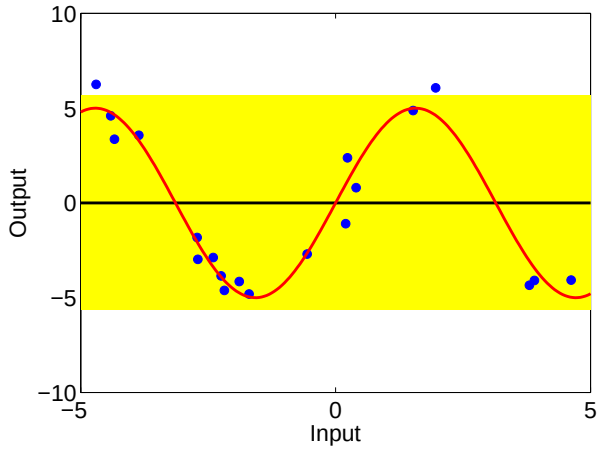
In the following, we will briefly introduce two computationally-efficient GP variants, Sparse Online Gaussian Process (SOGP) (Csató and Opper, 2002; Vaerenbergh et al., 2012) and Local Gaussian Processes (LGP) (Nguyen-Tuong et al., 2008; Urtasun and Darrell, 2008), which are used in this thesis. A detailed review of sparse approximate GPs can be found in Quinonero-Candela and Rasmussen (2005); Chalupka et al. (2013).

Sparse Online Gaussian Process (SOGP)

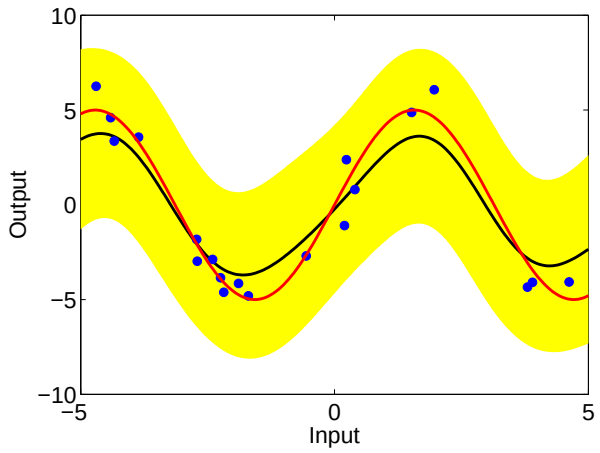
Sparse Online Gaussian Process (SOGP) of Csató and Opper (2002); Vaerenbergh et al. (2012) is a well-known algorithm for online learning of GP models. To cope with the fact that data arrives in an online manner, SOGP trains a GP model sequentially by updating the posterior mean and covariance of the latent function values of the training set. This online procedure is coupled with a sparsification strategy which iteratively selects a fixed-size subset of training points to form the active set, preventing the otherwise unbounded growth of the computation and memory load.

The key of SOGP is to maintain the joint posterior over the latent function values of the fixed-size active set $\mathcal{D}_{t-1}$, i.e., $\mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$, by recursively updating μ_{t-1} and Σ_{t-1}. We summarize the whole procedure of SOGP update in Algorithm 1.

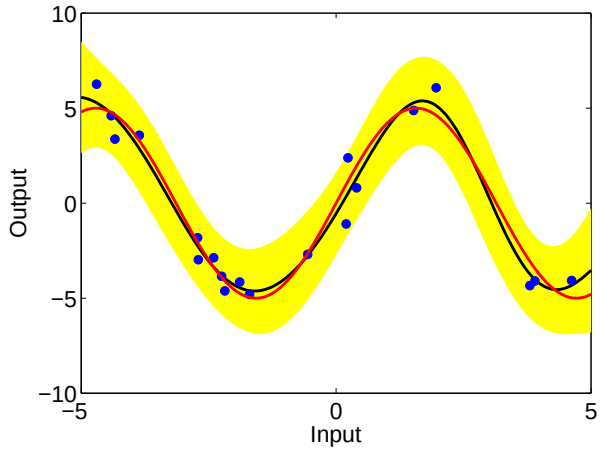
When a new data point $(\mathbf{x}_t, y_t)$ is available, we first perform the following update (Vaerenbergh



(a) Prior



(b) Posterior Without Hyperparameter Learning



(c) Posterior With Hyperparameter Learning

FIGURE 2.3: Predictive Posterior. In all the plots, the training data points are the blue dots. The predictive mean with 95% confidence interval is the black line with the yellow region. The ground truth of the latent function is the red curve.

et al., 2012) to take this new data point into account (Algorithm 1, Line 1) :

$$\mathbf{q}_t = Q_{t-1}\mathbf{k}_{t-1}(\mathbf{x}_t), \tag{2.4}$$

$$\rho_t^2 = k(\mathbf{x}_t, \mathbf{x}_t) - \mathbf{k}_{t-1}(\mathbf{x}_t)^T Q_{t-1}\mathbf{k}_{t-1}(\mathbf{x}_t), \tag{2.5}$$

$$\hat{\sigma}_t^2 = \sigma_y^2 + \rho_t^2 + \mathbf{q}_t^T \Sigma_{t-1}\mathbf{q}_t, \tag{2.6}$$

$$\delta_t = \begin{bmatrix} \Sigma_{t-1}\mathbf{q}_t \\ \rho_t^2 + \mathbf{q}_t^T \Sigma_{t-1}\mathbf{q}_t \end{bmatrix}, \tag{2.7}$$

$$\mu_t = \begin{bmatrix} \mu_{t-1} \\ \mathbf{q}_t^T \mu_{t-1} \end{bmatrix} + \hat{\sigma}_t^{-2}(y_t - \mathbf{q}_t^T \mu_{t-1})\delta_t, \tag{2.8}$$

$$\Sigma_t = \begin{bmatrix} \Sigma_{t-1} & \Sigma_{t-1}\mathbf{q}_t \\ \mathbf{q}_t^T \Sigma_{t-1} & \rho_t^2 + \mathbf{q}_t^T \Sigma_{t-1}\mathbf{q}_t \end{bmatrix} - \hat{\sigma}_t^{-2}\delta_t\delta_t^T, \tag{2.9}$$

where $\mathbf{k}_{t-1}(\mathbf{x}_t)$ is the kernel vector which is constructed from $\mathbf{x}_t$ and the active set $\mathcal{D}_{t-1}$, and Q_{t-1} is the inverse kernel matrix of the active set $\mathcal{D}_{t-1}$.

Next, we decide whether this new point $(\mathbf{x}_t, y_t)$ is added to the active set. Following the strategy suggested by (Csató and Opper, 2002; Vaerenbergh et al., 2012), we ignore this new point when ρ_t^2 in Equation 2.5 is smaller than ϵ (It is a small value that is close to zero. In our experiment, $\epsilon = 10^{-6}$). In this case (Algorithm 1, Line 2-4), μ_t, Σ_t in Equation (2.8) and (2.9) are reduced as $\mu_t \leftarrow [\mu_t]_{-i}$, $\Sigma_t \leftarrow [\Sigma_t]_{-i,-i}$ where $i = t$ is the index of the new point, $[\cdot]_{-i}$ removes the i-th entry of a vector, and $[\cdot]_{-i,-i}$ removes the i-th row and column of a matrix. Additionally, the inverse kernel matrix is simply $Q_t = Q_{t-1}$ because the new point is not included in the active set.

When $\rho_t^2 \geq \epsilon$, we add the new point to the active set $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, y_t)\}$ (Algorithm 1, Line 6-7). The μ_t, Σ_t are then the same as Equation (2.8) and (2.9), and the inverse kernel matrix is updated to be (Vaerenbergh et al., 2012)

$$Q_t = \begin{bmatrix} Q_{t-1} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} + \rho_t^{-2} \begin{bmatrix} \mathbf{q}_t\mathbf{q}_t^T & -\mathbf{q}_t \\ -\mathbf{q}_t^T & 1 \end{bmatrix}. \tag{2.10}$$

Because this new point $(\mathbf{x}_t, y_t)$ was added to the active set, we should compare the size of the active set to the fixed size N_A to maintain computation load. When the size of the active set is larger than N_A, we have to remove a data point (Algorithm 1, Line 8-12). This is done by selecting the one which minimally affects the predictions according to the squared predicted error. Following the strategy in Csató and Opper (2002); Vaerenbergh et al. (2012), we remove the j-th data point with

$$j = \arg\min_j \left(\frac{[Q_t\mu_t]_j}{[Q_t]_{j,j}} \right)^2, \tag{2.11}$$

where $[\cdot]_j$ selects the j-th entry of a vector and $[\cdot]_{j,j}$ select the jth diagonal entry of a matrix. Once a point has been selected for removal, μ_t, Σ_t and Q_t in Equation (2.8), (2.9) and (2.10)

Algorithm 1 SOGP Update.

input Previous posterior quantities μ_{t-1}, Σ_{t-1}, Q_{t-1}
input Previous active set $\mathcal{D}_{t-1}$
input New input-output observation $(\mathbf{x}_t, y_t)$ pair

```
Compute ρ_t, μ_t and Σ_t as in Equations (2.5), (2.8) and (2.9).
if ρ_t^2 < ε then
    Perform update μ_t ← [μ_t]_{-i}, Σ_t ← [Σ_t]_{-i,-i} where i is the index of the newly added row
    to μ_t.
    Set Q_t = Q_{t-1}, D_t = D_{t-1}.
else
    Compute Q_t as in Equation (2.10).
    Add to active set D_t = D_{t-1} ∪ {(x_t, y_t)}.
    if |D_t| > N_A then
        Select point j to remove using Equation (2.11).
        Perform update μ_t ← [μ_t]_{-j}, Σ_t ← [Σ_t]_{-j,-j} and Q_t ← [Q_t]_{-j,-j} − ([Q_t]_{-j,j}[Q_t]^T_{-j,j}) / [Q_t]_{j,j}.
        Remove j from active set D_t ← D_t \ {(x_j, y_j)}.
    end if
end if
```

output μ_t, Σ_t, Q_t and $\mathcal{D}_t$

are reduced to

$$\mu_t \leftarrow [\mu_t]_{-j} \tag{2.12}$$

$$\Sigma_t \leftarrow [\Sigma_t]_{-j,-j} \tag{2.13}$$

$$Q_t \leftarrow [Q_t]_{-j,-j} - \frac{[Q_t]_{-j,j}[Q_t]^T_{-j,j}}{[Q_t]_{j,j}}, \tag{2.14}$$

where $[\cdot]_{-j,j}$ selects the j-th column of the matrix with the j-th row removed and the point is removed from the active set $\mathcal{D}_t \leftarrow \mathcal{D}_t \setminus \{(\mathbf{x}_j, y_j)\}$.

Finally, the joint posterior at time t can be used to construct the predictive distribution for a new input $\mathbf{x}^*$,

$$p^{SOGP}(y|\mathbf{x}^*, \mathcal{D}_t, \Theta) = \mathcal{N}(y|\tilde{y}, \tilde{\sigma}^2), \tag{2.15}$$

where $\tilde{y} = \mathbf{k}_t(\mathbf{x}^*)^T Q_t \mu_t$ and $\tilde{\sigma}^2 = \sigma_y^2 + k(\mathbf{x}^*, \mathbf{x}^*) + \mathbf{k}_t(\mathbf{x}^*)^T (Q_t \Sigma_t Q_t - Q_t)\mathbf{k}_t(\mathbf{x}^*)$.

Local Gaussian Processes (LGP)

Another GP variants we used is Local Gaussian Processes (LGP), which was developed specifically to deal with large, multi-modal regression problems (Nguyen-Tuong et al., 2008; Urtasun and Darrell, 2008). In LGP, given a test input $\mathbf{x}^*$ and a set of input-output pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N\}$, the $M_\mathbf{x}$-nearest neighbors $\mathcal{D}_{\mathbf{x}^*} = \{(\mathbf{x}_\ell, y_\ell)\}_{\ell=1}^{M_\mathbf{x}}$ are selected based on the distance in the input space $d_\ell = \|\mathbf{x}_\ell - \mathbf{x}^*\|$. Then, for each of the $M_\mathbf{x}$ neighbors, M_y-nearest neighbors $\mathcal{D}_{y_\ell} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{M_y}$ are selected based on the distance in the output space to y_ℓ.

These neighbors are then combined to form a local GP expert which makes a Gaussian prediction with the following mean and covariance,

$$\begin{aligned}\mu_\ell &= B_{\mathcal{D}_{y_\ell}} K^{-1}_{\mathcal{D}_{y_\ell},\mathcal{D}_{y_\ell}} \mathbf{k}_{\mathcal{D}_{y_\ell}}(\mathbf{x}^*),\\ \sigma^2_\ell &= k(\mathbf{x}^*,\mathbf{x}^*) - \mathbf{k}_{\mathcal{D}_{y_\ell}}(\mathbf{x}^*)^T K^{-1}_{\mathcal{D}_{y_\ell},\mathcal{D}_{y_\ell}} \mathbf{k}_{\mathcal{D}_{y_\ell}}(\mathbf{x}^*) + \sigma^2_y,\end{aligned}$$

where $B_{\mathcal{D}_{y_\ell}}$ is the matrix whose columns are the M_y nearest neighbors of y_ℓ, $\mathbf{k}_{\mathcal{D}_{y_\ell}}(\mathbf{x}^*)$ is the vector of kernel function values for the input $\mathbf{x}^*$ and the points in $\mathcal{D}_{y_\ell}$, and $K_{\mathcal{D}_{y_\ell},\mathcal{D}_{y_\ell}}$ is the kernel matrix for the points in $\mathcal{D}_{y_\ell}$. The final predictive distribution is then formed by combining all local experts in a mixture model :

$$p^{LGP}(y|\mathbf{x}^*,\mathcal{D},\Theta) = \sum_{\ell=1}^{M_\mathbf{x}} w_\ell \mathcal{N}(y|\mu_\ell,\sigma^2_\ell), \tag{2.16}$$

with weights $w_\ell \propto 1/d_\ell$.

2.2 Bayesian Estimation

After introducing GPs, we focus on the other fundamental part of this thesis, i.e., Bayesian estimation. Since Bayesian estimation in this thesis is mainly based on state space model (SSM), we start from a brief introduction of SSM. Then we review a number of popular Bayesian estimation approaches that will be used in the following sections.

2.2.1 State Space Model

A general state space model (SSM) is a continuous-time dynamical model (Chen, 2003),

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{f}(t,\mathbf{x}_t,\mathbf{u}_t,\mathbf{w}_t), \tag{2.17}$$

$$\mathbf{y}_t = \mathbf{h}(t,\mathbf{x}_t,\mathbf{u}_t,\mathbf{v}_t), \tag{2.18}$$

where Equation 2.17 and 2.18 are respectively called transition and observation models [3]. $\mathbf{x}_t$ is the state vector, $\mathbf{y}_t$ is the observation vector, $\mathbf{u}_t$ is the system input vector (driving force) in a controlled environment, $\mathbf{f}$ and $\mathbf{h}$ are nonlinear functions, $\mathbf{w}_t$ and $\mathbf{v}_t$ are system and observation noise. In practice however the discrete-time SSM, which is a first-order hidden Markov model, is more widely used (Doucet et al., 2000b, 2001; Cappé et al., 2007; Barber, 2012) [4]

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1},\mathbf{w}_{t-1}), \tag{2.19}$$

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t,\mathbf{v}_t). \tag{2.20}$$

To take a Bayesian treatment, the transition and observation models (Equations 2.19 and 2.20) are often rewritten in a probabilistic manner, i.e., $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and $p(\mathbf{y}_t|\mathbf{x}_t)$, by assuming

3. The transition model (Equation 2.17) is also called as the prediction model or motion model.

4. For simplicity of mathematical deviation, driving force $\mathbf{u}_t$ is omitted here. However, the extension to a control system is straightforward.

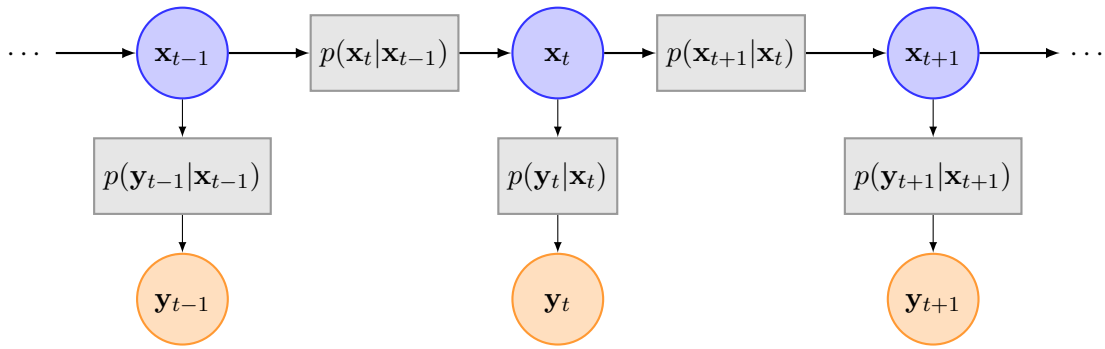


FIGURE 2.4: State Space Model.

that the probability distributions of $\mathbf{w}_t$ and $\mathbf{v}_t$ are known. The graph model is shown in Figure 2.4.

Bayesian estimation in SSM refers to a Bayesian filtering task where the problem is to infer the posterior of the latent states $p(\mathbf{x}_t|\mathbf{y}_{0:t})$ or $p(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})$ given a history of the observations $\mathbf{y}_{0:t} = \{\mathbf{y}_0, \mathbf{y}_1, \cdots, \mathbf{y}_t\}$.

2.2.2 Kalman Filter

We begin with a simple case in which a SSM is linear with Gaussian noises,

$$\mathbf{x}_t = F\mathbf{x}_{t-1} + \mathbf{w}_{t-1}, \tag{2.21}$$

$$\mathbf{y}_t = H\mathbf{x}_t + \mathbf{v}_t, \tag{2.22}$$

where $\mathbf{f}(\mathbf{x}_{t-1}, \mathbf{w}_{t-1})$ in Equation 2.19 and $\mathbf{h}(\mathbf{x}_t, \mathbf{v}_t)$ in Equation 2.20 become linear functions, $F\mathbf{x}_{t-1}$ and $H\mathbf{x}_t$ with a transition matrix F and an observation matrix H. Additionally, both noises $\mathbf{w}_t$ and $\mathbf{v}_t$ are assumed to be Gaussian, i.e., $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, Q)$ and $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, R)$ with covariance matrices Q and R.

To efficiently infer the posterior of the hidden states, $p(\mathbf{x}_t|\mathbf{y}_{0:t})$ is factorized as follows :

$$p(\mathbf{x}_t|\mathbf{y}_{0:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{0:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{0:t-1})}, \tag{2.23}$$

where

$$p(\mathbf{x}_t|\mathbf{y}_{0:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1})d\mathbf{x}_{t-1}, \tag{2.24}$$

$$p(\mathbf{y}_t|\mathbf{y}_{0:t-1}) = \int p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{0:t-1})d\mathbf{x}_t. \tag{2.25}$$

Equation 2.24 is a one-step prediction and Equation 2.25 is a normalized constant.

According to Equation 2.21 and 2.22, it is straightforward to obtain that

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{x}_{t-1}) &= \mathcal{N}(F\mathbf{x}_{t-1}, Q), \\ p(\mathbf{y}_t|\mathbf{x}_t) &= \mathcal{N}(H\mathbf{x}_t, R). \end{aligned}$$

Then all the distributions in Equation 2.23 are Gaussian, and this leads to a Kalman filter (Kalman, 1960). Suppose that the posterior at $t-1$ is $p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1}) = \mathcal{N}(\mathbf{x}_{t-1|t-1}, P_{t-1|t-1})$ with the mean $\mathbf{x}_{t-1|t-1}$ and the covariance $P_{t-1|t-1}$, then the one-step prediction in Equation 2.24 is Gaussian, $p(\mathbf{x}_t|\mathbf{y}_{0:t-1}) = \mathcal{N}(\mathbf{x}_{t|t-1}, P_{t|t-1})$

$$\begin{aligned} \mathbf{x}_{t|t-1} &= F\mathbf{x}_{t-1|t-1}, \\ P_{t|t-1} &= FP_{t-1|t-1}F^T + Q. \end{aligned}$$

Once the observation at t is available, one-step prediction can be updated to the posterior at t that is also Gaussian $p(\mathbf{x}_t|\mathbf{y}_{0:t}) = \mathcal{N}(\mathbf{x}_{t|t}, P_{t|t})$ according to Equation 2.23,

$$\begin{aligned} \Gamma_t &= P_{t|t-1}H^T(HP_{t|t-1}H^T + R)^{-1}, \\ \mathbf{x}_{t|t} &= \mathbf{x}_{t|t-1} + \Gamma_t(\mathbf{y}_t - H\mathbf{x}_{t|t-1}), \\ P_{t|t} &= P_{t|t-1} - \Gamma_t HP_{t|t-1}. \end{aligned}$$

Even though the Kalman filter is an optimal maximum-a-posterior (MAP) filter for linear systems with Gaussian noises, its performance is often poor in many real-world applications which are nonlinear and/or non-Gaussian systems (Jazwinski, 1970; Julier et al., 1995; Cappé et al., 2007). Many variants of the Kalman filter were proposed to improve the estimation performance, such as extended Kalman filter (Jazwinski, 1970) and unscented Kalman filter (Julier et al., 1995). But their estimation accuracy is still distorted when the systems are highly nonlinear and non-Gaussian.

To deal with Bayesian estimation for a general SSM (Equation 2.19 and 2.20), we introduce Monte Carlo sampling methods in the following to efficiently approximate a target distribution with a collection of random samples (Doucet et al., 2001; Andrieu et al., 2003; Cappé et al., 2007).

2.2.3 Monte Carlo Methods

The core of Monte Carlo sampling methods is to draw a set of samples $\mathbf{x}_{0:t}^{(i)}$ $(i = 1, \cdots, N_p)$ from the target posterior $p(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})$. Using these samples, Monte Carlo approximations are as follows :

$$\begin{aligned} \hat{p}(\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) &= \frac{1}{N_p}\sum_{i=1}^{N_p}\delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t}), \\ E[g(\mathbf{x}_{0:t})] &= \int g(\mathbf{x}_{0:t})p(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})d\mathbf{x}_{0:t} \approx \frac{1}{N_p}\sum_{i=1}^{N_p} g(\mathbf{x}_{0:t}^{(i)}), \end{aligned}$$

Algorithm 2 SIR Particle Filter.

```
for t = 0 to T do
    for i = 1 to N_p do
        x_t^(i) ~ p(x_t | x_{t-1}^(i))
        Compute the weight by Equation 2.30
    end for
    Normalize weights by Equation 2.29
    Resample particles according to the normalized weights
end for
```

where $\delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t})$ is the delta-dirac mass at $\mathbf{x}_{0:t}^{(i)}$ and $g(\mathbf{x}_{0:t})$ is a nonlinear function of $\mathbf{x}_{0:t}$.

However, it is often infeasible to directly draw samples from $p(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})$ due to the fact that for a general SSM (Equation 2.19 and 2.20) the target posterior $p(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})$ is analytically intractable (based on Equation 2.23). In the following, we will briefly review two popular Monte Carlo sampling methods for Bayesian estimation, i.e., sequential Monte Carlo (SMC) and Markov Chain Monte Carlo (MCMC). Detailed deviations and reviews can be found in Doucet et al. (2001); Chen (2003); Andrieu et al. (2003); Cappé et al. (2007).

Sequential Monte Carlo : Particle Filter

Sequential Monte Carlo (SMC) is mainly based on a sequential importance sampling mechanism in which importance sampling is performed recursively for approximation. According to importance sampling, we can write the following expectation as

$$\begin{aligned} E[g(\mathbf{x}_{0:t})] &= \int g(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) d\mathbf{x}_{0:t} \\ &= \int g(\mathbf{x}_{0:t}) \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})} q(\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) d\mathbf{x}_{0:t}, \end{aligned}$$

where the proposal distribution is $q(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})$ and the weight is defined as

$$\mathbf{w}_t(\mathbf{x}_{0:t}) = \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})}. \tag{2.26}$$

Then, the expectation above can be transformed into

$$E[g(\mathbf{x}_{0:t})] = \frac{\int g(\mathbf{x}_{0:t}) \mathbf{w}_t(\mathbf{x}_{0:t}) q(\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) d\mathbf{x}_{0:t}}{\int \mathbf{w}_t(\mathbf{x}_{0:t}) q(\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) d\mathbf{x}_{0:t}}. \tag{2.27}$$

Suppose the proposal distribution can be factorized as :

$$q(\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) = q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{0:t}) q(\mathbf{x}_{0:t-1}|\mathbf{y}_{0:t-1}),$$

Algorithm 3 Metropolis-Hastings Sampling.

```
Initialize x^(0)
for i = 0 to N_p do
    x* ~ q(x*|x^(i))
    u ~ U(0,1)
    if u < α(x*, x^(i)) = min{1, p(x*)q(x^(i)|x*) / p(x^(i))q(x*|x^(i))} then
        x^(i+1) = x*
    else
        x^(i+1) = x^(i)
    end if
end for
```

then the weight (Equation 2.26) can be computed recursively

$$\begin{aligned}\mathbf{w}_t(\mathbf{x}_{0:t}) &= \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})}{q(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})} \\ &= \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{0:t-1}|\mathbf{y}_{0:t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{0:t})q(\mathbf{x}_{0:t-1}|\mathbf{y}_{0:t-1})} \\ &= \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{0:t})}\mathbf{w}_{t-1}(\mathbf{x}_{0:t-1}). \end{aligned} \tag{2.28}$$

By drawing N_p samples $\mathbf{x}_t^{(i)}$ at time t from $q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{0:t})$, Monte Carlo approximations of the expectation (Equation 2.27) can be obtained as follows :

$$\hat{E}[g(\mathbf{x}_{0:t})] = \frac{\frac{1}{N_p}\sum_{i=1}^{N_p}\mathbf{w}_t(\mathbf{x}_{0:t}^{(i)})g(\mathbf{x}_{0:t}^{(i)})}{\frac{1}{N_p}\sum_{i=1}^{N_p}\mathbf{w}_t(\mathbf{x}_{0:t}^{(i)})} = \sum_{i=1}^{N_p}\tilde{\mathbf{w}}_t(\mathbf{x}_{0:t}^{(i)})g(\mathbf{x}_{0:t}^{(i)}),$$

where $\tilde{\mathbf{w}}_t(\mathbf{x}_{0:t}^{(i)})$ is the normalized weight,

$$\tilde{\mathbf{w}}_t(\mathbf{x}_{0:t}^{(i)}) = \frac{\mathbf{w}_t(\mathbf{x}_{0:t}^{(i)})}{\sum_{i=1}^{N_p}\mathbf{w}_t(\mathbf{x}_{0:t}^{(i)})}. \tag{2.29}$$

It is also straightforward to obtain a Monte Carlo approximation of the target posterior

$$\hat{p}(\mathbf{x}_{0:t}|\mathbf{y}_{0:t}) = \sum_{i=1}^{N_p}\tilde{\mathbf{w}}_t(\mathbf{x}_{0:t}^{(i)})\delta_{\mathbf{x}_{0:t}^{(i)}}(\mathbf{x}_{0:t}).$$

Generally, a suitable proposal $q(\mathbf{x}_{0:t}|\mathbf{y}_{0:t})$ is important for the performance of the approximation. The optimal proposal, which minimizes the variance on the importance weights, is given by (Doucet et al., 2000b; Andrieu et al., 2003)

$$q(\mathbf{x}_t|\mathbf{x}_{0:t-1},\mathbf{y}_{0:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{y}_t).$$

However, it is infeasible to sample from this unknown proposal. To simplify the sampling mechanism, the proposal is often chosen as the transition model,

$$q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{0:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}).$$

Then the weight (Equation 2.28) can be updated recursively according to the observation model $p(\mathbf{y}_t|\mathbf{x}_t)$,

$$\begin{aligned}\mathbf{w}_t(\mathbf{x}_{0:t}) &= \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{y}_{0:t})}\mathbf{w}_{t-1}(\mathbf{x}_{0:t-1}) \\ &= p(\mathbf{y}_t|\mathbf{x}_t)\mathbf{w}_{t-1}(\mathbf{x}_{0:t-1}). \end{aligned} \tag{2.30}$$

Additionally, sequential importance sampling often suffers from weight degeneracy where only a few particles have non-zero weight after a number of sampling iteration. To alleviate this difficulty, a resampling step was proposed by sampling the drawn particles with the importance weights (after importance sampling at each iteration). This is the well-known sampling importance resampling (SIR) particle filter (Doucet et al., 2001; Chen, 2003; Andrieu et al., 2003; Cappé et al., 2007). The whole algorithm is summarized in Algorithm 2 where at each filtering iteration a number of particles are firstly drawn from the transition model and then weighted by the observation model (Algorithm 2, Line 2-6). Finally they are resampled according to the normalized weights for the next step (Algorithm 2, Line 7).

It is worth mentioning that a particle filter allows us to efficiently perform online approximation of the target distribution with $O(N_p)$ where N_p is the number of the particles (Doucet et al., 2001). This motivating factor of the particle filter plays an important role in processing data streams in practice.

Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) is another Monte Carlo sampling method where the samples drawn from the proposal construct a Markov chain to approximate the target posterior in a batch way. In the following we mainly introduce two popular MCMC methods, i.e., the Metropolis-Hastings sampling method (Metropolis et al., 1953; Hastings, 1970; Andrieu et al., 2003) and the Gibbs sampling method (Geman and Geman, 1984; Andrieu et al., 2003).

The Metropolis-Hastings (MH) sampling [5] is one of the most practical MCMC methods. The whole MH sampling mechanism is summarized in Algorithm 3 where a candidate sample x^* is first drawn from a proposal $q(x^*|x)$ given the current x (Algorithm 3, Line 3). Then the Markov chain moves toward x^* with the following acceptance probability (Algorithm 3, Line 4-6),

$$\alpha(x^*, x) = \min\{1, \frac{p(x^*)q(x|x^*)}{p(x)q(x^*|x)}\}.$$

5. For the simplicity of the notations, we denote a general target distribution as $p(x)$.

Algorithm 4 Gibbs Sampling.

1: Initialization : $x_{1:m}^{(0)}$
2: **for** $i = 0$ **to** N_p **do**
3: $\quad x_1^{(i+1)} \sim p(x_1 | x_2^{(i)}, x_3^{(i)}, \cdots, x_m^{(i)})$
4: $\quad x_2^{(i+1)} \sim p(x_1 | x_1^{(i+1)}, x_3^{(i)}, \cdots, x_m^{(i)})$
5: $\quad \cdots$
6: $\quad x_m^{(i+1)} \sim p(x_n | x_1^{(i+1)}, x_2^{(i+1)}, \cdots, x_{m-1}^{(i+1)})$
7: **end for**

Otherwise, it stays at x (Algorithm 3, Line 7-9).

The Gibbs sampler can be seen as a special case of Metropolis-Hastings sampling where the acceptance rate is $\alpha = 1$ (Andrieu et al., 2003). The whole sampling mechanism is summarized in Algorithm 4 where m variables ($x_{1:m}$) can be sampled from m univariate conditional distributions (these univariate conditional distributions often belong to the family of standard distributions, such as Gaussian, Gamma, etc.) to approximate the joint distribution.

Even though MCMC approaches are often computationally expensive (since a good approximation of the target distribution may be achieved with hundreds of thousands of samples), their general and easily-implemented sampling mechanisms are still a good choice when the target distribution is highly complicated (Andrieu et al., 2003).

2.3 Conclusion

In this chapter, we briefly reviewed the technical details of Gaussian processes (GPs) and Bayesian estimation. Based on these techniques, we will introduce our contributions in the following sections to address a number of challenging difficulties in GPs and Bayesian estimation by performing *Interactions Between Gaussian Processes and Bayesian Estimation* in different directions.

Chapitre 3

Gaussian Processes for Bayesian Estimation

Bayesian estimation has been widely used in many domains such as robotics, bioinformatics and so on (Thrun et al., 2005; Deisenroth, 2010; Plagemann, 2008; Calderhead et al., 2008; Dondelinger et al., 2013). In general, the performance of Bayesian estimation is often dependent on the associated probabilistic models in the system. In this chapter, we take advantage of *Gaussian Processes for Bayesian Estimation*, where Gaussian processes (GPs) are used to learn the related data models in an elegant Bayesian nonparametric way, to improve the performance of Bayesian estimation in the following two contributions

1. The first contribution is an adaptive nonparametric particle filter (Wang and Chaib-draa, 2012b) in which we incorporated a GP trained optimal proposal into a KLD-Sampling particle filter to improve the accuracy and efficiency of recursive state estimation.
2. The second contribution is to use a GP for Bayesian parameter estimation in ordinary differential equations (ODEs) (Wang and Barber, 2014) where a novel GP-ODE generative model was proposed by directly linking the derivatives of the states to the observations in order to accelerate the efficiency and accuracy of parameter estimation in ODE systems.

3.1 An Adaptive Nonparametric Particle Filter for State Estimation

State estimation in dynamic systems refers to a filtering problem in which the latent states are recursively filtered out by using the history of the observations. Particle filtering is one of the most popular techniques for state estimation since it suitably interprets the multi-modal posteriors via sequential Monte Carlo sampling mechanism (Doucet et al., 2000b, 2001; Chen, 2003; Cappé et al., 2007). However, several difficulties of the traditional particle filter block

the accuracy and efficiency of state estimation in practice (Doucet et al., 2000b; Fox, 2001; Plagemann et al., 2007).

In Wang and Chaib-draa (2012b), we proposed a novel adaptive nonparametric particle filter where, inspired by *Gaussian Processes for Bayesian Estimation*, we incorporated a Gaussian process (GP) based proposal into a KLD-Sampling particle filter so that one can improve the accuracy and efficiency of state estimation.

3.1.1 Literature Review

During the past decades, a number of filtering techniques have been investigated in order to understand the underlying evolution of real-world dynamical systems. One of the best-known filters is the Kalman Filter (KF), which is a minimum-mean-squared-error estimator for linear systems with Gaussian noises (Kalman, 1960). However, the performance of KF is often deteriorated since in practice many dynamic systems are nonlinear and/or non-Gaussian. Two popular variants of KF, i.e. extended Kalman filter (EKF) (Jazwinski, 1970) and unscented Kalman Filter (UKF) (Julier et al., 1995), have been investigated to approximately estimate the hidden states by using a Taylor expansion and an unscented transformation (UT). But still, the performance of EKF and UKF will be distorted when dynamical systems are highly nonlinear and/or non-Gaussian (Julier et al., 1995; Cappé et al., 2007; Deisenroth et al., 2009; Turner and Rasmussen, 2012).

Alternatively, particle filtering is an effective sequential Monte Carlo sampling approach for nonlinear and/or non-Gaussian dynamic systems. A detailed literature review can be found in Doucet et al. (2000b, 2001); Chen (2003); Cappé et al. (2007). A standard particle filter (also called sampling importance resampling (SIR) particle filter) typically consists of two sampling steps, i.e., important sampling and resampling at each iteration. In the importance sampling step, N_p particles are drawn from a proposal model which is chosen as the transition model $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$. Then these particles are weighted using the observation model $p(\mathbf{y}_t|\mathbf{x}_t)$. In the resampling step, particles are sampled again according to their weights in order to obtain N_p equally-weighted new particles for the next step.

However, there are several drawbacks in the above-mentioned sampling mechanism of the SIR particle filter. First of all, a SIR particle filter suffers from weight degeneracy where the variance of the particle weights becomes large after a few sampling iterations. Consequentially, the performance of state estimation may be poor since most of all the particles are wasteful with very small particle weights (Doucet et al., 2000b, 2001; Chen, 2003; Cappé et al., 2007). To some degree, the resampling mechanism alleviates this problem by replicating the particles with large weights. However, it does not address weight degeneracy due to the fact that resampling transforms weight degeneracy into another form - the impoverishment of particle diversity, i.e., many particles are originally from a few particles with large weights.

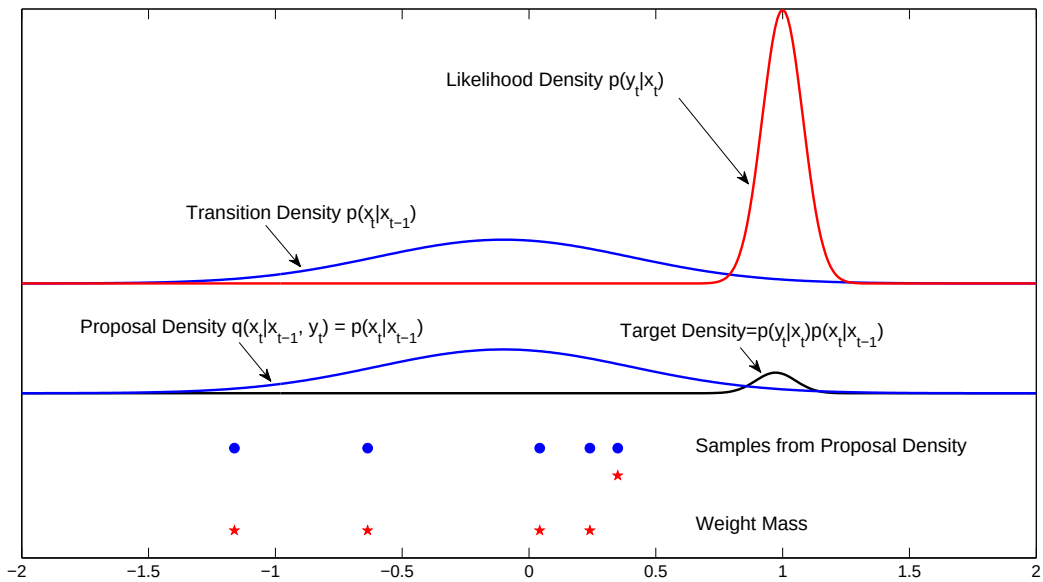


FIGURE 3.1: One-step importance sampling of SIR particle filter. The blue circles are the 5 particles (drawn from proposal). The red stars are the corresponding importance weights of these 5 particles (evaluated by likelihood). Note that there is only 1 non-zero weight (rightmost red star) in the weight mass.

In fact, the root cause of weight degeneracy is that the proposal $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$ does not contain the observation information $\mathbf{y}_t$, and this leads to the result that the drawn particles are located in low probabilistic regions of the true posterior (Doucet et al., 2000b, 2001; Chen, 2003; Cappé et al., 2007). This problem becomes even worse when the observation is accurate (the shape of the likelihood density is peaky), since the observation information in this case has a lot of impact on the estimation. As shown in Figure 3.1, we can clearly see that the weight mass is not a good approximation of the target density since there is only one non-zero-weight particle. The fundamental reason is that the current observation model $p(\mathbf{y}_t|\mathbf{x}_t)$ in this illustration contains the highly accurate information of the current state but the proposal $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$ does not consider it.

Therefore, it is crucial to design a good proposal for a SIR particle filter to achieve accurate state estimation. An extended Kalman particle filter and an unscented particle filter were proposed in van der Merwe et al. (2000) to approximate the optimal proposal via EKF and UKF. The estimation performance however lacks in robustness, especially when dynamic systems are highly nonlinear and non-Gaussian. To further improve the estimation accuracy, Plagemann et al. (2007) designed a nonparametric proposal trained by Gaussian process (GP) to address the weight degeneracy problem of the SIR particle filter. Additionally several new sampling frameworks such as particle learning (PL) (Carvalho et al., 2010) and particle Markov chain Monte Carlo (PMCMC) (Andrieu et al., 2010) were proposed to deal with weight degeneracy

to some degree. However, PL is only suitable to certain conditional dynamic models due to the use of conjugate priors (Carvalho et al., 2010) and PMCMC is often computationally expensive due to the MCMC sampling mechanism.

Another limitation of a SIR particle filter is the fixed number of particles. In practice, the true posterior of the latent state can vary vastly over time. In this case, the fixed number of particles will either lead to a poor efficiency (when the posterior is simple, many particles are wasteful) or a poor accuracy (when the posterior is complicated, the number of particles is not sufficient to capture the posterior). A KLD-Sampling particle filter was proposed in Fox (2001) to address this problem by adaptively adjusting the number of particles according to a Kullback-Leibler divergence (KLD) criterion. However, we note that KLD-Sampling particle filter still suffers from weight degeneracy since it is a modified version of SIR particle filter (Fox, 2001).

In Wang and Chaib-draa (2012b), we mainly address the above-mentioned two limitations in the SIR particle filter by incorporating a GP-trained proposal into the KLD-sampling particle filter. On one hand, we draw particles from a GP learned proposal in which the important observation information significantly improves the quality of the drawn particles. On the other hand, the number of particles is adaptively learned according to KLD. Consequentially, both perspectives in our resulting method are complementary to improve the performance of state estimation.

3.1.2 GP Based Proposal Density

In order to address the weight degeneracy problem of a SIR particle filter, the sampling proposal has to be optimal (Doucet et al., 2000b)

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t).$$

However, this optimal proposal is actually unknown. Hence, we propose to learn it. Furthermore, compared to parametric methods, nonparametric methods are more flexible to capture complicated dynamic phenomena. Following Plagemann et al. (2007), we hence use a Gaussian process (GP) to learn this optimal proposal.

Concretely, the optimal proposal $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t)$ can be modeled as a nonlinear relationship between $[\mathbf{x}_{t-1}, \mathbf{y}_t]$ and $\mathbf{x}_t$,

$$\mathbf{x}_t = \mathbf{g}(\mathbf{x}_{t-1}, \mathbf{y}_t) + \epsilon,$$

where $\mathbf{g}$ are unknown nonlinear functions that are learned by GPs[1], and the noise is $\epsilon \sim$

1. For simplicity, in the experiment we use independent GPs for each dimension of $\mathbf{g}$. One can also perform multi-output GP approaches (Boyle and Frean, 2004; Bonilla et al., 2007; Alvarez and Lawrence, 2008) to model $\mathbf{g}$ in a multi-task learning framework.

Algorithm 5 KLD-Sampling Particle Filter.

```
Initialize ε and δ
for t = 1 to T do
    Setting N_p = 0, k = 0
    while N_p ≤ N_f = (1/(2ε)) χ^2_{k-1,1-δ} (Equation 3.1) do
        Sampling a particle x_{t-1}^{(i)} with the normalized weights at t − 1
        Sampling a particle x_t^{(N_p)} ∼ p(x_t|x_{t-1}^{(i)})
        Calculating its weight according to p(y_t|x_t^{(N_p)})
        if x_t^{(N_p)} falls into an empty bin B then
            k = k + 1
            Setting the bin B non-empty
        end if
        N_p = N_p + 1
    end while
    return x_t^{(i)} (i = 1, 2, ...N_p)
end for
```

$\mathcal{N}(\mathbf{0}, \sigma_\epsilon^2 \mathbf{I})$. Given a training data set [2]

$$Data = \{[\mathbf{x}_{t-1}^{training}, \mathbf{y}_t^{training}], \mathbf{x}_t^{training}\}_{t=1}^{T},$$

we can directly use GP regression in Chapter 2 to obtain the optimal proposal that is used for filtering,

$$q(\mathbf{x}_t^{filtering} | \mathbf{x}_{t-1}^{filtering}, \mathbf{y}_t^{filtering}, Data).$$

As mentioned in Plagemann et al. (2007), a SIR particle filter with a GP based proposal improves the quality of the particles and thus improves the estimation accuracy.

3.1.3 KLD-Sampling Particle Filtering

Now we focus on the other limitation of a SIR particle filter, the fixed number of particles. In fact the number of particles should be adaptively changed over time according to the complexity of posterior density, i.e., if the posteriors at some steps are highly non-Gaussian, we need more particles to capture the important probabilistic regions and vice versa. That is the motivation of KLD-Sampling particle filtering in which the following Kullback-Leibler divergence (KLD) is used to learn the number of the particles over time (Fox, 2001)

$$KL[\hat{p}(\mathbf{x}) \| p(\mathbf{x})] = \sum_{\mathbf{x}} \hat{p}(\mathbf{x}) \log(\frac{\hat{p}(\mathbf{x})}{p(\mathbf{x})}),$$

where $\hat{p}(\mathbf{x})$ is the discrete estimation of $p(\mathbf{x})$.

2. This training data set is associated with the latent states, hence in practice we can perform a standard SIR particle filter as a preprocessing procedure to collect the latent states to construct this data set.

As mentioned in Fox (2001), for any discrete $p(\mathbf{x})$ with k different bins, the number of samples from its maximum likelihood estimation $\hat{p}(\mathbf{x})$ has to be

$$N_f = \frac{1}{2\varepsilon}\chi^2_{k-1,1-\delta} \tag{3.1}$$

in order to ensures that KLD between $p(\mathbf{x})$ and $\hat{p}(\mathbf{x})$ is smaller than ε with $1-\delta$ confidence. In Equation (3.1), χ^2_{k-1} is the chi-square distribution with $k-1$ degrees of freedom.

According to Equation 3.1, we can adaptively learn the number of particles which is strongly associated with the complexity of the posterior. However, true posteriors in the filtering problem are actually unknown, i.e., the number of bins k is unknown. Here we follow the strategy in Fox (2001); Thrun et al. (2005) to make k incremental. Concretely, for each filtering iteration, a new particle is first drawn and weighted using SIR manner (Algorithm 5, Line 5-7). Then, the number of bins k is replaced with $k+1$ if this particle falls into a new bin B (Algorithm 5, Line 8-11). Finally, the number of the particles N_p is replaced by N_p+1. These steps repeat in this filtering iteration until N_p exceeds N_f in Equation 3.1. The whole KLD-Sampling particle filter is summarized in Algorithm 5.

3.1.4 GP based KLD-Sampling Particle Filter

On one hand, a GP based SIR particle filter (Plagemann et al., 2007) alleviates weight degeneracy by learning the optimal proposal. But the fixed number of particles will reduce the estimation performance. On the other hand, KLD-Sampling particle filter (Fox, 2001) adaptively learns the number of particles to capture the time-varying posteriors. But weight degeneracy deteriorates its estimation performance since the particles are drawn from the transition model without observation information.

Therefore, we propose a hierarchical framework to combine these two methods together so that we can take advantage of their merits to improve accuracy and efficiency of state estimation. Specifically, we incorporate the GP based proposal into the KLD-Sampling particle filter. One benefit is from the optimal proposal learned by a GP due to the fact that the sampled particles are more likely located at the high probabilistic regions of the true posterior. The other benefit is from the KLD-sampling mechanism in which the number of the particles can be adaptively learned according to the complexity of true posteriors. Consequentially, in our resulting framework both benefits positively help each other to improve the performance of state estimation with the flexible number of high-quality particles.

3.1.5 Experiments

In order to show the effectiveness of our adaptive nonparametric particle filter, we compare our method to the standard SIR particle filter, KLD-Sampling particle filter and SIR particle filter with GP learned proposal by using the following two simulated models, i.e., an univariate nonstationary growth model and a two-link robot arm model.

Table 3.1: RMSE Comparison (Univariate Nonstationary Growth Model).

Particle Filter (PF)	Particle Number	RMSE
SIR PF	10	2.0183
SIR PF	100	1.7209
KLD-Sampling PF	average 45	1.6373
Gaussian Process Based PF	10	1.5775
our Proposed PF	average 42	**1.4348**

Univariate Nonstationary Growth Model

Our first simulation experiment is based on a benchmark univariate nonstationary growth model that has been widely investigated in the Bayesian filtering domain (Cappé et al., 2007; Andrieu et al., 2010),

$$\begin{aligned} x_t &= \frac{x_{t-1}}{2} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2t) + w_t, \\ y_t &= 0.05x_t^2 + v_t, \end{aligned}$$

where the system and observation noise are assumed to be Gaussian $w_t \sim \mathcal{N}(0, 10)$, $v_t \sim \mathcal{N}(0, 1)$ respectively. The prior for initial state is $x_0 \sim \mathcal{N}(0, 10)$. Time interval is set to 0.01 and the terminal time is 0.5. Hence, the objective posterior is $p(x_{0:50}|y_{0:50})$. In our experiment, both ε and δ are set to 0.05. The range of the bins is $-50 : 10 : 50$. Furthermore, the covariance function of GP is chosen to be squared exponential and we collected a training data set, which consists of 50 training data points at each step, to learn a GP based proposal.

We first show the qualitative performance of state estimation in Figure 3.2 where our adaptive non-parametric particle filter works well since the estimated state closely overlaps the ground truth with a compact confidence interval.

Next, we evaluate the quantitative performance of our approach based on root mean squared error (RMSE). The comparison results with other related particle filters are shown in Table 3.1. As expected, the performance of the 10-particle SIR particle filter is poor due to the fact that the blindly drawn particles (from the transition model) introduce a heavy weight degeneracy. In order to improve the accuracy, the number of particles in the SIR particle filter has to increase (here we set it to 100). However, the performance of the 100-particle SIR particle filter is sill inefficient, compared to the average-45-particle KLD-Sampling particle filter. The reason is that the fixed number of the SIR particle filter reduces its estimation performance. Additionally, the 10-particle SIR particle filter with the GP learned proposal outperforms the 10-particle standard SIR particle filter. This illustrates the fact that the GP learned proposal uses observation information to guide the particles to locate at the important regions of the true posterior.

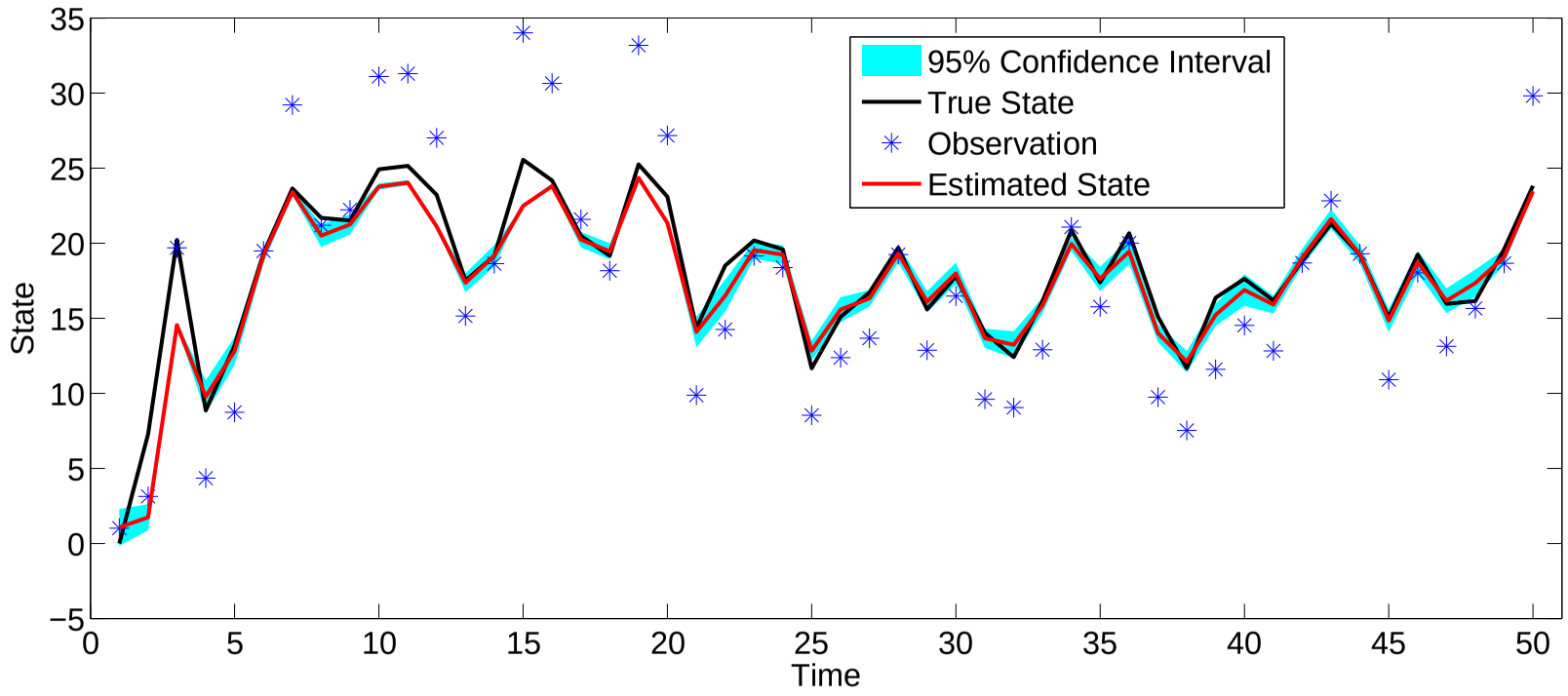


FIGURE 3.2: State estimation by our adaptive nonparametric particle filter.

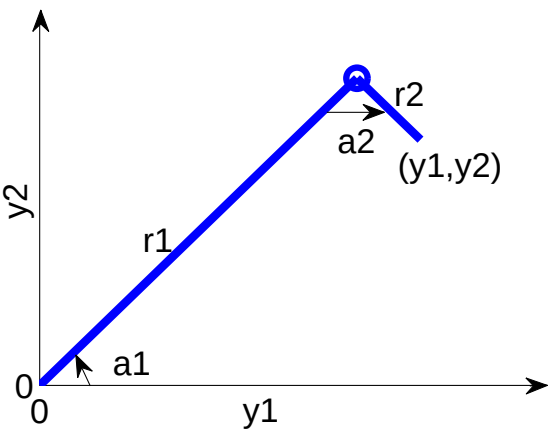


FIGURE 3.3: Two-link Robot Arm.

To sum up, our adaptive nonparametric particle filter obtains the best estimation performance with only average 42 particles. Compared to the KLD-Sampling particle filter, our approach incorporates observation information into the proposal (using a GP). Compared to the GP based SIR particle filter, our approach takes advantage of the KLD-sampling mechanism to adaptively learn the number of the particles according to the complexity of the true posterior. Both perspectives are complementary to make our resulting approach an efficient and accurate state estimator.

Two-link Robot Arm

Our second simulation experiment is based on a two-link robot arm model (Chen, 2003). As shown in Figure 3.3, the kinematic model of this robot is

$$\begin{aligned} y_1 &= r_1 \cos(a_1) - r_2 \cos(a_1 + a_2), \\ y_2 &= r_1 \sin(a_1) - r_2 \sin(a_1 + a_2), \end{aligned}$$

where $r_1 = 0.8$, $r_2 = 0.2$, the limited regions of the angles a_1 and a_2 are $a_1 \in [0.3, 1.2]$ and $a_2 \in [\pi/2, 3\pi/2]$, (y_1, y_2) is the cartesian position of the end-effector.

We mainly address an inverse kinematic problem where the angles a_1 and a_2 (latent states) are estimated to achieve a desired cartesian position (y_1, y_2) (observations). This problem is particularly important in motion planning where the robot is assigned to move in a desired trajectory. Following (Chen, 2003), we choose a simple state space model,

$$\begin{aligned} \mathbf{x}_t &= \mathbf{x}_{t-1} + \mathbf{w}_{t-1}, \\ \mathbf{y}_t &= \begin{bmatrix} r_1 cos(a_{1,t}) - r_2 cos(a_{1,t} + a_{2,t}) \\ r_1 sin(a_{1,t}) - r_2 sin(a_{1,t} + a_{2,t}) \end{bmatrix} + \mathbf{v}_t, \end{aligned}$$

where $\mathbf{x}_t = [a_{1,t} \;\; a_{2,t}]^T$ and $\mathbf{y}_t = [y_{1,t} \;\; y_{2,t}]^T$, $\mathbf{w}_{t-1} \sim \mathcal{N}(\mathbf{0}, diag\{0.02^2, \; 0.2^2\})$ and $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, diag\{0.001^2, \; 0.001^2\})$. Notice that the measurement of the observations is very accurate

since the variance of the observation noise is quite small. Additionally, the prior of initial state is set to be the same as the system noise. The robot arm moves 12 time steps, thus the objective of state estimation is $p(\mathbf{x}_{0:12}|\mathbf{y}_{0:12})$. Both ε and δ are set to 0.1, the range of the bins is $0.2 : 0.3 : 0.8$ for a_1 and $0.2\pi : 0.2\pi : 1.6\pi$ for a_2. Moreover, the covariance function of GP is chosen to be squared exponential and we collected a training set, which is composed of 300 training data points at each time step, to train a GP based proposal.

Similarly to our experiments in the univariate nonstationary growth model, we first evaluate the qualitative performance of our approach for this robot arm. From Figure 3.4 and 3.5, we can see that our approach successfully captures the latent states a_1 and a_2 over time. Additionally, an illustration of state estimation at each time step is shown in Figure 3.6 where our method outperforms the SIR particle filter in most of the filtering steps.

Then we show the quantitative comparison of different particle filters in Table 3.2. As expected, 20-particle SIR particle filter performs worst among all the methods. There are two main reasons. **Firstly**, the SIR particle filter heavily suffers from weight degeneracy in this experiment. As we mentioned before, the observation measurement in this experiment is highly accurate. However, the proposal in SIR particle filter is the state transition model which does not contain the important observation information. Hence the blindly drawn particles are most likely located at the low probabilistic regions of the true posterior, and this restricts the performance of the SIR particle filter. That is also why the 20-particle GP learned SIR particle filter performs better than the 20-particle standard SIR particle filter. **Secondly**, the number of particles in the SIR particle filter is fixed. From Figure 3.7, it is clearly shown that 20 particles in the SIR particle filter is not adequate since the KLD-Sampling particle filter adaptively learns average 25 particles to get better estimation. The only way to get a smaller error for the standard SIR particle filter is to increase the number of the particles (here we set it to 150). However, the 150-particle SIR particle filter is still worse than the KLD-Sampling particle filter. This illustrates that it is important to learn the number of the particles according the complexity of the posterior.

To sum up, our adaptive nonparametric particle filter combines the merits of the KLD-Sampling mechanism and the GP based proposal to get the best estimation with the smaller number of particles.

3.1.6 Summary

Based on *Gaussian Processes for Bayesian Estimation*, we proposed an adaptive nonparametric particle filter in Wang and Chaib-draa (2012b). Firstly, the optimal proposal was learned by a GP so that the drawn particles are located at the important probabilistic regions of the true posterior. Then, we incorporated this GP based proposal into a KLD-Sampling particle filter in which the number of particles is adaptively learned according to the complexity of

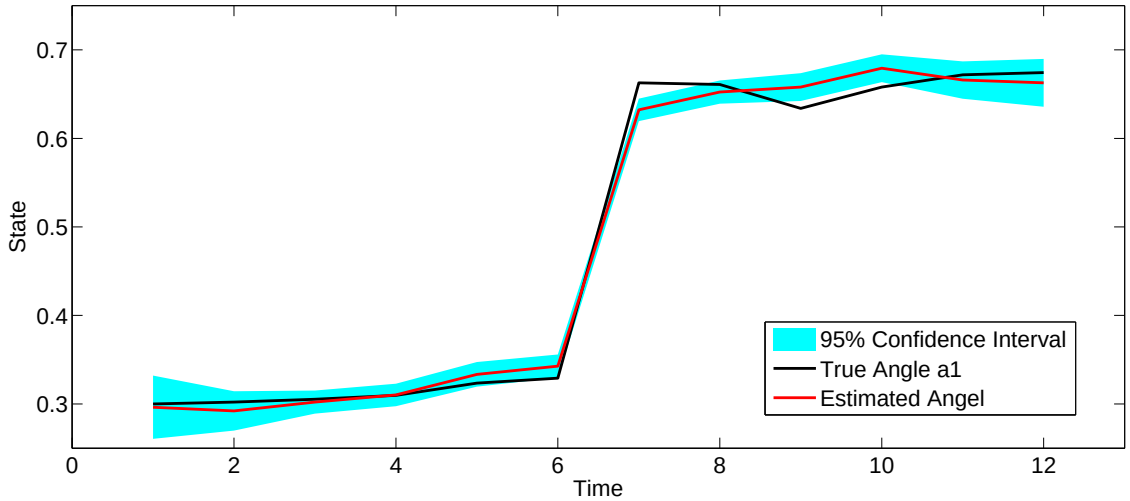


FIGURE 3.4: a_1 estimation by our adaptive nonparametric particle filter.

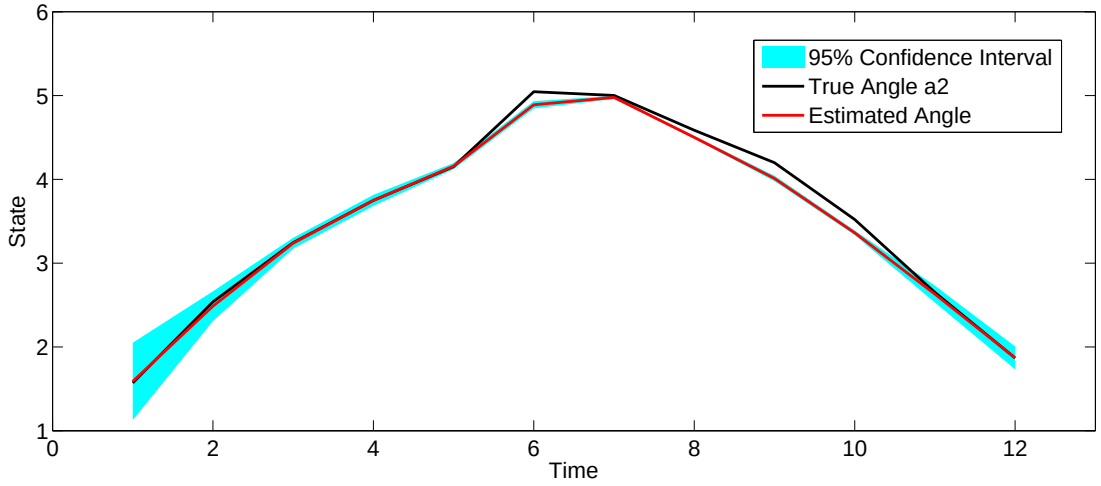


FIGURE 3.5: a_2 estimation by our adaptive nonparametric particle filter.

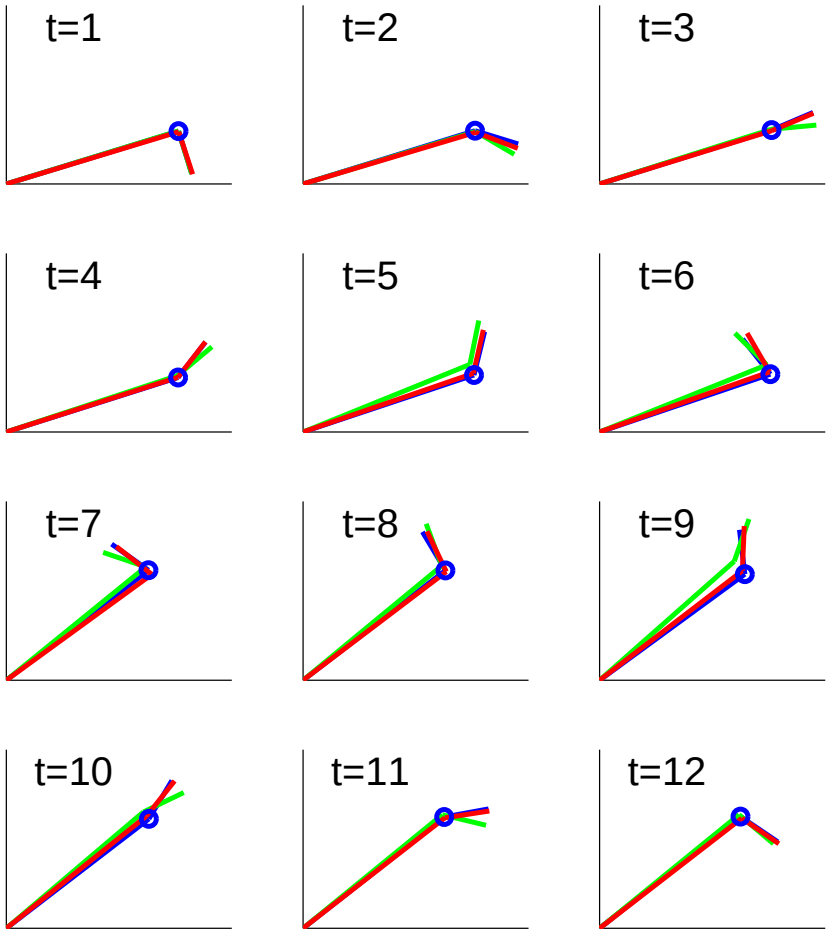


FIGURE 3.6: Two-link robot arm position over time. The blue line with circle is the true state, the red line is the estimated state by our proposed method, the green line is the estimated state by SIR particle filter. It is shown that our proposed approach is better than SIR particle filter.

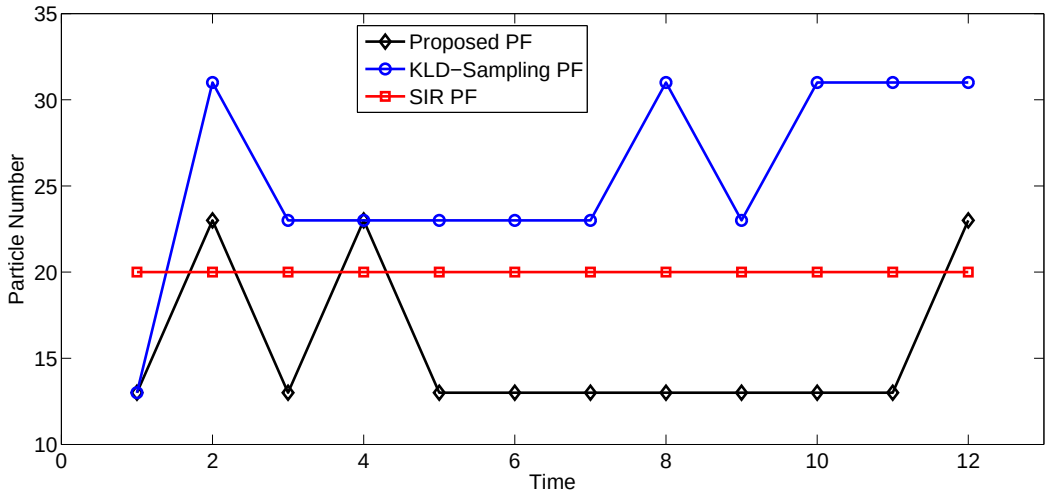


FIGURE 3.7: The number of the particles as a function of time steps.

TABLE 3.2: RMSE Comparison (Robot Arm).

Particle Filter (PF)	Particle Number	RMSE for a_1	RMSE for a_2
SIR PF	20	0.0378	0.1711
SIR PF	150	0.0207	0.1258
KLD-Sampling PF	average 25	0.0182	0.1140
Gaussian Process Based PF	20	0.0177	0.1085
Our Proposed PF	average 15	**0.0154**	**0.0933**

the time-varying posteriors. Our resulting framework takes advantage of the merits in both approaches to improve the accuracy and efficiency of state estimation.

In the following, we will use *Gaussian Processes for Bayesian Estimation* to solve a Bayesian parameter estimation problem in ordinary differential equations (Wang and Barber, 2014).

3.2 Gaussian Processes for Bayesian Estimation in Ordinary Differential Equations

Bayesian parameter estimation in coupled ordinary differential equations (ODEs) is challenging due to the high computational cost of numerical integration. In gradient matching approaches, a separate data model is introduced with the property that its gradient may be calculated easily. Parameter estimation is then achieved by requiring consistency between the gradients computed from the data model and those specified by the ODE. In Wang and Barber (2014), we proposed a Gaussian process (GP) based model that directly links state derivative information

with system observations, simplifying previous approaches and improving estimation accuracy.

3.2.1 Introduction

Ordinary differential equations (ODEs) are continuous time models with the interaction between variables described by $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \theta)$, for vector $\mathbf{x}$ and vector output function $\mathbf{f}$. The task is to estimate any unknown parameters θ of the ODEs by fitting them to observed data collected at a set of discrete observation times, $t_1, \ldots, t_T$. A principled approach to this problem is to first numerically integrate the ODEs for a given value of θ and initial value $\mathbf{x}_0$ to obtain a vector of values $\mathbf{X} \equiv [\mathbf{x}(t_1), \mathbf{x}(t_2), \ldots, \mathbf{x}(t_T)]$. Parameter estimation is achieved by finding θ such that $\mathbf{X}$ closely matches the observed data. However, numerical integration is computationally demanding, rendering this scheme impractical in all but the smallest systems, see (Vyshemirsky and Girolami, 2008) for example.

In gradient matching, explicit numerical integration can be avoided by considering an alternative model of the data, $\mathbf{x}(t) = \mathbf{g}(t, \phi)$. Given this fit to the data, one can compute the gradients of the fitted function at the observed timepoints, $\dot{\mathbf{x}}(t) = \dot{\mathbf{g}}(t, \phi)$. Gradient matching estimates parameters θ of the ODE and parameters ϕ of the fitted function $\mathbf{g}$ by requiring that the gradients in both models are consistent at the observed timepoints. A review of this class of approaches can be found in Ramsay et al. (2007).

However, as described in Calderhead et al. (2008), previous gradient matching approaches provided only limited point-parameter estimates or can prove numerically inconsistent. Recently, Gaussian Processes (GPs) have been considered as data models within the gradient matching framework (Calderhead et al., 2008; Dondelinger et al., 2013) , due to the fact that GPs provide a distribution over fitted functions and associated gradients. Concretely, Calderhead et al. (2008) proposed a hierarchical sampling mechanism where GP parameters ϕ are sampled first to fit to the data, and then ODE parameters θ are inferred by using the sampled ϕ. The estimation accuracy of this approach is however limited by the lack of feedback from ODE parameter inference to GP parameter inference. To address this, Dondelinger et al. (2013) introduced bidirectional interaction between ODE and GP parameters in a joint distribution, demonstrating improved parameter estimation.

Even though all these GP approaches have similar computational complexity and can run up to two orders of magnitude faster than numerical integration, they lack a natural interpretation of the data generation process and thus restrict the estimation performance. In Wang and Barber (2014), we introduced a novel generative model, which directly links state derivatives to system observations using a GP, to simplify the previous GP approaches and improve the estimation accuracy with a similar computational cost. Additionally, compared to previous GP approaches, our resulting framework plays a more similar role to numerical integration.

ODE System Description

We consider continuous time dynamical systems in which the motions of K states $\mathbf{x}(t) \equiv [x_1(t), x_2(t), \ldots, x_K(t)]^T$ are represented by a set of K ODEs,

$$\dot{\mathbf{x}}(t) \equiv \frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), \theta),$$

where θ is a vector of parameters of the ODE. For notational convenience, we additionally define the state matrix $\mathbf{X} \equiv [\mathbf{x}(t_1), \mathbf{x}(t_2), \ldots, \mathbf{x}(t_T)]$ and k-th state sequence $\mathsf{x}_k \equiv [x_k(t_1), x_k(t_2), \ldots, x_k(t_T)]^T$. Given potentially noisy observations of $\mathbf{X}$ (see below), the task is to infer a posterior distribution over the parameters θ.

Observation Model

The T observations $\mathbf{Y} = [\mathbf{y}(t_1), \mathbf{y}(t_2), \ldots, \mathbf{y}(t_T)]$ are obtained from the states according to independent additive noise $\mathbf{y}(t) = \mathbf{x}(t) + \boldsymbol{\epsilon}(t)$ where the noise for the k-th state, $k \in \{1, 2, \ldots, K\}$, is Gaussian, $\epsilon_k(t) \sim \mathcal{N}(0, \sigma_k^2)$. This gives then an observation model,

$$p_{OBS}(\mathbf{Y}|\mathbf{X}) = \prod_t p_{OBS}(\mathbf{y}(t)|\mathbf{x}(t)),$$

with $p_{OBS}(\mathbf{y}(t)|\mathbf{x}(t)) = \mathcal{N}(\mathbf{x}(t), \sigma^2\mathbf{I})$.

If unknown, the parameters of the observation model (σ in this case) form part of the parameters that need to be estimated. This is achieved by placing a prior over their values and incorporating these parameters into the model in the standard way. This step is unproblematic and, to avoid notational clutter, we drop these observation parameters as variables in the model descriptions below (they will however be included in the experiments).

Bayesian Numerical Integration

Given the ODE and an initial value, we can (in principle) numerically integrate the system [3]. For example ($K = 1$), one can discretize time in small intervals of τ, and then iteratively use $x'_{n+1} = x'_n + \tau f(x'_n, \theta)$ with $x'_0 = x_0$ until the desired end time to obtain a numerical value for the integrated path.

Hence, for a given initial value x_0 and ODE parameters θ, numerical integration can be considered as a procedure that can produce a deterministic distribution (in the case of ODE systems) $p(\mathsf{x}|x_0, \theta) = \delta\left(\mathsf{x} - \mathsf{x}'(x_0)\right)$ over the values of the state at the observation times. Here $\delta(\cdot)$ is the Dirac delta function. Then we put a prior on x_0 and θ so that we can draw samples from the following joint distribution,

$$p(\mathsf{y}, \mathsf{x}, x_0, \theta) = p_{OBS}(\mathsf{y}|\mathsf{x})p(\mathsf{x}|x_0, \theta)p(x_0)p(\theta),$$

3. In practice we use the Runge-Kutta method.

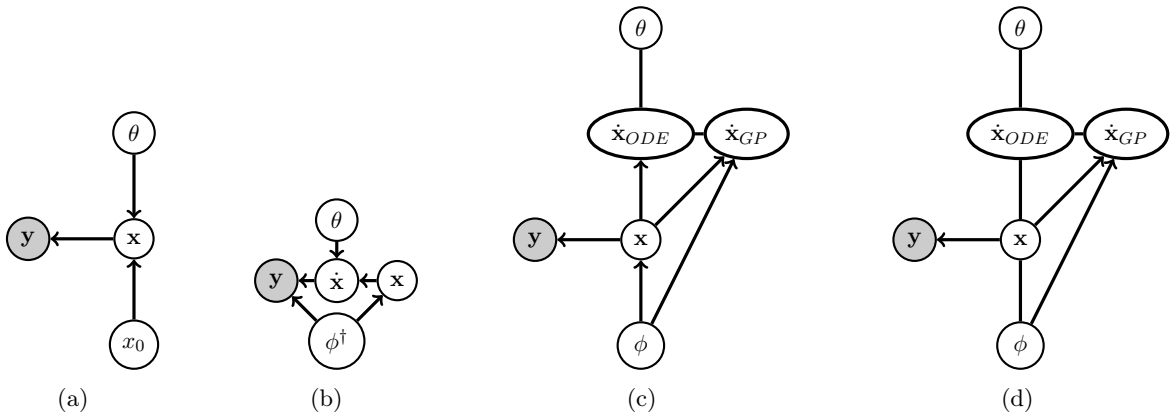


FIGURE 3.8: (a) Numerical integration with an initial term x_0. (b) Our GP-ODE approach corresponds to a generative belief network. (c) Calderhead et al. (2008) approach, which is based on a form of compatibility function, expressed as a chain graph. (d) The chain graph of the Dondelinger et al. (2013) approach uses a modified compatibility function. Note that the difference between (c) and (d) is that in (d) the links $\mathbf{x} - \dot{\mathbf{x}}_{ODE}$ and $\phi - \mathbf{x}$ are undirected, reflecting the different normalisation requirement. We discuss the approaches of Calderhead et al. (2008) and Dondelinger et al. (2013) in Subsection 3.2.4.

to estimate $p(\theta, x_0|\mathsf{y})$.

Even though this procedure of Bayesian numerical integration can produce excellent results (Vyshemirsky and Girolami, 2008), the computational cost is prohibitive in large models due to the fact that numerical integration needs to be carried out for every value of θ and x_0 of interest (Calderhead et al., 2008).

3.2.2 The GP-ODE generative model

As an alternative to explicit Bayesian numerical integration, we propose the following generative model over states $\mathbf{X}$, their derivatives $\dot{\mathbf{X}}$, observations $\mathbf{Y}$ and remaining parameters using a simple belief network shown in Figure 3.8b,

$$p(\mathbf{Y}, \mathbf{X}, \dot{\mathbf{X}}, \phi^\dagger, \theta) = p(\theta)p(\phi^\dagger)p_{GP}(\mathbf{Y}|\dot{\mathbf{X}}, \phi^\dagger)p_{ODE}(\dot{\mathbf{X}}|\mathbf{X}, \theta)p_{GP}(\mathbf{X}|\phi^\dagger), \tag{3.2}$$

where $\phi^\dagger \equiv (\mathbf{x}_0, \phi)$. To generate data from this model we first sample parameters $\phi^\dagger$, θ from their priors and then a state $\mathbf{X}$ from the GP prior $p_{GP}(\mathbf{X}|\phi^\dagger)$. A state derivative is subsequently obtained by sampling from $p_{ODE}(\dot{\mathbf{X}}|\mathbf{X}, \theta)$. Finally, given these state derivatives $\dot{\mathbf{X}}$, observations $\mathbf{Y}$ are generated by sampling from the GP $p_{GP}(\mathbf{Y}|\dot{\mathbf{X}}, \phi^\dagger)$. In this way we combine a smoothness prior assumption on the state $\mathbf{X}$ together with derivative information obtained from the ODE in a single generative model [4].

4. It is natural to consider forming the joint $p(y, x, \dot{x})$ as $p_{OBS}(y|x)p_{ODE}(\dot{x}|x)p_{GP}(x)$. However, the marginal $p(y, x) = p_{OBS}(y|x)p_{GP}(x)$ is then vacuous, containing no contribution from the ODE. All models, including

Encoding the ODE : $p_{ODE}(\dot{\mathbf{X}}|\mathbf{X},\theta)$

The temporal evolution of the ODE is encoded in the distribution $p_{ODE}(\dot{\mathbf{X}}|\mathbf{X},\theta)$. In the deterministic ODE case[5], it will simply be a delta function distribution

$$p_{ODE}(\dot{\mathbf{X}}|\mathbf{X},\theta) = \delta(\dot{\mathbf{X}} - \mathbf{f}(\mathbf{X},\theta)) \equiv \prod_t \delta\left(\dot{\mathbf{x}}(t) - \mathbf{f}(\mathbf{x}(t),\theta)\right).$$

Prior on latent state : $p_{GP}(\mathbf{X}|\phi^\dagger)$

The GP prior assumes that each state dimension is a priori independent $p_{GP}(\mathbf{X}|\phi^\dagger) = \prod_k p_{GP}(\mathsf{x}_k|\phi_k^\dagger)$, with $p_{GP}(\mathsf{x}_k|\phi_k^\dagger)$ formed from a GP with mean function[6] $\mu_{\phi_k}(t)$ and covariance function $c_{\phi_k}(t,t')$.

Implicit Integration : $p_{GP}(\mathbf{Y}|\dot{\mathbf{X}},\phi^\dagger)$

The term $p_{GP}(\mathbf{Y}|\dot{\mathbf{X}}) = \prod_k p_{GP}(\mathsf{y}_k|\dot{\mathsf{x}}_k)$ (dropping parameter dependencies on the r.h.s for compactness of notation) plays a key role in our GP-ODE model since it specifies how to implicitly integrate a given state derivative curve to arrive at a distribution over observations according to

$$p_{GP}(\mathsf{y}_k|\dot{\mathsf{x}}_k) = \int p_{OBS}(\mathsf{y}_k|\mathsf{x}_k) p_{GP}(\mathsf{x}_k|\dot{\mathsf{x}}_k) d\mathsf{x}_k.$$

Since differentiation is a linear operation, the derivative of a GP is also a GP (Solak et al., 2002). Consequentially, the joint distribution $p_{GP}(\mathsf{y}_k,\dot{\mathsf{x}}_k)$ is a Gaussian distribution which is constructed by using the mean functions,

$$\bar{y}_k(t) = \bar{x}_k(t) = \mu_\phi(t), \;\; \dot{\bar{x}}_k(t) = \partial\mu_\phi(t)/\partial t,$$

and covariance functions,

$$\begin{aligned}
cov(\dot{x}_k(t),\dot{x}_k(t')) &= \frac{\partial^2 c_{\phi_k}(t,t')}{\partial t \partial t'} - \frac{\partial \mu_{\phi_k}(t)}{\partial t}\frac{\partial \mu_{\phi_k}(t')}{\partial t'}, \\
cov(\dot{x}_k(t),x_k(t')) &= \frac{\partial c_{\phi_k}(t,t')}{\partial t} - \mu_{\phi_k}(t')\frac{\partial \mu_{\phi_k}(t)}{\partial t}, \\
cov(y_k(t),\dot{x}_k(t')) &= cov(x_k(t),\dot{x}_k(t')), \\
cov(y_k(t),y_k(t')) &= c_{\phi_k}(t,t') + \sigma^2\delta(t-t').
\end{aligned}$$

Figure 3.8b,c,d, are 'incorrect' compared to the true model Figure 3.8a; the challenge is to combine aspects of numerical integration with GP and observation model that achieves coherent parameter estimation with reduced computational cost over explicit numerical integration.

5. We make this assumption throughout this contribution, and Gaussian additive noise would be straightforward to incorporate for the case of Gaussian SDEs.

6. There are different ways to define $p(\mathbf{x},\mathbf{x}_0)$. One approach is to express this as $p_{GP}(\mathbf{x}|\mathbf{x}_0)p(\mathbf{x}_0)$, which allows one to use the same prior as for Bayesian numerical integration model. In the experiments we more simply defined a joint GP, for each k, with mean $\mu_{\phi_k}(t)$ equal to the mean of the observed data, for all t. This is equivalent to defining a Gaussian prior on $\mathbf{x}_0$ with mean that of the observed data.

Then given the state derivatives, the observations are also Gaussian distributed [7]

$$p_{GP}(\mathsf{y}_k|\dot{\mathsf{x}}_k) \sim \mathcal{N}(\boldsymbol{\mu}_k^{y|\dot{x}}, \boldsymbol{\Sigma}_k^{y|\dot{x}}),$$

where

$$\begin{aligned}\boldsymbol{\mu}_k^{y|\dot{x}} &= \boldsymbol{\mu}_{\phi_k} + \mathbf{C}_{\phi_k}^{x\dot{x}}(\mathbf{C}_{\phi_k}^{\dot{x}\dot{x}})^{-1}\left(\dot{\mathsf{x}}_k - \dot{\bar{\mathsf{x}}}_k\right),\\ \boldsymbol{\Sigma}_k^{y|\dot{x}} &= \mathbf{C}_{\phi_k} + \sigma_k^2\mathbf{I} - \mathbf{C}_{\phi_k}^{x\dot{x}}(\mathbf{C}_{\phi_k}^{\dot{x}\dot{x}})^{-1}\mathbf{C}_{\phi_k}^{\dot{x}x},\end{aligned}$$

$\mathbf{C}_{\phi_k}^{\dot{x}\dot{x}}$, $\mathbf{C}_{\phi_k}^{\dot{x}x}$ and $\mathbf{C}_{\phi_k}^{x\dot{x}}$ are constructed using the above-mentioned covariance functions evaluated at the observation times $t_1, t_2, \ldots, t_T$.

Parameter Estimation

From Equation 3.2, the marginal distribution over observations, latent states and parameters is given by integrating out $\dot{\mathbf{X}}$,

$$\begin{aligned}p(\mathbf{Y}, \mathbf{X}, \phi^\dagger, \theta) =& p(\phi^\dagger)p(\theta)p_{GP}(\mathbf{X}|\phi^\dagger)\int p_{GP}(\mathbf{Y}|\dot{\mathbf{X}}, \phi^\dagger)p_{ODE}(\dot{\mathbf{X}}|\mathbf{X}, \theta)d\dot{\mathbf{X}}\\ =& p(\phi^\dagger)p(\theta)p_{GP}(\mathbf{X}|\phi^\dagger)p_{GP}(\mathbf{Y}|\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, \theta), \phi^\dagger),\end{aligned} \tag{3.3}$$

where in the deterministic ODE case, $p_{ODE}(\dot{\mathbf{X}}|\mathbf{X}, \theta) = \delta(\dot{\mathbf{X}} - \mathbf{f}(\mathbf{X}, \theta))$, the integral

$$\int p_{GP}(\mathbf{Y}|\dot{\mathbf{X}}, \phi^\dagger)p_{ODE}(\dot{\mathbf{X}}|\mathbf{X}, \theta)d\dot{\mathbf{X}}$$

is simply reduced to $p_{GP}(\mathbf{Y}|\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}, \theta), \phi^\dagger)$.

Estimation of the parameters and latent states $\mathbf{X}$ can be then carried out, for example, by sampling from the posterior $p(\mathbf{X}, \phi^\dagger, \theta|\mathbf{Y}) \propto p(\mathbf{Y}, \mathbf{X}, \phi^\dagger, \theta)$, see Subsection 3.2.3.

3.2.3 Inference in the GP-ODE model

There are a number of approaches one could take to draw samples from the GP-ODE posterior $p(\mathbf{X}, \phi^\dagger, \theta|\mathbf{Y})$ and our strategy was to choose the simplest that provides good results. Writing $\Phi = \left\{\phi^\dagger, \sigma\right\} = \{\mathbf{x}_0, \phi, \sigma\}$ for all the parameters of the GP and observation model, we sample from $p(\mathbf{X}, \Phi, \theta|\mathbf{Y})$ using a Gibbs procedure to produce a set of samples $\Phi^i, \theta^i, \mathbf{X}^i$. We initialize Φ^0, θ^0 at random and draw $\mathbf{X}^0 \sim p_{GP}(\mathbf{X}|\mathbf{Y}, \Phi^0)$. We subsequently draw samples, indexed by $i = 1 : L$ by alternately drawing from

1. $\theta^i, \Phi^i \sim p(\theta, \Phi|\mathbf{X}^{i-1}, \mathbf{Y})$
2. $\mathbf{X}^i \sim p(\mathbf{X}|\theta^i, \Phi^i, \mathbf{Y})$

We present a naive approach for drawing from these conditionals below [8].

7. For a Gaussian defined on joint variables $\mathbf{z} = (\mathbf{a}, \mathbf{b})$ with $p(\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_{z,z})$, the conditional is Gaussian with mean and covariance given from the block mean and covariances, $p(\mathbf{a}|\mathbf{b}) = \mathcal{N}\left(\boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{a,b}\boldsymbol{\Sigma}_{b,b}^{-1}(\mathbf{b} - \boldsymbol{\mu}_b), \boldsymbol{\Sigma}_{a,a} - \boldsymbol{\Sigma}_{a,b}\boldsymbol{\Sigma}_{b,b}^{-1}\boldsymbol{\Sigma}_{b,a}\right)$, see e.g. (Barber, 2012; Rasmussen and Williams, 2006).

8. More sophisticated sampling strategies could be considered. However for the benchmark experiments, these approaches have proved adequate.

Parameter sampling

We draw from $p(\theta^i, \Phi^i | \mathbf{X}^{i-1}, \mathbf{Y})$ using Gibbs sampling :

1. Set $\theta^{i,0} = \theta^{i-1}$,$\Phi^{i,0} = \Phi^{i-1}$
2. For $j = 1 : L_p$
   a) $\Phi^{i,j} \sim p(\Phi | \mathbf{X}^{i-1}, \theta^{i,j-1}, \mathbf{Y})$
   b) $\theta^{i,j} \sim p(\theta | \mathbf{X}^{i-1}, \Phi^{i,j}, \mathbf{Y})$
3. Set $\theta^i = \theta^{i,L_p}$, $\Phi^i = \Phi^{i,L_p}$

where these conditional distributions can be obtained from the joint distribution (Equation 3.3). Where there are multiple components of a parameter, we again use Gibbs sampling to obtain a univariate sample of a component conditioned on the remaining components. In the experiments we assume that the parameters take values from known discrete sets, in which case sampling from these conditionals is particularly straightforward.

State sampling

It is natural to consider drawing samples from $p(\mathbf{X}|\theta, \Phi, \mathbf{Y})$ using Metropolis-Hastings (similar to (Dondelinger et al., 2013)) with $p_{GP}(\mathbf{X}|\Phi, \mathbf{Y})$ as the proposal. However, in our experience, this results in poor mixing due to the curse of dimensionality. We therefore use Gibbs sampling in which we draw a state from $p(\mathbf{x}(t)|\mathbf{X}_{\setminus t}, \theta, \Phi, \mathbf{Y})$, where $\mathbf{X}_{\setminus t}$ are the states except for $\mathbf{x}(t)$, drawing samples in sequence from times $t \in \{t_1, \ldots, t_T\}$. To draw from $p(\mathbf{x}(t)|\mathbf{X}_{\setminus t}, \theta, \Phi, \mathbf{Y})$ we use either Metropolis-Hastings with proposal $p_{GP}(\mathbf{x}(t)|\mathbf{X}_{\setminus t}, \Phi, \mathbf{Y})$ or Gibbs sampling for each component of the vector $\mathbf{x}(t)$ based on discrete values[9]. After L_x sweeps through all timepoints, we obtain the new sample $\mathbf{X}^i$.

3.2.4 Relation to previous approaches

Gradient Matching

The approach in Calderhead et al. (2008) is based on matching gradients via what could be termed a 'compatability' function that is a product of the data model $p_{GP}(\dot{\mathbf{x}}|\mathbf{x}, \phi)$ (from GP) and the ODE model $p_{ODE}(\dot{\mathbf{x}}|\mathbf{x}, \theta)$ (for the case $K = 1$ and fixed σ for notational simplicity),

$$\omega(\dot{\mathbf{x}}, \mathbf{x}|\theta, \phi) \equiv p_{GP}(\dot{\mathbf{x}}|\mathbf{x}, \phi) p_{ODE}(\dot{\mathbf{x}}|\mathbf{x}, \theta),$$

This is used to define

$$p(\theta|\mathbf{x}, \phi) \propto p(\theta)\omega'(\mathbf{x}|\theta, \phi),$$

using the marginal compatibility

$$\omega'(\mathbf{x}|\theta, \phi) \equiv \int \omega(\dot{\mathbf{x}}, \mathbf{x}|\theta, \phi) d\dot{\mathbf{x}},$$

9. For the experiments, the Gibbs approach proved adequate.

with presumably the intention that this has high value when the gradient distributions overlap [10]. The marginal compatibility $\omega'(\mathbf{x}|\theta,\phi)$ is analytically computed since the terms under the integral are Gaussian. The authors modify the deterministic ODE by the addition of fictitious noise to give a Gaussian distribution for $p_{ODE}(\dot{\mathbf{x}}|\mathbf{x},\theta)$ with mean $\mathbf{f}(\mathbf{x},\theta)$. To ease comparison with our approach (the extension to the stochastic ODE case is trivial), we take the deterministic ODE case, for which the above reduces to

$$p(\theta|\mathbf{x},\phi) \propto p(\theta)p_{GP}(\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\theta)|\mathbf{x},\phi). \tag{3.4}$$

The joint distribution over observations, latent states and parameters is then defined as

$$p(\mathbf{y},\mathbf{x},\theta,\phi) = p_{OBS}(\mathbf{y}|\mathbf{x})p(\theta|\mathbf{x},\phi)p_{GP}(\mathbf{x}|\phi)p(\phi). \tag{3.5}$$

Inference is then achieved by sampling, conditioned on the observed sequence $\mathbf{y}$. The unknown normalisation term of (3.4) is a function of ϕ and thus makes direct Gibbs sampling from this posterior problematic. The approach taken in Calderhead et al. (2008) is to re-factorize the joint distribution in the form [11]

$$p(\mathbf{y},\mathbf{x},\theta,\phi) = p(\theta|\mathbf{x},\phi)p(\mathbf{x}|\phi,\mathbf{y})p(\phi|\mathbf{y})p(\mathbf{y}).$$

Conditioned on $\mathbf{y}$, ancestral sampling is then performed :

$$\begin{aligned}
&\phi \sim p(\phi|\mathbf{y}),\\
&\mathbf{x} \sim p(\mathbf{x}|\phi,\mathbf{y}) \propto p_{OBS}(\mathbf{y}|\mathbf{x})p_{GP}(\mathbf{x}|\phi),\\
&\theta \sim p(\theta|\mathbf{x},\phi).
\end{aligned}$$

Here, $p(\phi|\mathbf{y}) \propto p(\phi)\int p_{OBS}(\mathbf{y}|\mathbf{x})p_{GP}(\mathbf{x}|\phi)d\mathbf{x}$ for which the integral can be evaluated analytically. A disadvantage of this model is that the posterior $p(\phi|\mathbf{y})$ does not take the ODE system dynamics into consideration. Effectively, a GP is fitted to the data first (without knowledge of the system dynamics) and the parameters θ of the ODE are subsequently adjusted to best match the fitted GP.

The gradient matching approach can be defined as a chain graph (see for example (Koller and Friedman, 2009)) shown in Figure 3.8c with factors [12]

$$p_{OBS}(\mathbf{y}|\mathbf{x})p_{ODE}(\dot{\mathbf{x}}_{ODE}|\mathbf{x},\theta)p_{GP}(\dot{\mathbf{x}}_{GP}|\mathbf{x},\phi)\delta\left(\dot{\mathbf{x}}_{ODE} - \dot{\mathbf{x}}_{GP}\right)p_{GP}(\mathbf{x})p(\theta)p(\phi).$$

The undirected link between θ and $\dot{\mathbf{x}}_{ODE}$ is necessary to ensure that the variables $\dot{\mathbf{x}}_{ODE}$, $\dot{\mathbf{x}}_{GP}$, θ form a component of the chain graph. Marginalising this chain distribution over $\dot{\mathbf{x}}_{GP}$ and

10. The mathematical motivation for this is less clear. Given distributions p and q, their 'overlap' $\int p(x)q(x)dx$ is maximal when $q(x)$ is a delta distribution placing all its mass on the most likely state of $p(x)$; this is not necessarily the same as matching q to p.

11. Whilst this can be interpreted as generative model, this is unnatural since the term $p(\theta|\mathbf{x},\phi)$ means that the parameters of the ODE depend on the generated state $\mathbf{x}$.

12. This chain graph structure is the same for the trivial extension to the stochastic ODE case.

$\dot{\mathbf{x}}_{ODE}$ gives the marginal distribution,

$$p_{OBS}(\mathbf{y}|\mathbf{x})p_{GP}(\mathbf{x}|\phi)p(\phi)\frac{p(\theta)\omega'(\mathbf{x}|\theta,\phi)}{\int p(\theta)\omega'(\mathbf{x}|\theta,\phi)d\theta}, \tag{3.6}$$

which matches (3.5). We can also write this as

$$p_{OBS}(\mathbf{y}|\mathbf{x})p_{GP}(\mathbf{x}|\phi)p(\phi)p(\theta)m_{GM}(\mathbf{x}|\theta,\phi), \tag{3.7}$$

where the gradient matching function is defined as

$$m_{GM}(\mathbf{x}|\theta,\phi) \equiv \frac{\omega'(\mathbf{x}|\theta,\phi)}{\int p(\theta)\omega'(\mathbf{x}|\theta,\phi)d\theta}. \tag{3.8}$$

Adaptive Gradient Matching

Dondelinger et al. (2013) considered a modified gradient matching approach with joint distribution,

$$p(\mathbf{y},\mathbf{x},\dot{\mathbf{x}},\theta,\phi) \propto p_{OBS}(\mathbf{y}|\mathbf{x})p_{GP}(\mathbf{x}|\phi)\omega(\dot{\mathbf{x}},\mathbf{x}|\theta,\phi)p(\theta)p(\phi),$$

and marginal,

$$p(\mathbf{y},\mathbf{x},\theta,\phi) \propto p_{OBS}(\mathbf{y}|\mathbf{x})p_{GP}(\mathbf{x}|\phi)\omega'(\mathbf{x}|\theta,\phi)p(\theta)p(\phi).$$

Sampling is carried out using Metropolis-Hastings under a factorised proposal distribution $q(\mathbf{x},\theta,\phi) = q(\mathbf{x})q(\theta)q(\phi)$. A benefit of this approach is that the marginal,

$$p(\mathbf{y}|\phi) \propto \int p_{OBS}(\mathbf{y}|\mathbf{x})p_{GP}(\mathbf{x}|\phi)\int \omega'(\mathbf{x}|\theta,\phi)p(\theta)d\theta d\mathbf{x},$$

does depend on the ODE. Hence, in contrast to the approach in Calderhead et al. (2008), the parameters of the GP in this approach are influenced by the ODE (and vice versa). The marginal $p(\mathbf{y},\mathbf{x},\theta,\phi)$ is then given by the same expression as (3.7) but with the adaptive gradient matching function,

$$m_{AGM}(\mathbf{x}|\theta,\phi) \equiv \frac{\omega'(\mathbf{x}|\theta,\phi)}{\int p(\theta)p(\phi)p_{GP}(\mathbf{x}|\theta)\omega'(\mathbf{x}|\theta,\phi)d\theta d\mathbf{x}d\phi}.$$

The factors in the corresponding chain graph, Figure 3.8d, are the same as for the gradient matching method of Subsection 3.2.4. However, all variables except $\mathbf{y}$ form a component in the chain graph, giving the correct form for the marginal distribution on $p(\mathbf{y},\mathbf{x},\theta,\phi)$.

As for the gradient matching approaches (Calderhead et al., 2008; Dondelinger et al., 2013), there is no natural interpretation as a generative model of the data. On the contrary, no such issue arise in our GP-ODE generative model in which the coupling between GP and ODE parameters occurs through the implicit numerical integration mechanism to ensure agreement between the ODE and GP curves.

3.2.5 Experiments

We illustrate our framework on two benchmark systems, Lotka-Volterra and Signal Transduction Cascade in Dondelinger et al. (2013). To aid comparison, wherever possible, we have chosen the same parameter settings and priors as the original authors. Our main interest is to study the implications of the different joint distributions specified by the competing approaches. As such, we wish to make as similar as possible the sampling approaches for the three competing models in order to minimize differences due to different sampling strategies. To facilitate comparison we therefore used the same discretized sampling strategy for all methods. For the AGM and our GP-ODE approach we used Gibbs sampling for a discretized set of values, analogous to Subsection 3.2.3 and the hybrid Gibbs-MH scheme of Subsection 3.2.3 for state samples. The cost of drawing a single sample in all competing approaches is similar, scaling $O(KT^3)$ where K is the dimension of the state vector and T is the number of the observations. We stopped each sampling scheme (all written in Matlab) after a similar CPU time. Following (Dondelinger et al., 2013), we set $p(\theta)$ to a Gamma prior $Ga(4, 0.5)$, $p(\phi)$ to a uniform prior $U(0, 100)$ and $p(\sigma)$ to a Gamma prior $Ga(1, 1)$. For the sampling process, the standard deviations of the observation noise σ in both models are initialized as the ground truth. For comparison we ran the Bayesian Numerical Integration approach using the same discretized parameter values wherever possible.

Lotka-Volterra is an ecological system that is used to describe the periodical interaction between a prey species $[S]$ and a predator species $[W]$:

$$\frac{d[S]}{dt} = [S](\alpha - \beta[W]), \qquad \frac{d[W]}{dt} = -[W](\gamma - \delta[S]),$$

where $\theta = [\alpha, \beta, \gamma, \delta]^T$ and $\mathbf{x}(t) = [[S], [W]]$. The ground truth data are generated using numerical integration over the interval [0, 2] with $\theta = [2, 1, 4, 1]$ and initial state values $[S] = 5$, $[W] = 3$. The clean data are then sampled with the sampling interval 0.2. Finally clean data are corrupted with additive Gaussian noise $\mathcal{N}(\mathbf{0}, 0.5^2\mathbf{I})$ to form the observations $\mathbf{Y}$.

We chose the squared-exponential covariance function,

$$c_{\phi_k}(t, t') = \sigma_k^x \exp(-l_k(t - t')^2),$$

where $\phi_k = [\sigma_k^x, l_k]$. Assuming a common parameter across observation dimensions, the parameter vector ϕ is simplified to $[\sigma^x, l]$; we initialize it as $[1, 10]$. The ODE parameters are initialized as $\theta = [1.5, 0.5, 3.5, 0.5]$. We discretized the ODE parameters α, β, γ, δ over [1.5, 2.5], [0.5, 1.5], [3.5, 4.5], [0.5, 1.5] all with the interval 0.1 ; the parameter σ^x is discretized over the range [0.1,1] with interval 0.1 ; the lengthscale l is discretized over [5, 50] with interval 5 ; the standard deviation of the noise σ was discretized over [0.1, 1] with interval 0.1. The parameter x_0 was, for each state dimension k, discretized in the range $[0, 10]$ using 20 uniformly spaced bins.

After drawing ODE parameters θ from the posterior (see Table 3.2.5), we plot the numerically integrated curves (setting x_0 to the true value to aid visual comparison), see Figure 3.9. The 'best' method is that which most closely approximates the Bayesian Numerical Integration method of Subsection 3.2.1. For small noise levels (not shown), all three competing methods produce similar results ; however as the noise increases, the Adaptive Gradient Matching and Gradient Matching approaches diverge markedly from the Bayesian Numerical Integration approach, whilst the GP-ODE approach fairs well.

Signal Transduction Cascade is described by a 5-dimensional ODE system,

$$\begin{aligned}
\frac{d[S]}{dt} &= -k_1[S] - k_2[S][R] + k_3[RS], \\
\frac{d[S_d]}{dt} &= k_1[S], \\
\frac{d[R]}{dt} &= -k_2[S][R] + k_3[RS] + \frac{V[Rpp]}{Km + [Rpp]}, \\
\frac{d[RS]}{dt} &= k_2[S][R] - k_3[RS] - k_4[RS], \\
\frac{d[Rpp]}{dt} &= k_4[RS] - \frac{V[Rpp]}{Km + [Rpp]},
\end{aligned}$$

where $\theta = [k_1, k_2, k_3, k_4, V, Km]$ and $\mathbf{x}(t) = [[S], [S_d], [R], [RS], [Rpp]]^T$. The ground truth data are generated over the interval [0, 100] with $\theta = [0.07, 0.6, 0.05, 0.3, 0.017, 0.3]$ and initial state $[S] = 1$, $[S_d] = 0$, $[R] = 1$, $[RS] = 0$, $[Rpp] = 0$. Then the data are sampled at t= [0, 1, 2, 4, 5, 7, 10, 15, 20, 30, 40, 50, 60, 80, 100]. Finally the drawn samples are corrupted with additive Gaussian noise $\mathcal{N}(\mathbf{0}, 0.1^2\mathbf{I})$ to construct the noisy observations. The non-stationarity is captured by the covariance function [13],

$$c_{\phi_k}(t, t') = \sigma_k^x \arcsin\left(\frac{a_k + b_k t t'}{\sqrt{(a_k + b_k t^2 + 1)(a_k + b_k t'^2 + 1)}}\right),$$

where $\phi_k = [\sigma_k^x, a_k, b_k]$. The ODE parameters are initialized as $\theta = [0.05, 0.4, 0.03, 0.1, 0.015, 0.1]$. We discretized the ODE parameters k_1, k_2, k_3, k_4, V, Km over [0.05, 0.09], [0.4, 0.8], [0.03, 0.07], [0.1, 0.5], [0.015, 0.019], [0.1, 0.5] with the respective intervals 0.01, 0.1, 0.01, 0.1, 0.001, 0.1 ; the parameters σ^x, a, b over [0.1,0.9], [0.5, 2.5], [0.5, 2.5] with the respective intervals 0.2, 0.5, 0.5 ; the standard deviations of the noise σ over [0.06, 0.14] with the interval 0.02. The 5 components of x_0 were discretized in the intervals [0.5 1.5], [-0.1 0.1], [0.5 1.5], [-0.1 0.1], [-0.1 0.1] using 50 uniformly spaced bins. All three competing approaches were run for approximately 30mins CPU time. The results for all methods produce reasonable solutions and the corresponding reconstructions using numerical integration with parameters θ sampled from the respective posteriors are similar. For this reason we show only the ideal Bayesian Numerical Integration procedure and our GP-ODE method in Figure 3.10. In Table 3.2.5 we note

13. In contrast to the Lotka Volterra model, here we use a GP with separate hyperparameters for each state dimension due to the different length scales in each dimension.

the same pattern as for the Lotka-Volterra experiments, namely that the GP-ODE method closely matches the Bayesian Numerical Integration approach.

3.2.6 Summary

Bayesian numerical integration is an accurate but computationally prohibitive method for parameter estimation in ODEs due to the fact that explicit numerical integration is required to evaluate every sample of the parameter vector in the posterior. Whilst previous GP based approaches have demonstrated some success in circumventing the high computational cost, they are not natural generative models of the data.

In contrast, our GP-ODE method has a natural interpretation for data generation by using GP to directly link the state derivatives to the observations. It is conceptually the closest match amongst competing GP approaches to Bayesian numerical integration. Finally, we would like to thank Dirk Husmeier and Benn Macdonald for their helpful discussions.

3.3 Conclusion

In this chapter, we mainly worked on *Gaussian Processes for Bayesian Estimation* to address two challenging tasks from the perspective of Bayesian estimation.

In Wang and Chaib-draa (2012b), we proposed an adaptive nonparametric particle filter by incorporating a GP trained optimal proposal into a KLD-Sampling particle filter. On one hand, the GP based proposal allows us to draw high-quality particles from the important regions of the posterior. On the other hand, the KLD-Sampling mechanism allows us to adaptively learn the number of particles according to the complexity of the posteriors. Both perspectives positively help each other to make our resulting estimator accurate and efficient.

In Wang and Barber (2014), we proposed a novel GP-ODE generative model for Bayesian parameter estimation in ODE systems. By directly connecting the derivative of the states to the observations, our GP-ODE is a more natural generative model than the previous GP based approaches and thus improves the estimation accuracy.

In the next chapter, we concentrate on the other direction, *Bayesian Estimation for Gaussian Processes*, to address the challenging tasks from the perspective of GP.

Parameter	True Value	Num-Bayes	GP-ODE	AGM	GM
α	2	2.2680±0.1853	2.2380±0.1953	1.9480±0.2819	1.6429±0.2488
β	1	1.2070±0.1249	1.1490±0.1910	1.1750±0.1909	0.9242±0.2256
γ	4	3.8330±0.2640	3.9590±0.3002	3.8390±0.2947	3.6449±0.2223
δ	1	0.9850±0.0857	0.9860±0.0995	1.2320±0.1638	1.1737±0.1803
k_1	0.07	0.0683±0.0122	0.0747±0.0130	0.0771±0.0130	0.0762±0.0130
k_2	0.6	0.6800±0.0876	0.6230±0.1246	0.5460±0.1259	0.5632±0.1256
k_3	0.05	0.0548±0.0125	0.0530±0.0135	0.0593±0.0111	0.0594±0.0115
k_4	0.3	0.2290±0.0498	0.2960±0.0281	0.3750±0.0999	0.3754±0.1051
V	0.017	0.0177±0.0012	0.0177±0.0014	0.0172±0.0015	0.0173±0.0014
Km	0.3	0.3860±0.0792	0.4220±0.0690	0.4090±0.0911	0.4186±0.0953

TABLE 3.3: ODE Parameter Estimation for both the Lotka Volterra (upper) and Signal Transduction Cascade (lower) ODEs. In each we plot the mean and standard deviation of the θ parameter samples from the respective competing approaches, namely the 'ideal' Bayesian Numerical Integration procedure, our GP-ODE approach, the Adaptive Gradient Matching approach (Dondelinger et al., 2013) and the Gradient Matching approach (Calderhead et al., 2008).

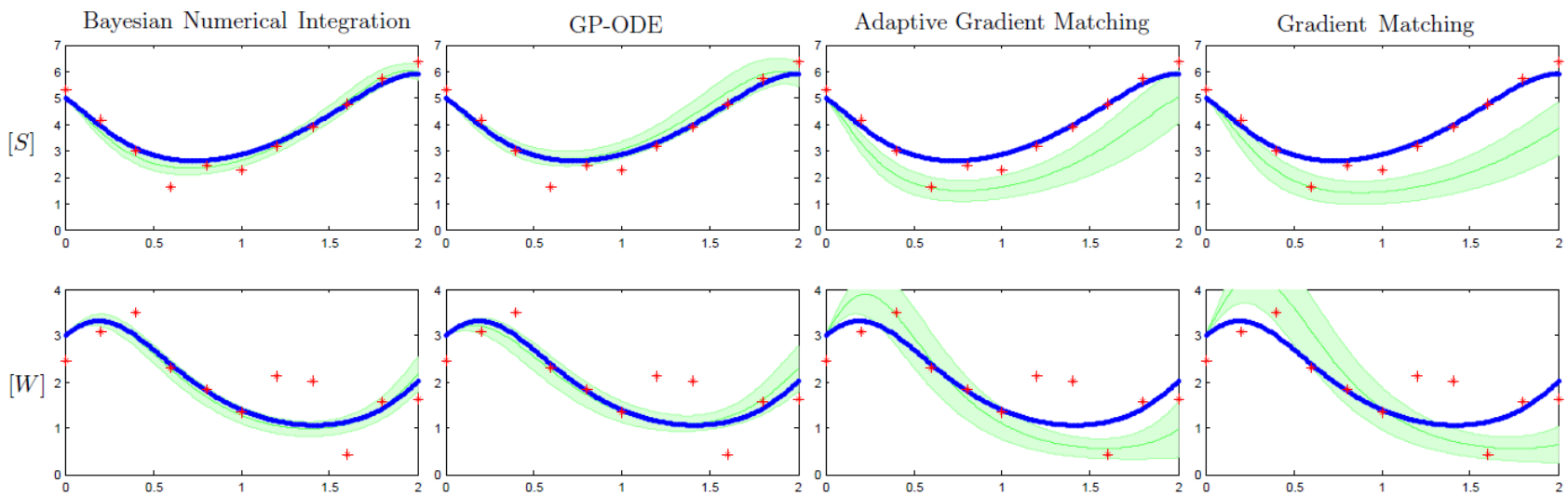


FIGURE 3.9: Bayesian Inference for Lotka-Volterra. The results for prey and predator are respectively shown in the first and second row. In all the plots, observations are red stars and the ground truth is the blue curve. Plotted in green are the reconstructions using the posterior samples of θ. To aid comparison between the approaches we numerically integrated the ODE starting from the true initial state $[5, 3]$ with each curve generated by numerical integration using a parameter sample θ. The green plots are the mean and one standard deviation of these resulting curves. First column : Bayesian Numerical Integration. This represents the solution that we wish to approximate. Second column : Our GP-ODE method. Third column : The Adaptive Gradient Matching method. Fourth column : The Gradient Matching method. All competing methods were run for approximately 300s of CPU time.

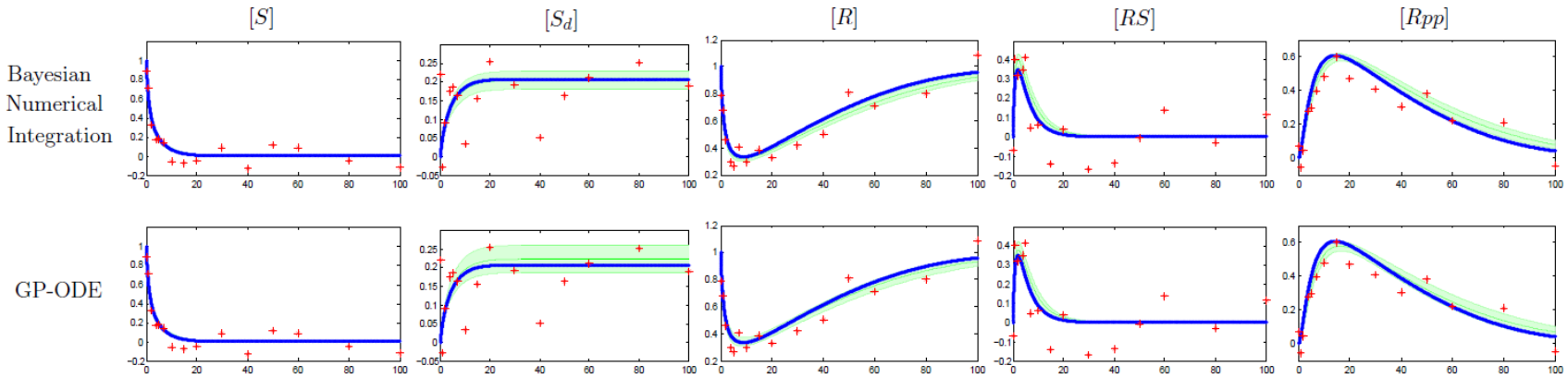


FIGURE 3.10: Bayesian Numerical Integration (top row) and GP-ODE results (bottom row) for Signal Transduction Cascade. The results for $[S]$, $[S_d]$, $[R]$, $[RS]$, $[Rpp]$ (from left to right). In all the plots, observations are red stars and the ground truth is the blue curve. The green curves show reconstructions using sampled parameters θ from each posterior. Each curve is obtained by numerical integration using the sampled parameter, starting from the same initial point $x_0 = [1, 0, 1, 0, 0]$.

Chapitre 4

Bayesian Estimation for Gaussian Processes

A Gaussian process (GP) is a powerful Bayesian nonparametric model (MacKay, 1998; Rasmussen and Williams, 2006). Due to the fact that it can be interpreted as a nonparametric prior, it allows us to make inference in a probabilistic Bayesian manner. However, the traditional inference mechanism with GPs is often difficult to interpret the real-world data properties, i.e., large size of data sets (Rasmussen and Williams, 2006; Snelson and Ghahramani, 2005) or input-dependent properties (Paciorek and Schervish, 2003; Plagemann, 2008). In this chapter, we took advantage of *Bayesian Estimation for Gaussian Processes* to design effective inference mechanisms for GPs to handle these challenges :

1. Firstly, we proposed a marginalized particle GP (Wang and Chaib-draa, 2012a) to reduce the high computational burden of GP regression in a recursive Bayesian framework. Meanwhile, our approach provided us with a novel online method for GP hyperparameter training.
2. Secondly, we proposed a KNN-based Kalman filter GP (Wang and Chaib-draa, 2013) to capture the input-dependent nonstationarity.
3. Finally, we proposed two sequential Monte Carlo based GPs (Wang and Chaib-draa, 2015a) to deal with the temporal-related nonstationarity and heteroscedasticity in time-varying applications.

4.1 Introduction

Over the past years, GPs have become popular in the machine learning community due to its Bayesian nonparametric framework (Neal, 1997; MacKay, 1998; Rasmussen and Williams, 2006). However, it is often difficult for a standard GP to model the challenging data properties of real-world applications (Rasmussen and Williams, 2006; Quinonero-Candela and Rasmussen, 2005; Kuss, 2006; Plagemann, 2008).

4.1.1 Large Data Size

Due to the fact that the computational burden of a standard GP implementation is mainly governed by $O(n^3)$ where n is the size of the data set, it is often intractable to use GPs for large data sets. In order to reduce the unsatisfactory computation, several GP extensions have been attempted with different sparse mechanisms (Tresp, 2000; Smola and Bartlett, 2000; Csató and Opper, 2002; Seeger et al., 2003; Quinonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2005; Rasmussen and Ghahramani, 2001; Urtasun and Darrell, 2008; Reece and Roberts, 2010). A detailed review of sparse approximate GPs can be found in Quinonero-Candela and Rasmussen (2005); Chalupka et al. (2013). In the following, we will briefly discuss several computationally efficient GPs which are used in this thesis.

One popular way for computation reduction is to learn a small inducing set offline from the whole training set to achieve sparsification. A well-known approach is called sparse pseudo-input Gaussian process (SPGP) (Snelson and Ghahramani, 2005) where the covariance of the SPGP is parameterized by the locations of several pseudo-input points that are learned by gradient-based optimization. However, its accuracy is often limited, especially when the size of the pseudo set is getting larger and/or the dimensionality of the input space is getting higher (Snelson and Ghahramani, 2005). Recently, a sparse spectrum Gaussian process (SSGP) (Lázaro-Gredilla et al., 2010) has been proposed with a stationary trigonometric Bayesian model which retains the computational efficiency of the SPGP while improving its accuracy. But similar to the SPGP, the SSGP may present overfitting since the optimization is implemented over a large number of model parameters. Additionally, it is worth mentioning that the SPGP and SSGP, to some degree, have the capability to deal with input-dependent properties (such as heteroscedasticity) because of pseudo-input or spectral-point learning.

Another popular way for computation reduction is to probabilistically divide the whole training set into several small subsets using a mixture of GP experts. In this case, the computation load is naturally reduced since the prediction is based on a number of small-sized GP experts. One well-known method is the infinite mixture of Gaussian process experts (iMGPE) (Rasmussen and Ghahramani, 2001) where an input-dependent Dirichlet process is designed as a gating network for GP expert selection. However, the hierarchical model structure is not easy to implement in practice (Paciorek and Schervish, 2003). Several practical variants were proposed by constructing a local mixture of GP experts for large-scale computer vision and robotics applications (Urtasun and Darrell, 2008; Nguyen-Tuong et al., 2008). Compared to iMGPE, the local approaches are suitable choices when a tradeoff between accuracy and efficiency is required. Additionally, iMGPE and the local variants can be used to deal with the input-dependent data properties due to the GP mixture representation.

The last but not least popular approach for computation reduction is to sequentially process the training data set in an online fashion to speed up the computation efficiency. A well-known

method is sparse online GP (SOGP) (Csató and Opper, 2002) where recursive Bayesian inference is coupled with an active set selection mechanism to balance the tradeoff between accuracy and efficiency. The most attractive point of SOGP is its online estimation for the latent function values of a data stream. But GP hyperparameters in SOGP have to be learned offline, and this may restrict the performance of SOGP for time-varying applications. Recently, a kernel recursive least-square tracker (KRLS) (Vaerenbergh et al., 2012) was proposed to capture the time-varying data properties by taking online hyperparameter learning into account. But its performance may still be limited as KRLS only learns a scale parameter which is assumed to be the same in the kernel function and noise variance. Finally, it is worth mentioning a recent Kalman filter Gaussian process (KFGP) (Reece and Roberts, 2010) which naturally reduces the computation by recursively correlating GP priors of different training subsets in an efficient Kalman filter framework. Furthermore, it provides us with a novel view to connect GP with Kalman filter (i.e., Using Kalman filter as Bayesian inference mechanism for GP), which is our key inspiration of *Bayesian Estimation for Gaussian Processes*. We will further discuss our motivation in Subsection 4.1.3.

4.1.2 Input-Dependent Properties

Many real-world applications in geophysics, biology and robotics often reflect strong input-dependent characteristics (Rasmussen and Ghahramani, 2001; Paciorek and Schervish, 2003; Kersting et al., 2007; Plagemann, 2008; Adams and Stegle, 2008; Lázaro-Gredilla and Titsias, 2011). It is often difficult to model these phenomena correctly with a standard GP.

One input-dependent property is input-dependent smoothness (nonstationarity) where the correlation between the latent function values $f(\mathbf{x})$ and $f(\mathbf{x}')$ does not only depend on $\mathbf{x} - \mathbf{x}'$ but it is also related to the locations of inputs $\mathbf{x}$ and $\mathbf{x}'$ (Rasmussen and Williams, 2006; Plagemann, 2008; Adams and Stegle, 2008). A popular method is to predefine a nonstationary covariance function for the GP. A review of several existing nonstationary covariance functions can be found in Rasmussen and Williams (2006). However, the performance of this method is often limited due to the fact that the predefined nonstationary covariance functions require either prior knowledge on the specific nonstationarity (Rasmussen and Williams, 2006; Adams and Stegle, 2008) or complicated inference mechanisms for model parameter learning (Paciorek and Schervish, 2003).

Another input-dependent property is input-dependent noise (heteroscedasticity) where the variance of the observation noise σ_y^2 depends on the location of the input $\mathbf{x}$. A popular treatment is to add a GP prior on the input-dependent noise (Goldberg et al., 1998). However, it makes Bayesian inference intractable. Gibbs sampling has been used with a heavy computation load (Goldberg et al., 1998). To apply this heteroscedastic GP for real-life applications, a most likely heteroscedastic Gaussian process (MLHGP) was proposed in a fast (hard) EM learning framework (Kersting et al., 2007). But MLHGP is not guaranteed to converge and may

instead oscillate (Lázaro-Gredilla and Titsias, 2011). A more recent work is a variational heteroscedastic Gaussian process (VHGP) (Lázaro-Gredilla and Titsias, 2011), which is not prone to overfitting. However, the computational cost is roughly twice as much as a standard GP (Lázaro-Gredilla and Titsias, 2011). Additionally, there exists several warped GPs in which input-dependent properties are achieved by warping GP with different nonlinear functions (Snelson et al., 2003; Adams and Stegle, 2008; Lázaro-Gredilla, 2012).

4.1.3 Motivations

From the introduction above, we can see that the performance of a standard GP is often deteriorated when the data sets reflect several challenging data properties. Therefore, we were motivated to address these real-world difficulties within an efficient and accurate Bayesian estimation framework. The recent Kalman filter Gaussian process (KFGP) (Reece and Roberts, 2010), which was mainly introduced for computation reduction in GPs, opens a door for us. In KFGP, the classical Kalman filter is used as an online Bayesian inference mechanism for a GP so that the latent function values can be efficiently estimated. Inspired by this novel connection between a GP and a Kalman filter, we proposed *Bayesian Estimation for Gaussian Processes* in which we will design several sequential Bayesian estimation mechanisms for GPs to interpret challenging data properties in a number of efficient and accurate Bayesian filtering frameworks.

4.2 A Marginalized Particle Gaussian Process Regression

As mentioned in Chapter 2, a Gaussian process is an elegant nonparametric Bayesian model for nonlinear regression. However, the computational complexity of $O(n^3)$ for training a standard GP limits its applicability in practice, when the number of the training points n is larger than a few thousands (Rasmussen and Williams, 2006).

In Wang and Chaib-draa (2012a), we proposed a novel marginalized particle Gaussian process (MPGP) regression approach in which we took advantage of a fast and accurate marginalized particle filtering framework to learn GP hyper-parameters and hidden function values in an online fashion.

4.2.1 Data Collection

In practice, the whole training set is often constructed by gathering several small subsets, one at a time. For the t-th collection, the training subset $(X_t, \mathbf{y}_t)$ consists of n_t input-output pairs, i.e., $\{(\mathbf{x}_t^1, y_t^1), \cdots (\mathbf{x}_t^{n_t}, y_t^{n_t})\}$ where each scalar output $y_t^i \in R$ is generated from a nonlinear function $f(\mathbf{x}_t^i)$ of an input vector $\mathbf{x}_t^i \in R^{D_x}$ with an additive Gaussian noise $\mathcal{N}(0, a_0^2)$. All the pairs are separately organized as an input matrix $X_t = [\mathbf{x}_t^1, \cdots \mathbf{x}_t^{n_t}]^T$ and an output vector $\mathbf{y}_t = [y_t^1, \cdots y_t^{n_t}]^T$. For simplicity, the whole training data set with T collections is

symbolized as $(X_{1:T}, \mathbf{y}_{1:T})$. Our goal is to estimate the function values of $f(\mathbf{x})$ at m test inputs $X_\star = [\mathbf{x}_\star^1, \cdots \mathbf{x}_\star^m]^T$ given $(X_{1:T}, \mathbf{y}_{1:T})$.

4.2.2 Reorganization of Gaussian Process Regression

In this contribution, we follow a popular zero-mean GP setting (Rasmussen and Williams, 2006), $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$. Furthermore, we choose a compounded covariance function $k_{SENN}(\mathbf{x}, \mathbf{x}')$ to deal with non-stationary phenomena in the real world,

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

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

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

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

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

The augmented input in $k_{NN}(\mathbf{x}, \mathbf{x}')$ is

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}.$$

For simplicity, we collect the noise variance and all the hyper-parameters into a vector

$$\theta = [a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4]^T. \tag{4.1}$$

According to the sequential procedure of data collection, we reorganize the target posterior,

$$p(f(X_\star), \theta | X_{1:T}, \mathbf{y}_{1:T}, X_\star) = p(\theta | X_{1:T}, \mathbf{y}_{1:T}, X_\star) p(f(X_\star) | X_{1:T}, \mathbf{y}_{1:T}, X_\star, \theta). \tag{4.2}$$

The standard GP regression can be used to estimate this posterior in a two-step off-line manner. However, the standard batch implementation of a GP suffers from the high computation complexity of $O(n^3)$ where $n = \sum_{t=1}^T n_t$. This makes GP regression often infeasible for large data sets.

Inspired by the fact that the training set is constructed sequentially, we propose to take advantage of this sequential manner to derive an sequential inference framework for GP regression so that the latent $f(X_\star)$ and θ can be estimated efficiently and accurately.

4.2.3 Gaussian Process Based State Space Model

In order to make sequential Bayesian estimation for GP, we first explore a GP based state space model (SSM) in the following. Then according to the characteristics of our SSM, we propose to design a marginalized particle filtering framework to speed up the efficiency while preserving the accuracy.

A standard SSM consists of a prediction model (also called as transition model or motion model) and an observation model. We first discuss about how to construct the prediction model which reflects the Markovian evolution of the latent states. In our context of GPs, the latent states are actually the unknown hyper-parameter vector θ (in Equation 4.1) and the latent function values.

For the unknown hyper-parameter vector θ, a popular way in Bayesian filtering techniques is to introduce an artificial dynamics of θ (Liu and West, 2001; Li et al., 2004; Kantas et al., 2009),

$$\theta_t = b\theta_{t-1} + (1-b)\bar{\theta}_{t-1} + s_{t-1}, \tag{4.3}$$

where $b = (3\delta - 1)/(2\delta)$, δ is a discount factor which is typically around 0.95-0.99, $\bar{\theta}_{t-1}$ is the Monte Carlo mean of θ at $t-1$, and $s_{t-1} \sim \mathcal{N}(0, r^2\Sigma_{t-1})$, $r^2 = 1 - b^2$, Σ_{t-1} is the Monte Carlo variance matrix of θ at $t-1$.

For the latent function values, we explore the correlation between the $(t-1)_{th}$ and t_{th} data subsets by using a GP. For simplicity, we denote $X_t^c = X_t \cup X_\star$ and $f_t^c = f(X_t^c)$. If $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$, the prior $p(f_t^c, f_{t-1}^c | X_{t-1}^c, X_t^c, \theta_t)$ is jointly Gaussian

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

Then according to the conditional property of the Gaussian distribution, we can obtain

$$p(f_t^c | f_{t-1}^c, X_{t-1}^c, X_t^c, \theta_t) = \mathcal{N}(G(\theta_t) f_{t-1}^c, Q(\theta_t)), \tag{4.4}$$

where

$$G(\theta_t) = K_{\theta_t}(X_t^c, X_{t-1}^c) K_{\theta_t}^{-1}(X_{t-1}^c, X_{t-1}^c), \tag{4.5}$$

$$Q(\theta_t) = K_{\theta_t}(X_t^c, X_t^c) - K_{\theta_t}(X_t^c, X_{t-1}^c) K_{\theta_t}^{-1}(X_{t-1}^c, X_{t-1}^c) K_{\theta_t}(X_t^c, X_{t-1}^c)^T. \tag{4.6}$$

In fact, the conditional density (in Equation 4.4) represents the prediction model for the latent function values, which is a linear state transition with an additive Gaussian noise $v_t^f \sim \mathcal{N}(\mathbf{0}, Q(\theta_t))$,

$$f_t^c = G(\theta_t) f_{t-1}^c + v_t^f. \tag{4.7}$$

After constructing the prediction model for latent function values and hyper-parameters, we now focus on the observation model which can be straightforwardly obtained from the t-th

data collection,

$$\mathbf{y}_t = H_t f_t^c + v_t^y, \tag{4.8}$$

where $H_t = [I_{n_t} \quad \mathbf{0}]$ 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 Subsection 4.2.1 where the covariance of the noise $R(\theta_t)$ is $a_{0,t}^2 I$. Note that a_0 is a static, unknown hyper-parameter. We use the symbol $a_{0,t}$ just because of the consistency with the artificial evolution of θ.

Now our SSM is fully specified by the prediction model (Equation 4.3 and 4.7) and the observation model (Equation 4.8). Based on this SSM, we propose a sequential inference framework in the following to learn the latent function values and unknown hyper-parameters in an online fashion.

4.2.4 Bayesian Inference by Marginalized Particle Filter

In contrast to the standard GP regression with a two-step off-line inference, we propose an online filtering framework to simultaneously estimate hidden function values and learn the hyper-parameters. According to our SSM, the inference problem refers to computing the posterior distribution,

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

which is analytically intractable. One of the most popular approximation techniques is Monte Carlo sampling (Gilks et al., 1996; Doucet et al., 2001). Due to the fact that our SSM is based on a sequential data collection procedure, we hence investigate particle filtering approaches that are based on sequential Monte Carlo (SMC) sampling (Doucet et al., 2000b; Cappé et al., 2007).

Furthermore, for our SSM the traditional sampling importance resampling (SIR) particle filter introduces an unnecessary computational burden since Equation 4.7 in our SSM is a linear structure given θ_t. This inspires us to apply a more efficient marginalized particle filter (also called Rao-Blackwellised particle filter) (Li et al., 2004; Doucet et al., 2000a; de Freitas, 2002; Schön et al., 2005) in which the latent states in the conditional linear structure can be efficiently estimated by a Kalman filter.

Specifically, the posterior can be factorized by using Bayes rule,

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

As the first term $p(\theta_{1:t} | X_{1:t}, X_\star, \mathbf{y}_{1:t})$ is analytically intractable, we approximate it by a number of particles. After the particle approximation of $\theta_{1:t}$ is obtained, the second term $p(f_t^c | \theta_{1:t}, X_{1:t}, X_\star, \mathbf{y}_{1:t})$ can be efficiently computed by Kalman filter since f_t^c is the hidden state in the linear substructure (Equation 4.7) of SSM.

Firstly, we focus on $p(\theta_{1:t}|X_{1:t}, X_\star, \mathbf{y}_{1:t})$. Since it is not analytically tractable, we factorize it in the following recursive form so that it can be approximated using a sequential importance sampling mechanism,

$$p(\theta_{1:t}|X_{1:t}, X_\star, \mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)p(\theta_t|\theta_{t-1})p(\theta_{1:t-1}|X_{1:t-1}, X_\star, \mathbf{y}_{1:t-1}).$$

At each iteration of sequential importance sampling, the particles for the hyper-parameter vector are drawn from the proposal distribution $p(\theta_t|\theta_{t-1})$ (easily obtained from (Equation 4.3)). 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 could be solved analytically,

$$\begin{aligned}
p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star) &= \int p(\mathbf{y}_t, f_t^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)df_t^c \\
&= \int p(\mathbf{y}_t|f_t^c, \theta_t, X_t, X_\star)p(f_t^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)df_t^c \\
&= \int \mathcal{N}(H_t f_t^c, R(\theta_t))\mathcal{N}(f_{t|t-1}^c, P_{t|t-1}^c)df_t^c \\
&= \mathcal{N}(H_t f_{t|t-1}^c, H_t P_{t|t-1}^c H_t^T + R(\theta_t)), \qquad (4.9)
\end{aligned}$$

where $p(\mathbf{y}_t|f_t^c, \theta_t, X_t, X_\star)$ is Gaussian distributed $\mathcal{N}(H_t f_t^c, R(\theta_t))$ due to our Gaussian observation model (Equation 4.8). Moreover, $p(f_t^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)$ is also Gaussian distributed $\mathcal{N}(f_{t|t-1}^c, P_{t|t-1}^c)$ with the predictive mean $f_{t|t-1}^c$ and covariance $P_{t|t-1}^c$ as it is the prediction step of Kalman filter for f_t^c.

Secondly, we explain how to compute $p(f_t^c|\theta_{1:t}, X_{1:t}, X_\star, \mathbf{y}_{1:t})$. As mentioned before, this posterior can be recursively obtained using a Kalman filter since Equations 4.7 and 4.8 are linear and Gaussian given $\theta_{1:t}$. Specifically, we use Bayes rule to factorize this posterior as follows :

$$p(f_t^c|\theta_{1:t}, X_{1:t}, X_\star, \mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|f_t^c, \theta_t, X_t, X_\star)p(f_t^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)}. \qquad (4.10)$$

In the prediction step, the goal is to compute $p(f_t^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)$,

$$\begin{aligned}
p(f_t^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star) &= \int p(f_t^c, f_{t-1}^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)df_{t-1}^c \\
&= \int p(f_t^c|f_{t-1}^c, \theta_t, X_{t-1:t}, X_\star)p(f_{t-1}^c|\mathbf{y}_{1:t-1}, \theta_{1:t-1}, X_{1:t-1}, X_\star)df_{t-1}^c \\
&= \int \mathcal{N}(G(\theta_t)f_{t-1}^c, Q(\theta_t))\mathcal{N}(f_{t-1|t-1}^c, P_{t-1|t-1}^c)df_{t-1}^c \\
&= \mathcal{N}(G(\theta_t)f_{t-1|t-1}^c, \quad G(\theta_t)P_{t-1|t-1}^c G(\theta_t)^T + Q(\theta_t)), \qquad (4.11)
\end{aligned}$$

where $p(f_t^c|f_{t-1}^c, \theta_t, X_{t-1:t}, X_\star)$ is directly from Equation 4.4, and $p(f_{t-1}^c|\mathbf{y}_{1:t-1}, \theta_{1:t-1}, X_{1:t-1}, X_\star)$ is the posterior estimation for f_{t-1}^c, i.e., $\mathcal{N}(f_{t-1|t-1}^c, P_{t-1|t-1}^c)$. Since $p(f_t^c|\mathbf{y}_{1:t-1}, \theta_{1:t}, X_{1:t}, X_\star)$ is also expressed as $\mathcal{N}(f_{t|t-1}^c, P_{t|t-1}^c)$, then the prediction step of Kalman filter is summarized as :

$$\begin{aligned}
f_{t|t-1}^c &= G(\theta_t)f_{t-1|t-1}^c, \qquad (4.12) \\
P_{t|t-1}^c &= G(\theta_t)P_{t-1|t-1}^c G(\theta_t)^T + Q(\theta_t). \qquad (4.13)
\end{aligned}$$

In the update step, the current observation density $p(\mathbf{y}_t|f_t^c, \theta_t, X_t, X_\star) = \mathcal{N}(H_t f_t^c, R(\theta_t))$ is used to correct the prediction. Putting Equations 4.9 and 4.11 into Equation 4.10, the posterior $p(f_t^c|\theta_{1:t}, X_{1:t}, X_\star, \mathbf{y}_{1:t})$ is actually Gaussian distributed $\mathcal{N}(f_{t|t}^c, P_{t|t}^c)$ with Kalman Gain Γ_t :

$$\Gamma_t = P_{t|t-1}^c H_t^T (H_t P_{t|t-1}^c H_t^T + R(\theta_t))^{-1}, \tag{4.14}$$

$$f_{t|t}^c = f_{t|t-1}^c + \Gamma_t(\mathbf{y}_t - H_t f_{t|t-1}^c), \tag{4.15}$$

$$P_{t|t}^c = P_{t|t-1}^c - \Gamma_t H_t P_{t|t-1}^c. \tag{4.16}$$

The whole algorithm is summarized in Algorithm 6. At each iteration, we first draw N_p particles for the hyper-parameter vector (Algorithm 6, Line 3). Then we use these particles to calculate the computation quantities of our SSM (Algorithm 6, Line 4) so that we can subsequently perform Kalman filter for latent function values (Algorithm 6, Line 5-6). Next we compute the weights of the particles and estimate the latent states of our SSM (Algorithm 6, Line 7-10). Finally, according to the normalized weights we resample the particles for the next step (Algorithm 6, Line 11). 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, $S = \max(n_t), \quad (t = 1, 2, ...T)$ is the max size among all the data subsets. Compared to a standard GP regression, our MPGP naturally cut down the computational load by sequentially processing a number of small data subsets. Moreover, note that $f(X_\star)$ is estimated as a Gaussian mixture at each iteration (since each hyper-parameter particle accompanies with a Kalman filter for $f(X_\star)$), and the recursive filtering framework allows us to propagate the previous estimation to improve the current accuracy. Therefore, our MPGP could accelerate the computational speed while preserving the accuracy.

Finally, it is worth mentioning that Kalman filter GP (KFGP) (Reece and Roberts, 2010) can be treated as a special case of our MPGP since KFGP firstly trains the hyper-parameter vector off-line, and then estimates $p(f_t^c|\theta_{1:t}, X_{1:t}, X_\star, \mathbf{y}_{1:t})$ by Kalman filter. However, the off-line learning procedure in KFGP will either take a long time using a large extra training data or fall into an unsatisfactory local optimum using a small extra training data. On the contrary, in our MPGP the local optimum solution can be used as the initial setting of hyper-parameters, and then the underlying θ is further learned online by the marginalized particle filter to improve the performance.

4.2.5 Experiments

In the following, we compare our MPGP to several state-of-the-art fast GP approaches by evaluating the accuracy and the efficiency of all the approaches on a number of synthetic and real-world data sets.

Algorithm 6 Marginalized Particle Gaussian Process (MPGP) Regression.

for $t = 1$ to T **do**
  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 $\tilde{\theta}_{t-1}^i$ with (4.3)
    Using θ_t^i to specify $k(\mathbf{x}, \mathbf{x}')$ in GP to construct $G(\theta_t^i)$, $Q(\theta_t^i)$, $R(\theta_t^i)$ in (4.5-4.6, 4.8)
    Kalman Predict : Using $\tilde{f}_{t-1|t-1}^{c,i}$, $\tilde{P}_{t-1|t-1}^{c,i}$ into (4.12-4.13) 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 (4.14-4.16) 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 (4.9) to compute the importance weight $\bar{w}_t^i$
  end for
  Normalizing the weight : $w_t^i = \bar{w}_t^i / (\sum_{i=1}^{N_p} \bar{w}_t^i) \quad (i = 1, ...N_p)$
  Hyper-parameter and hidden function value estimation by summarizing the Gaussian mixture $\sum_{i=1}^{N_p} w_t^i \mathcal{N}(f_{t|t}^{c,i}, P_{t|t}^{c,i})$ with $\mathcal{N}(\hat{f}_{t|t}^c, \hat{P}_{t|t}^c)$

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

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

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

  where $H_t^\star = [\mathbf{0} \;\; I_m]$ is an index matrix to get the function value estimation at $X_\star$
  Resampling : For $i = 1, ...N_p$, resample $\theta_t^i, f_{t|t}^{c,i}, P_{t|t}^{c,i}$ with respect to the importance weight w_t^i to obtain $\tilde{\theta}_t^i, \tilde{f}_{t|t}^{c,i}, \tilde{P}_{t|t}^{c,i}$ for the next step
end for

Two Synthetic Data Sets

We first evaluate our proposed MPGP on two synthetic one-dimensional data sets. One is a function with a sharp peak which is spatially inhomogeneous smooth (DiMatteo et al., 2001)

$$f_1(x) = \sin(x) + 2\exp(-30x^2).$$

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], 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) :

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

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], 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.

In the **first** experiment, we evaluate the inference performance of our MPGP in comparison with the closely-related KFGP in 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 MPGP with k_{SE} and MPGP with $k_{SE}+k_{NN}$. The number of particles in MPGP is set to 10.

First, we plot the estimation surface of our MPGP. It is shown in Figures 4.1 and 4.2 that the estimation performance for both KFGP and MPGP is getting better and tends to converge over time since the previous estimation would be incorporated into the current estimation in the recursive Bayesian filtering. However, our MPGP reduces the uncertainty, which provides us with a more compact confidence interval than KFGP, due to the online hyper-parameter learning (Figure 4.3 and 4.4) [1].

Second, we evaluate the accuracy of our MPGP over time. 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). From Figures 4.5 and 4.6, we can see that the accuracy of MPGP for both f_1 and f_2 are better than KFGP, especially in the MNLP criterion which penalizes both the uncertainty and the inconsistency. The main reason is that KFGP uses the off-line learned hyper-parameters all the time. On the contrary, our MPGP initializes the hyper-parameters using the ones of KFGP, then learns the hyper-parameters online. Hence the MNLP of MPGP is much lower than KFGP. Additionally, if we only focus on our MPGP, then we can find that our SENN-MPGP is better than our SE-MPGP since 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 number of particles, we run the SE-MPGP and SENN-MPGP 5 times and compute the average NMSE and MNLP. From Figures 4.7 and 4.8, we find that, when we increase the number of particles, the NMSE and MNLP of SE-MPGP and SENN-MPGP decrease while the running time increases over time. The reason is that the estimation accuracy and computational load of particle filters tend to increase when the number of particles increases. Furthermore, the average performance of SENN-MPGP is better than SE-MPGP since it captures the spatial non-stationarity, but SENN-MPGP needs more running time than SE-MPGP since the dimension of the hyper-parameter vector in SENN-MPGP is larger than SE-MPGP.

In the **third** experiment, we compare our MPGP with other fast GP approaches. The state-of-art sparse GP methods we chose are sparse pseudo-input Gaussian process (SPGP) (Snelson and Ghahramani, 2005) and sparse spectrum Gaussian process (SSGP) (Lázaro-Gredilla et al.,

1. Since $\theta = [a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4]^T > 0$, in the experiment we construction the artificial transition of $\log(\theta)$ based on(4.3).

Table 4.1: Benchmarks Comparison for Synthetic Datasets. The NMSEi, MNLPi, RTimei represent the NMSE, MNLP and running time for the function $f_i \quad (i = 1, 2)$

Method	NMSE1	MNLP1	RTime1	NMSE2	MNLP2	RTime2
5-SPGP	0.2243	0.5409	28.6418s	0.5445	1.5950	30.3578s
10-SSGP	0.0887	0.1606	18.8605s	0.1144	1.1208	10.2025s
5-SE-MPGP	0.0880	1.6318	12.5737s	0.1687	1.3524	12.4801s
5-SENN-MPGP	0.0881	0.1820	18.7513s	0.1289	1.1782	11.5909s

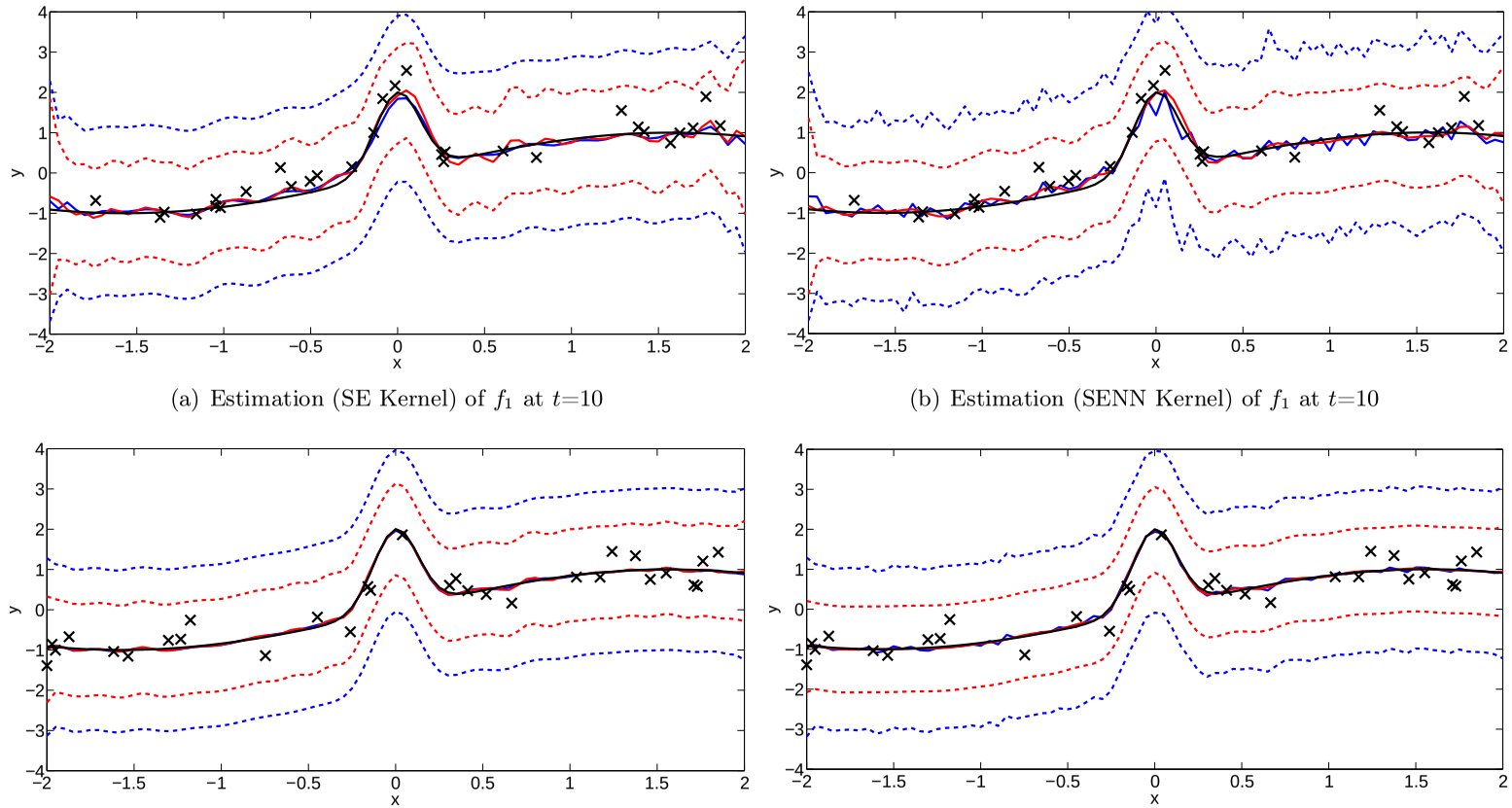
2010). To take the tradeoff between accuracy and computation efficiency into account, we implemented SPGP with 5 pseudo inputs (5-SPGP), SSGP with 10 basis functions (10-SSGP), SE-MPGP with 5 particles (5-SE-MPGP), SENN-MPGP with 5 particles (5-SENN-MPGP). Moreover, due to the fact that the chosen training 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 whether the good estimation of our MPGP heavily depends on the order of training data collection. Hence we randomly permute the order of training subsets (we used before) in this experiment. In Table 4.1, our 5-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 5-SENN-MPGP with a non-stationary kernel to show that our MPGP is competitive with SSGP, and much more accurate and computationally efficient than SPGP.

Global Surface Temperature Data Set

We here analyze the Global Surface Temperature Dataset (January 2011) [2]. We first gather the training data 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 : the first one is a grid test input set (Longitude : -180 :40 :180, Latitude : -90 :20 :90) that is used to show the estimated surface of the temperature. The second test set (100 points) is randomly selected from the data website after we obtain all the training data collections.

In the **first** experiment, we show the predicted surface temperature at the grid test inputs. For this data set, we set the number of particles in our SE-MPGP and SENN-MPGP as 20. From Figure 4.9, we can see that KFGP methods stuck in the local optimum. SE-KFGP seems to underfit since it does not model the cold region around the location (100, 50) and SENN-KFGP seems to overfit since it unexpectedly models the cold region around (-100, -50). On the contrary, SE-MPGP and SENN-MPGP suitably fit the data set by online hyper-parameter learning (Figure 4.10).

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



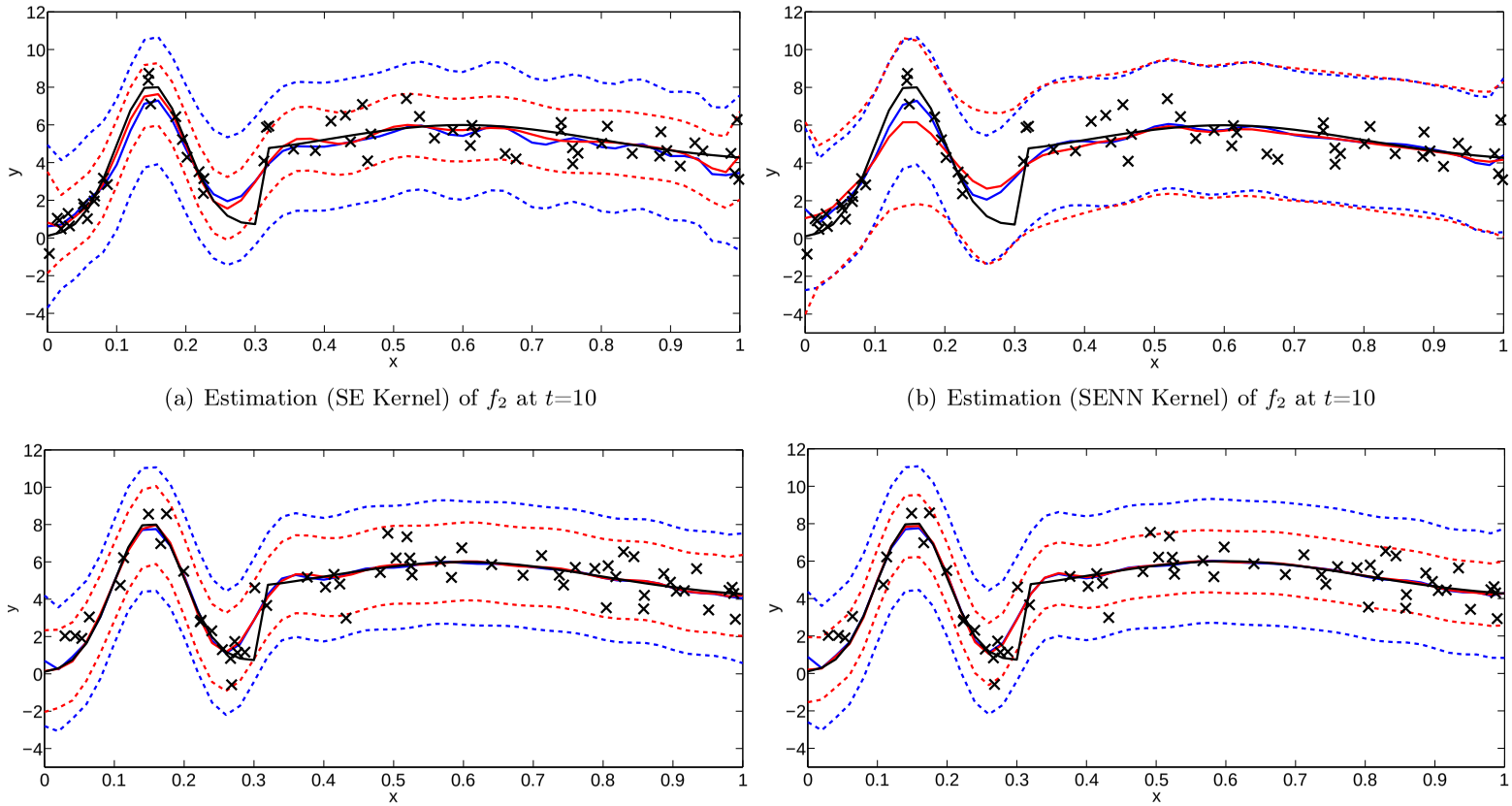
(a) Estimation (SE Kernel) of f_1 at t=10

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

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

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

FIGURE 4.1: Estimation result comparison for f_1. Figure 4.1(a)-4.1(b) show the estimation for f_1 at $t = 10$ by SE-KFGP (blue line with blue dashed interval in Figure 4.1(a)), SE-MPGP (red line with red dashed interval in Figure 4.1(a)), SENN-KFGP (blue line with blue dashed interval in Figure 4.1(b)), SENN-MPGP (red line with red dashed interval in Figure 4.1(b)). Figure 4.1(c)-4.1(d) show the estimation for f_1 at $t = 100$ by SE-KFGP (blue line with blue dashed interval in Figure 4.1(c)), SE-MPGP (red line with red dashed interval in Figure 4.1(c)), SENN-KFGP (blue line with blue dashed interval in Figure 4.1(d)), SENN-MPGP (red line with red dashed interval in Figure 4.1(d)). The black crosses in Figure 4.1(a)-4.1(b) and 4.1(c)-4.1(d) are respectively the training outputs at $t = 10$ and $t = 100$. In all the plots, the black line is the true $f_1(X_\star)$.



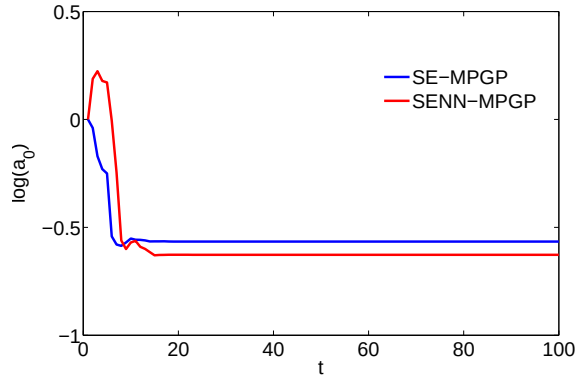
(a) Estimation (SE Kernel) of f_2 at t=10

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

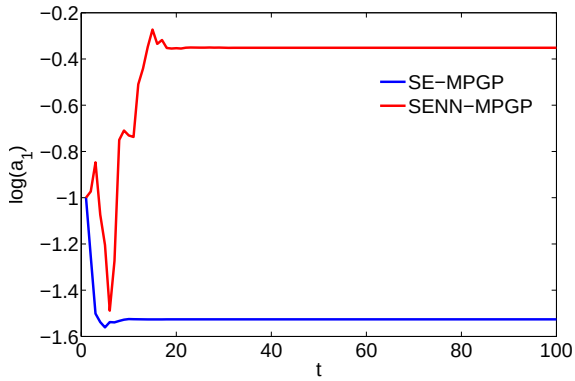
(c) Estimation (SE Kernel) of f_2 at t=50

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

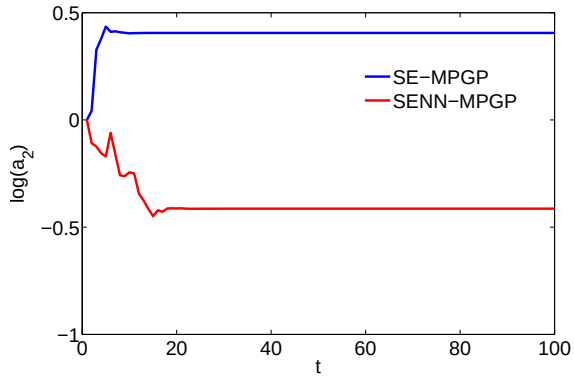
FIGURE 4.2: Estimation result comparison for f_2. Figure 4.2(a)-4.2(b) show the estimation for f_2 at $t = 10$ by SE-KFGP (blue line with blue dashed interval in Figure 4.2(a)), SE-MPGP (red line with red dashed interval in Figure 4.2(a)), SENN-KFGP (blue line with blue dashed interval in Figure 4.2(b)), SENN-MPGP (red line with red dashed interval in Figure 4.2(b)). Figure 4.2(c)-4.2(d) show the estimation for f_2 at $t = 50$ by SE-KFGP (blue line with blue dashed interval in Figure 4.2(c)), SE-MPGP (red line with red dashed interval in Figure 4.2(c)), SENN-KFGP (blue line with blue dashed interval in Figure 4.2(d)), SENN-MPGP (red line with red dashed interval in Figure 4.2(d)). The black crosses in Figure 4.2(a)-4.2(b) and 4.2(c)-4.2(d) are respectively the training outputs at $t = 10$ and $t = 50$. In all the plots, the black line is the true $f_2(X_\star)$.



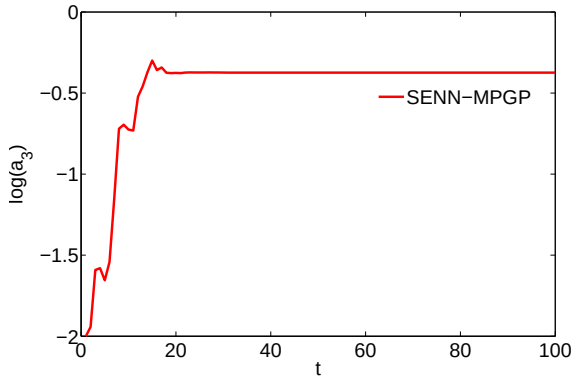
(a) Estimation of $\log(a_0)$ over time



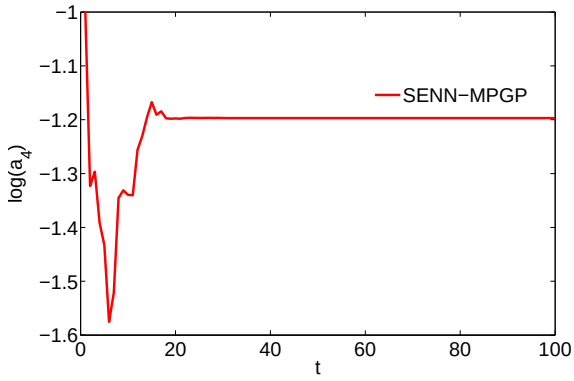
(b) Estimation of $\log(a_1)$ over time



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

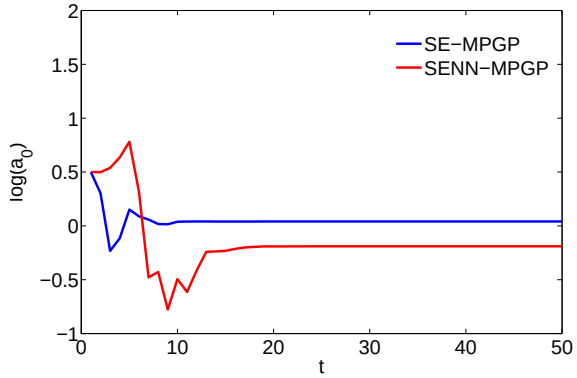


(d) Estimation of $\log(a_3)$ over time

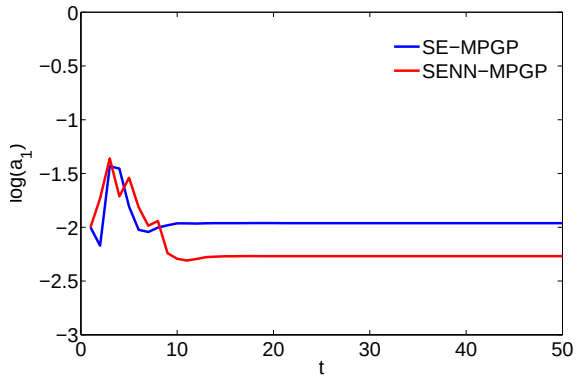


(e) Estimation of $\log(a_4)$ over time

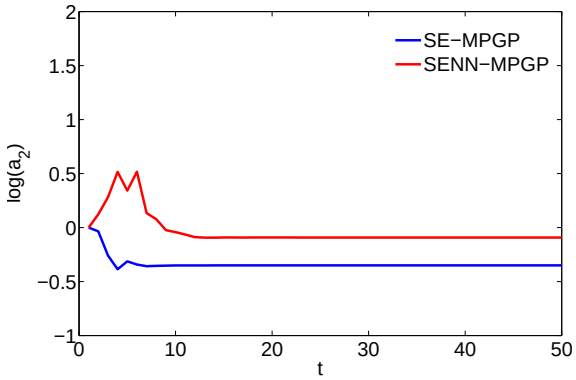
FIGURE 4.3: Online hyper-parameter learning for f_1.



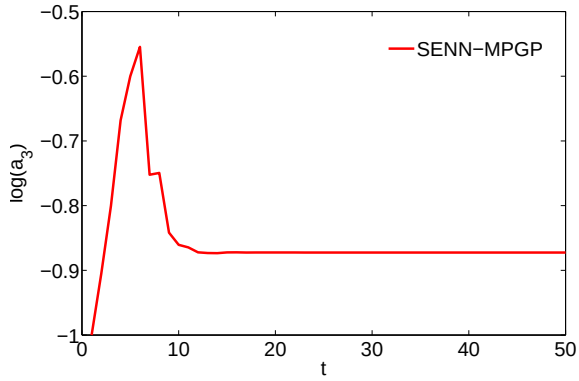
(a) Estimation of $\log(a_0)$ over time



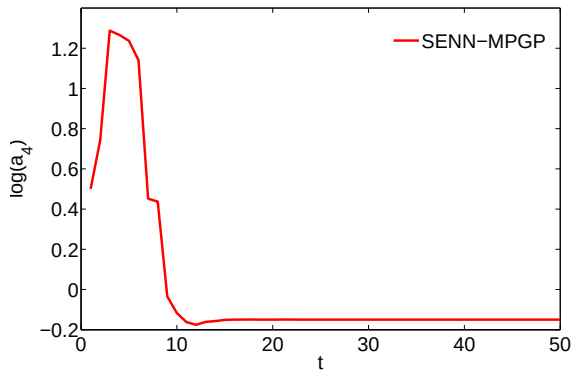
(b) Estimation of $\log(a_1)$ over time



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

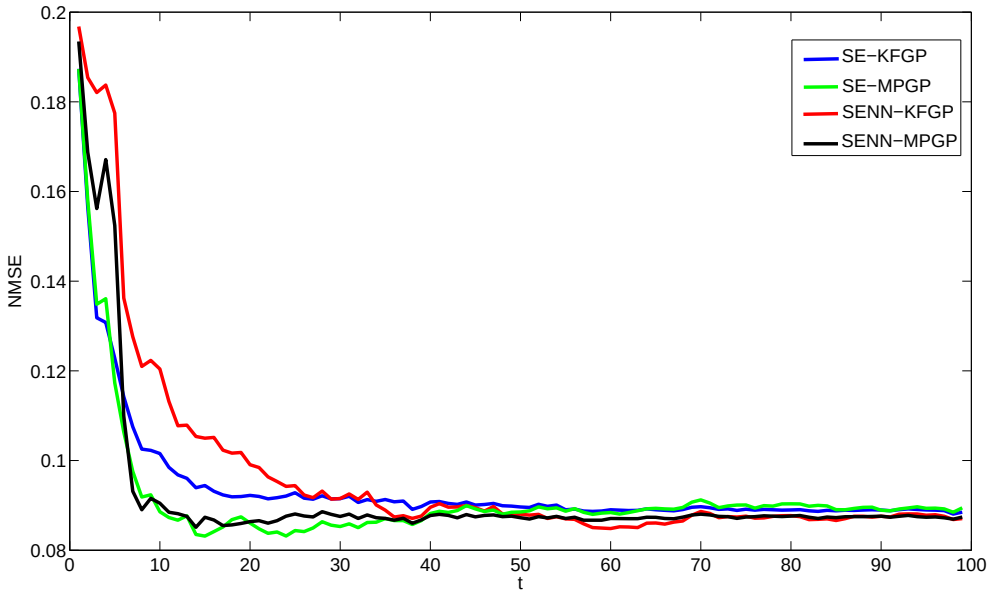


(d) Estimation of $\log(a_3)$ over time

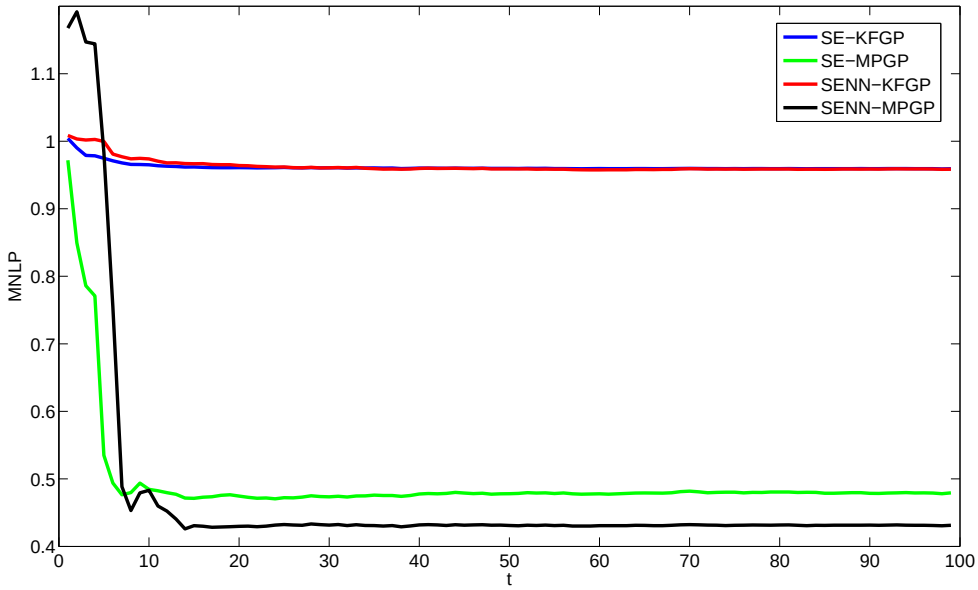


(e) Estimation of $\log(a_4)$ over time

FIGURE 4.4: Online hyper-parameter learning for f_2.

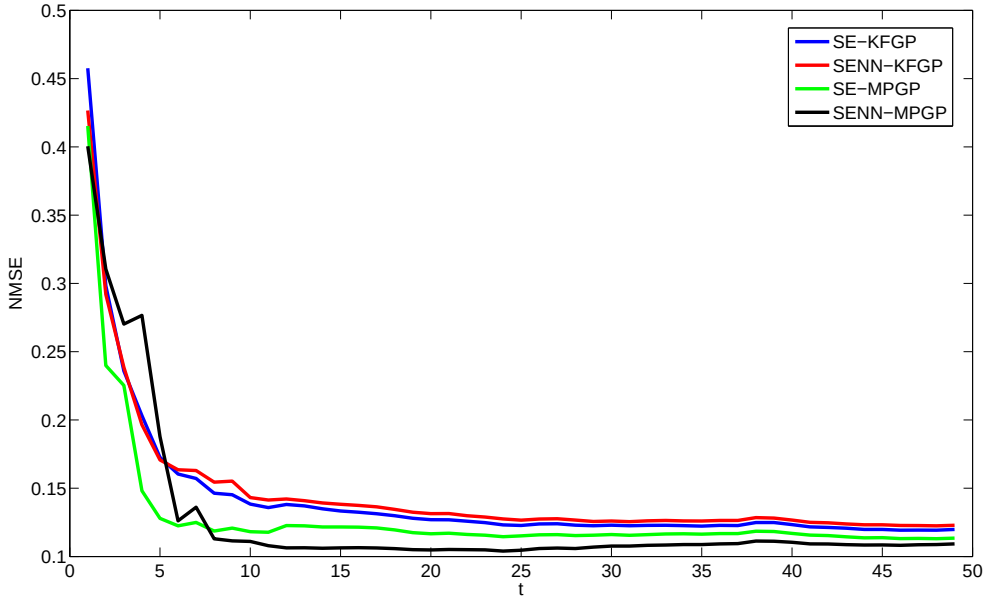


(a) NMSE for f_1 over time

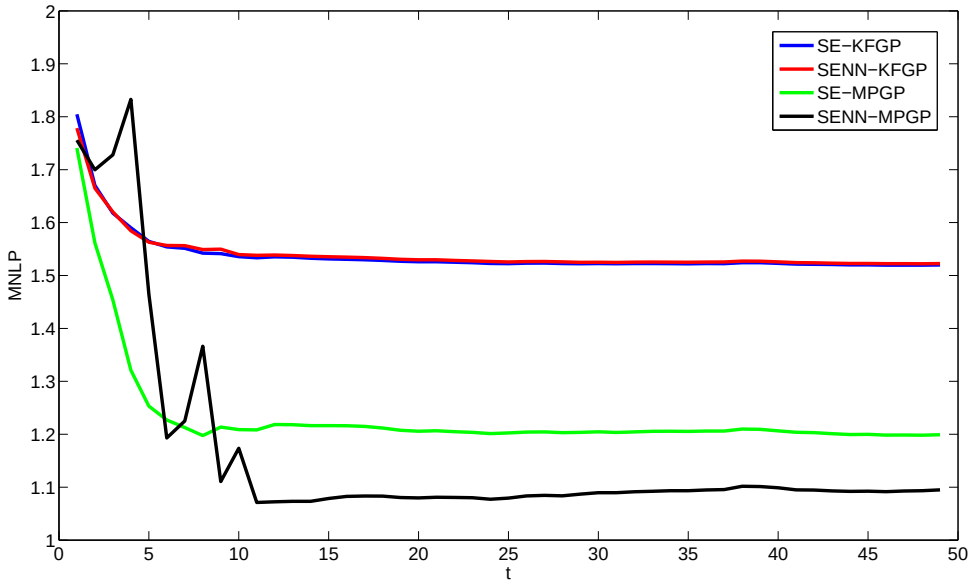


(b) MNLP for f_1 over time

FIGURE 4.5: Accuracy evaluation for f_1 over time.

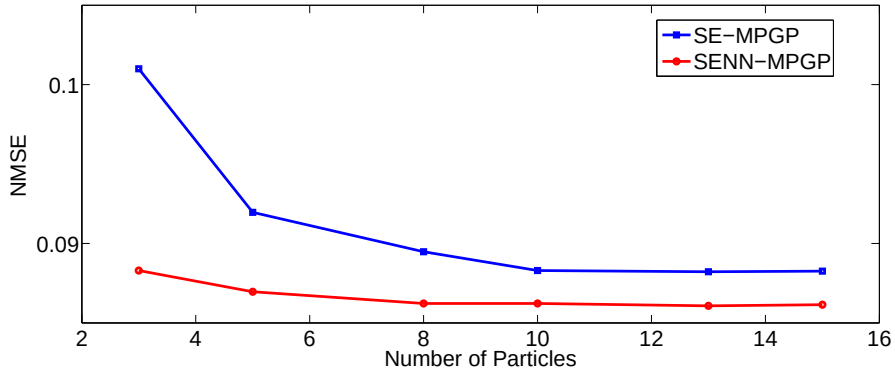


(a) NMSE for f_2 over time

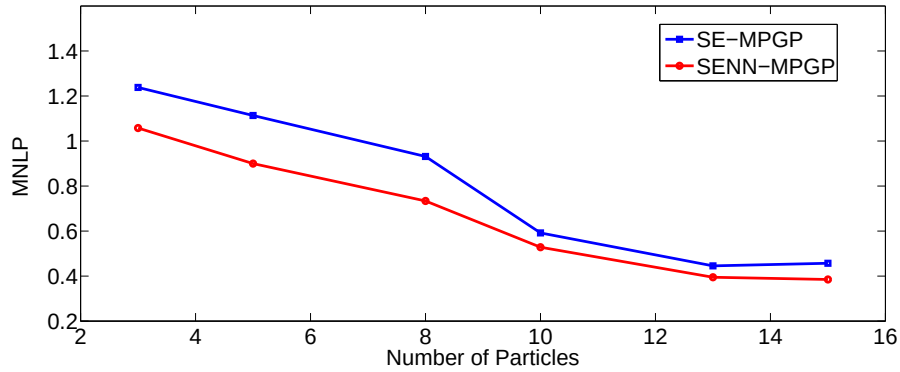


(b) MNLP for f_2 over time

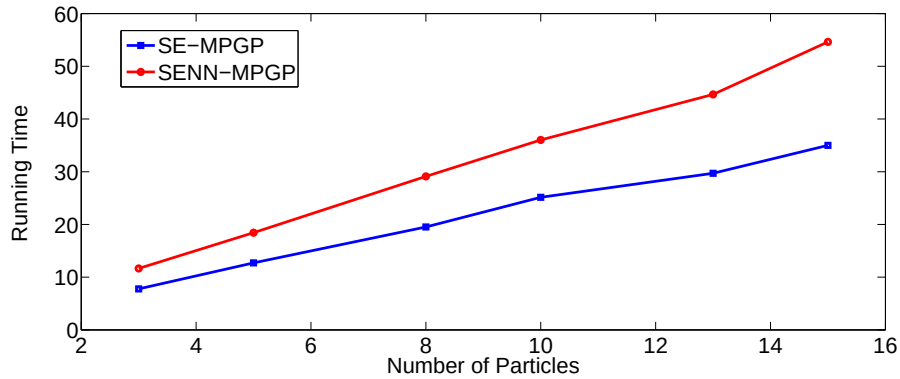
FIGURE 4.6: Accuracy evaluation for f_2 over time.



(a) NMSE

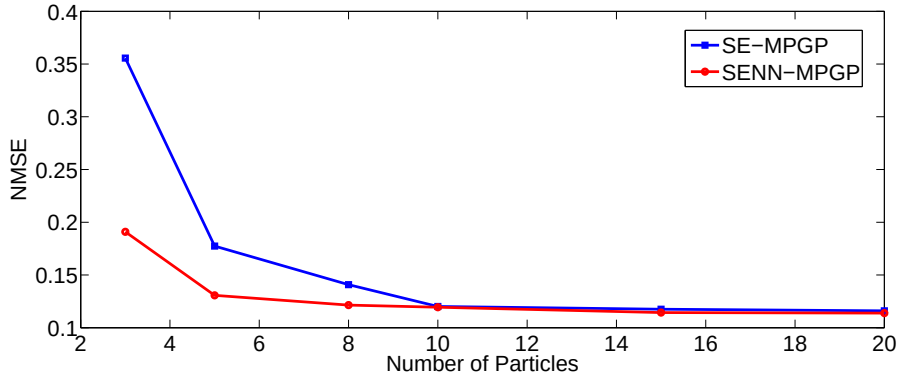


(b) MNLP

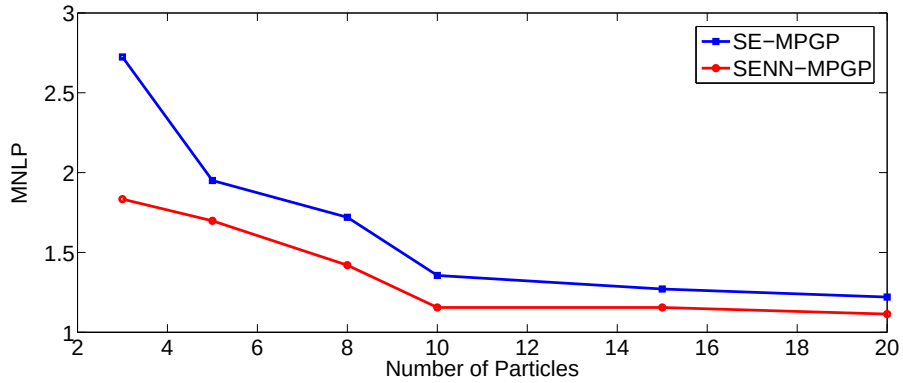


(c) Running Time

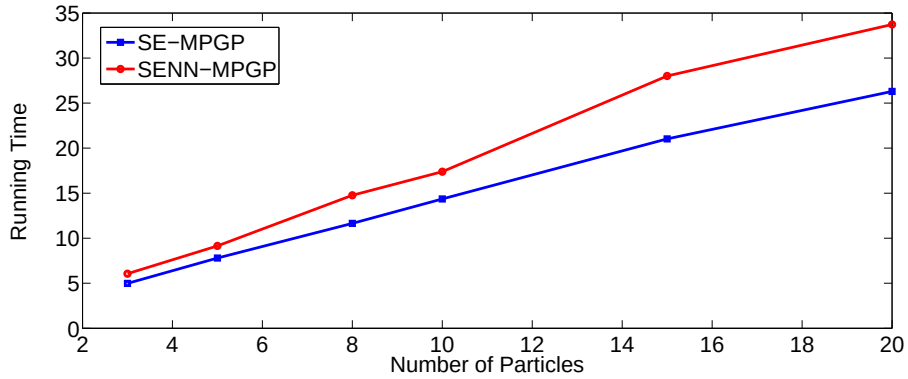
FIGURE 4.7: Accuracy and efficiency evaluation for f_1 as a function of the number of the particles.



(a) NMSE



(b) MNLP



(c) Running Time

FIGURE 4.8: Accuracy and efficiency evaluation for f_2 as a function of the number of the particles.

Table 4.2: Benchmarks Comparison for temperature and pendulum datasets.

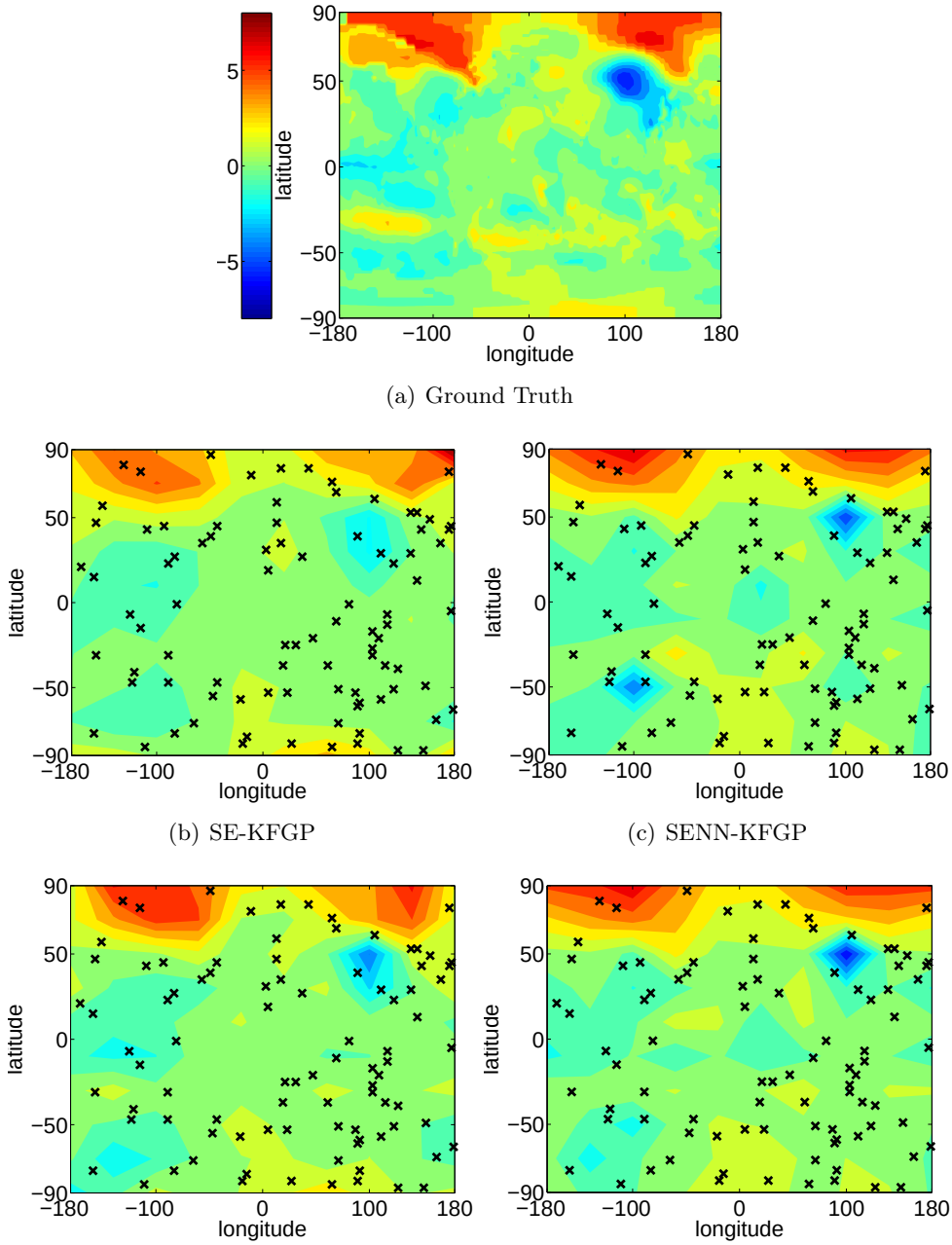
Temperature	NMSE	MNLP	RTime	Pendulum	NMSE	MNLP	RTime
5-SPGP	0.48	1.62	181.3s	10-SPGP	0.61	1.98	16.54s
10-SSGP	0.27	1.33	97.16s	10-SSGP	1.04	10.85	23.59s
5-SE-MPGP	0.11	1.05	50.99s	20-SE-MPGP	0.63	2.20	7.04s
5-SENN-MPGP	0.10	1.16	59.25s	20-SENN-MPGP	0.58	2.12	8.60s

In the **second** experiment, we evaluate the estimation error of our MPGP using the second test data. Firstly, we run all the methods to compute the NMSE and MNLP as a function of the number of the training data collections. From Figure 4.11 we can see that, when the number of the training data collections increases, the NMSE and MNLP error for all the methods decrease. However the NMSE and MNLP for our MPGP are much lower than KFGP since online hyper-parameter learning further improves the estimation performance. Furthermore, our SENN-MPGP is more accurate than SE-MPGP, and this shows that SENN-MPGP successfully models the spatial non-stationarity of the temperature data by using the non-stationary kernel function. Secondly, we evaluate the accuracy and efficiency of our MPGP as a function of the number of the particles. For each value of the number of the particles, we run SE-MPGP, SENN-MPGP 3 times to evaluate the average NMSE, MNLP and running time. It is shown in Figure 4.12 that our SENN-MPGP fits the data better than SE-MPGP but the trade-off is the longer running time.

In the **third** experiment, we compare our MPGP with the benchmark fast GP approaches. All the denotations are same as the third experiment of the synthetic data sets. We also randomly interrupt the order of training subsets for the robustness consideration. From Table 4.2, the comparison results show that our MPGP uses a shorter running time with a better estimation performance than SPGP and SSGP.

Pendulum Data Set

This is a small data set which contains 315 training points. In Lázaro-Gredilla et al. (2010), it is mentioned that SSGP model tends to overfit this data due to the gradient ascent optimization. We are interested in whether our method can successfully capture the nonlinear property of this pendulum data. We firstly collect the training data 9 times, and 35 training data for each collection. Then, 100 test points are randomly selected for evaluating the performance. From Table 4.2, our SENN-MPGP obtains the estimation with the fastest speed and the smallest NMSE among all the methods, and the MNLP is competitive to SPGP.



(a) Ground Truth

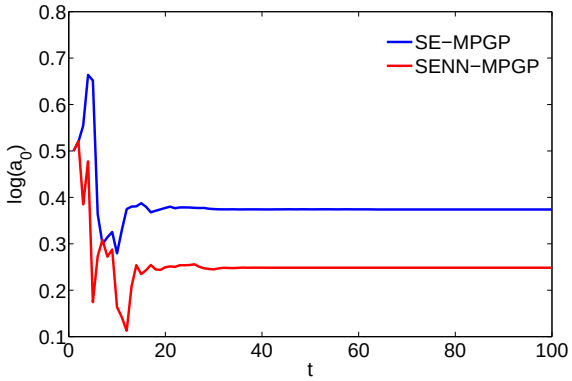
(b) SE-KFGP

(c) SENN-KFGP

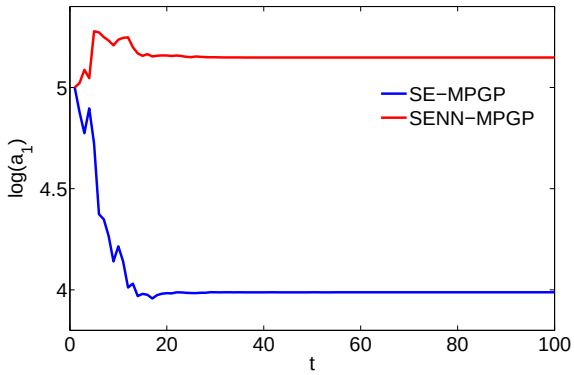
(d) Our SE-MPGP

(e) Our SENN-MPGP

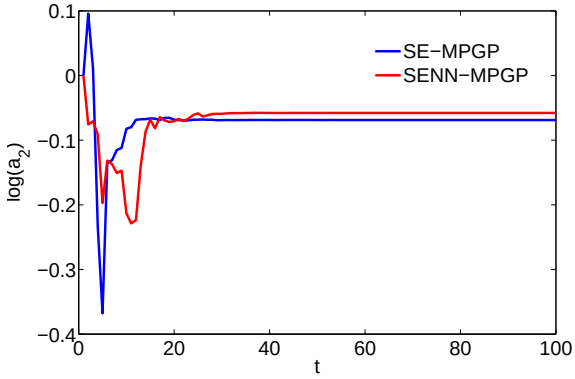
FIGURE 4.9: The temperature estimation at $t = 100$. The black crosses are the 100th collection of the training data ponits.



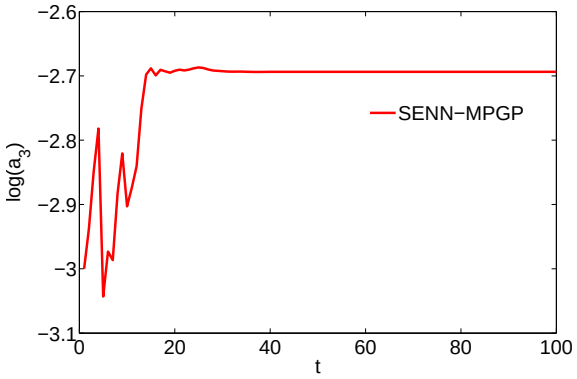
(a) Estimation of $\log(a_0)$ over time



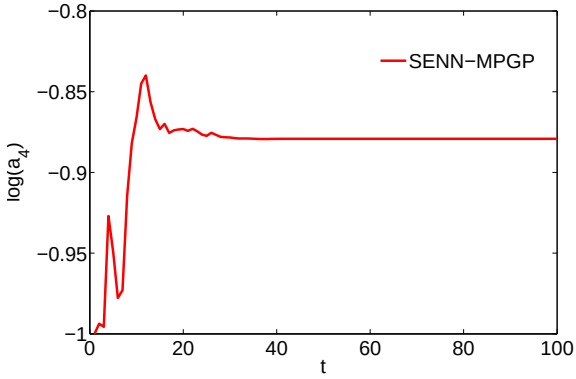
(b) Estimation of $\log(a_1)$ over time



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

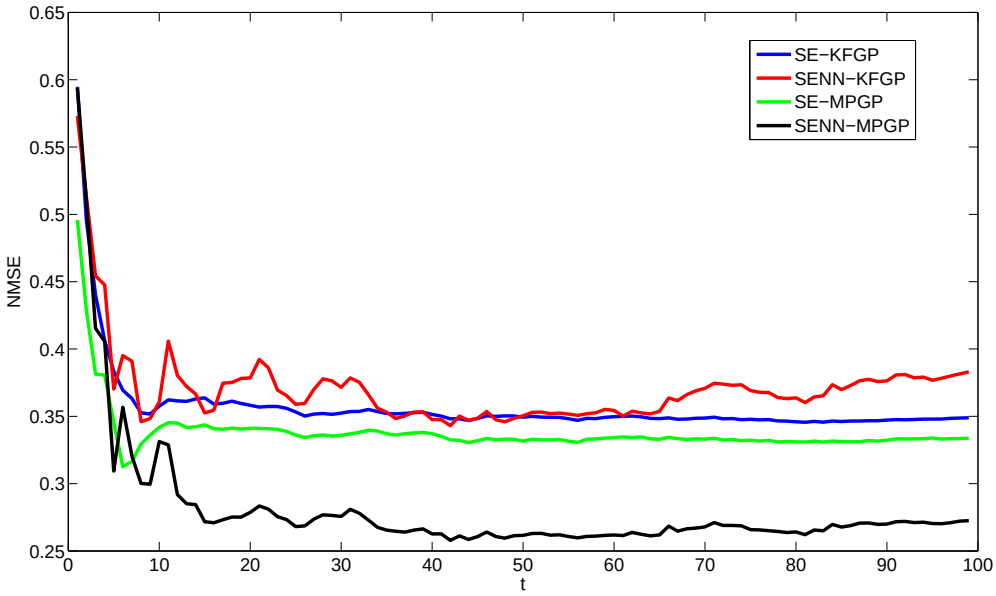


(d) Estimation of $\log(a_3)$ over time

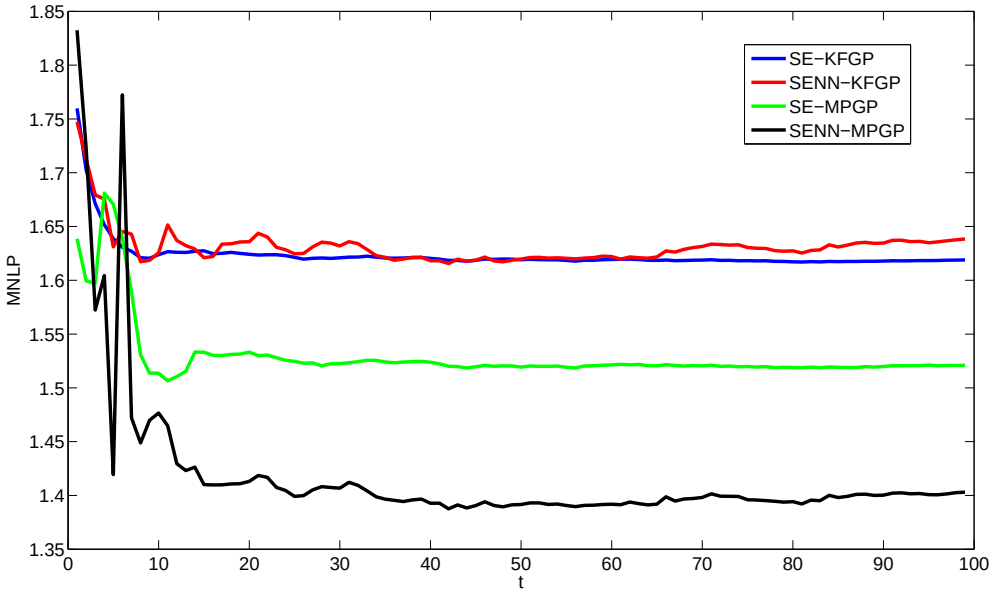


(e) Estimation of $\log(a_4)$ over time

FIGURE 4.10: Online hyper-parameter learning for the temperature data set.

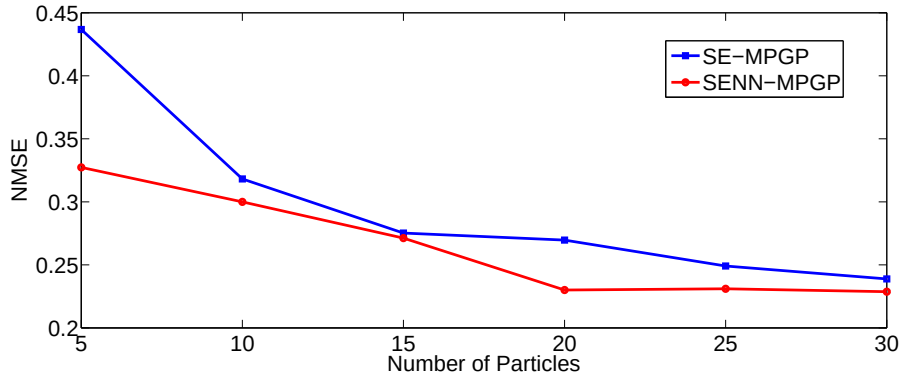


(a) NMSE for the temperature data over time

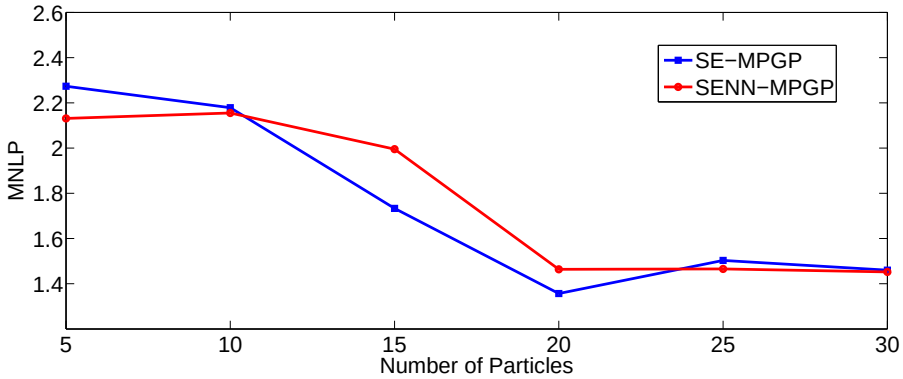


(b) MNLP for the temperature data over time

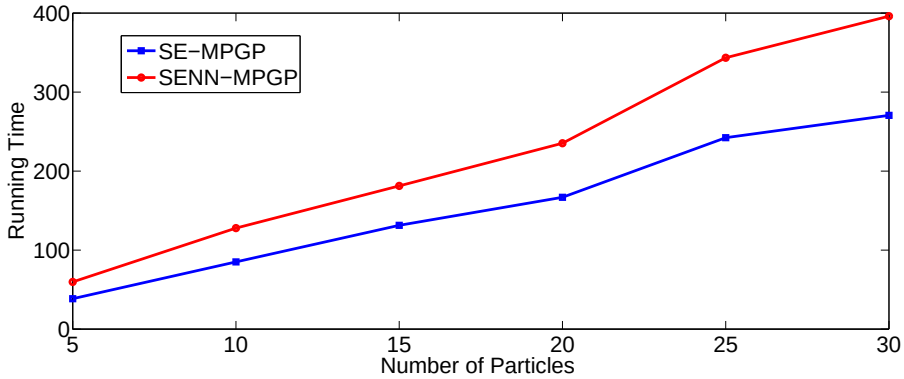
FIGURE 4.11: Accuracy evaluation for the temperature data over time.



(a) NMSE



(b) MNLP



(c) Running Time

FIGURE 4.12: Accuracy and efficiency evaluation for the temperature data as a function of the number of the particles.

4.2.6 Summary

In Wang and Chaib-draa (2012a), we mainly focus on one of the most important data properties, i.e., the large size of the data set which leads to a challenging computation burden in GPs. We have proposed a novel Bayesian filtering framework for GP regression. Our MPGP framework does not only estimate the function value successfully, but it is also a new online technique to learn the unknown static hyper-parameters. The small training set at each iteration naturally reduces the computation load, while the estimation performance is improved over iteration due to the fact that the recursive filtering would propagate the previous estimation to enhance the current estimation. In comparison with other benchmark fast GP approaches, we have shown that our MPGP provides a robust estimation with a competitively computational speed.

In the following, we mainly focus on another important data property, non-stationarity where the correlation between two function values does not only depend on the distance between two inputs but also associates with the locations of these two inputs. Since using the non-stationary covariance function directly may still restrict the flexibility of the GP model (Rasmussen and Williams, 2006), we propose to design a flexible Bayesian inference mechanism for GP regression to capture the non-stationarity of a data distribution.

4.3 A KNN Based Kalman Filter Gaussian Process Regression

The standard Gaussian process (GP) regression is often deteriorated when the data set is non-stationarity (Paciorek and Schervish, 2003). A popular approach is to predefine a non-stationary covariance function for a GP (Paciorek and Schervish, 2003; Rasmussen and Williams, 2006). However, this approach is highly dependent on the prior knowledge of the nonstationarity that is often unknown (Plagemann, 2008; Adams and Stegle, 2008). Hence its performance is limited for real-world data sets.

In the following contribution (Wang and Chaib-draa, 2013), we proposed a K nearest neighbor based Kalman filter Gaussian process (KNN-KFGP) to explore a flexible inference mechanism for GPs in order to capture non-stationarity. Inspired by *Bayesian Estimation for Gaussian Processes*, we firstly construct a state space model based on a KNN-driven data grouping. Then we design an efficient Kalman filtering framework to infer the latent function values in a recursive Bayesian estimation framework. Due to the fact that KNN allows each test point to find its strongly-correlated local training subset, our KNN-KFGP is a suitable way to address the non-stationary problems while preserving computational efficiency.

4.3.1 KNN Constructed State Space Model

Non-stationarity is related to input-dependent smoothness, so a suitable way to interpret non-stationarity at a number of test input locations $\{\mathbf{x}_\star^t\}_{t=1}^m$ is to capture the local data structure

of these points. To achieve this, we propose to use KNN to search the strongly correlated local structure of each test point. Concretely, for $\mathbf{x}_\star^t$ we find its K_t nearest training inputs $X_t = \{\mathbf{x}_i^t\}_{i=1}^{K_t}$ according to the Euclidian distance. Then we use X_t and the corresponding outputs $\mathbf{y}_t = \{y_i^t\}_{i=1}^{K_t}$ as a small training set $(X_t, \mathbf{y}_t)$ for $\mathbf{x}_\star^t$, where y_i^t is generated by a nonlinear function $f(\mathbf{x}_i^t)$ of $\mathbf{x}_i^t$ with an additive Gaussian noise $\mathcal{N}(0, \sigma^2)$. For simplicity, we denote $X_t^c = X_t \cup \mathbf{x}_\star^t$ and $f_t^c = f(X_t^c)$.

Based on the KNN driven data grouping, we now use a GP to design a state space model (SSM) which consists of a prediction model and an observation model. Suppose the latent function $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$, where we choose an automatic relevance determination (ARD) squared exponential covariance function in this contribution,

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 exp[-0.5(\mathbf{x} - \mathbf{x}')^T L^{-2} (\mathbf{x} - \mathbf{x}')].$$

The length-scale matrix is $L = diag([l_1 \cdots l_d])$. For convenience, all the hyper-parameters and the standard deviation of the observation noise σ are denoted into a vector $\theta = [\sigma \;\; \sigma_f \;\; l_1 \cdots l_d]^T$.

Similar to (Wang and Chaib-draa, 2012a), the prediction model of our SSM refers to a transition between the $(t-1)$-th and t-th latent states which are the latent function values of the $(t-1)$-th and t-th data subset. As $f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$, it is straightforward to find that the conditional distribution $p(f_t^c | f_{t-1}^c, X_{t-1}^c, X_t^c, \theta)$ is Gaussian,

$$p(f_t^c | f_{t-1}^c, X_{t-1}^c, X_t^c, \theta) = \mathcal{N}(G(\theta) f_{t-1}^c, Q(\theta)), \tag{4.17}$$

where

$$G(\theta) = K_\theta(X_t^c, X_{t-1}^c) K_\theta^{-1}(X_{t-1}^c, X_{t-1}^c), \tag{4.18}$$

$$Q(\theta) = K_\theta(X_t^c, X_t^c) - K_\theta(X_t^c, X_{t-1}^c) K_\theta^{-1}(X_{t-1}^c, X_{t-1}^c) K_\theta(X_t^c, X_{t-1}^c)^T. \tag{4.19}$$

In fact, this $p(f_t^c | f_{t-1}^c, X_{t-1}^c, X_t^c, \theta)$ is the prediction model of our SSM,

$$f_t^c = G(\theta) f_{t-1}^c + v_t^f, \tag{4.20}$$

which is a linear evolution between f_t^c and f_{t-1}^c with an additive Gaussian noise $v_t^f \sim \mathcal{N}(\mathbf{0}, Q(\theta))$.

Additionally, the observation model can also be obtained straightforwardly,

$$\mathbf{y}_t = H_t f_t^c + v_t^y, \tag{4.21}$$

where $H_t = [I_{K_t} \;\; \mathbf{0}]$ is an index matrix so that $H_t f_t^c = f(X_t)$, and the observation noise is Gaussian distributed $v_t^y \sim N(\mathbf{0}, R(\theta) = \sigma^2 I)$. Therefore, our SSM is fully specified by Equation 4.20 and 4.21. The graph model is shown in Figure 4.13.

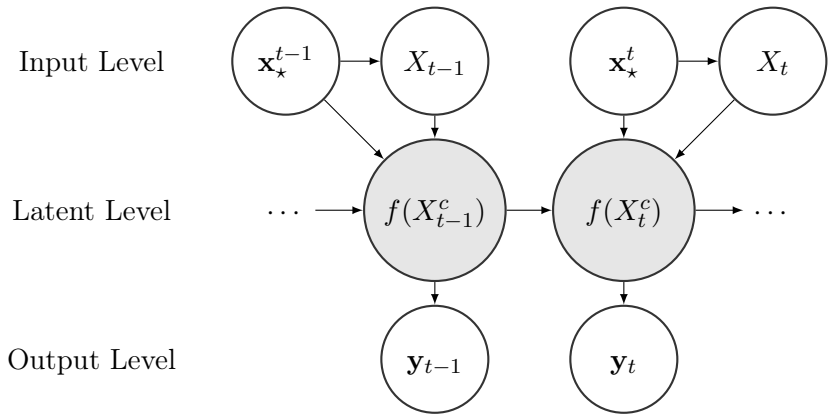


FIGURE 4.13: State space model that is established by a test-input-driven KNN data collection procedure. The arrows between $\mathbf{x}_\star^{t-1}$ and X_{t-1}, $\mathbf{x}_\star^{t}$ and X_t represent the KNN mechanism.

4.3.2 Bayesian Inference Using Kalman Filter

After constructing our SSM, we can now consider a regression task as a state estimation problem. Given an initialized θ, Equation 4.20 and 4.21 in our SSM are linear with additive Gaussian noises. The well-known Kalman Filter (KF) provides an optimal solution with respect to maximum-a-posteriori (MAP) (Kalman, 1960).

Suppose that the $(t-1)$-th posterior distribution $p(f_{t-1}^c|\mathbf{y}_{1:t-1}, X_{1:t-1}, \mathbf{x}_\star^{1:t-1}, \theta)$ is Gaussian,

$$p(f_{t-1}^c|\mathbf{y}_{1:t-1}, X_{1:t-1}, \mathbf{x}_\star^{1:t-1}, \theta) = \mathcal{N}(f_{t-1|t-1}^c, P_{t-1|t-1}^c).$$

The first step is to predict the t-th hidden function values by combining the information of the transition model $p(f_t^c|f_{t-1}^c, X_{t-1:t}, \mathbf{x}_\star^{t-1:t}, \theta)$ (directly from Equation 4.17) with the $(t-1)$-th posterior $p(f_{t-1}^c|\mathbf{y}_{1:t-1}, X_{1:t-1}, \mathbf{x}_\star^{1:t-1}, \theta)$,

$$\begin{aligned}
&p(f_t^c|\mathbf{y}_{1:t-1}, X_{1:t}, \mathbf{x}_\star^{1:t}, \theta) \\
&= \int p(f_t^c|f_{t-1}^c, X_{t-1:t}, \mathbf{x}_\star^{t-1:t}, \theta) p(f_{t-1}^c|\mathbf{y}_{1:t-1}, X_{1:t-1}, \mathbf{x}_\star^{1:t-1}, \theta) df_{t-1}^c \\
&= \int \mathcal{N}(G(\theta) f_{t-1}^c, Q(\theta)) \mathcal{N}(f_{t-1|t-1}^c, P_{t-1|t-1}^c) df_{t-1}^c \\
&= \mathcal{N}(G(\theta) f_{t-1|t-1}^c, \quad G(\theta) P_{t-1|t-1}^c G(\theta)^T + Q(\theta)). \qquad (4.22)
\end{aligned}$$

We denote $p(f_t^c|\mathbf{y}_{1:t-1}, X_{1:t}, \mathbf{x}_\star^{1:t}, \theta)$ as $\mathcal{N}(f_{t|t-1}^c, P_{t|t-1}^c)$, then the prediction step is fully expressed by the mean $f_{t|t-1}^c$ and the covariance $P_{t|t-1}^c$,

$$f_{t|t-1}^c = G(\theta) f_{t-1|t-1}^c, \qquad (4.23)$$

$$P_{t|t-1}^c = G(\theta) P_{t-1|t-1}^c G(\theta)^T + Q(\theta). \qquad (4.24)$$

Algorithm 7 KNN-KFGP Regression.

1: Training the hyper-parameter vector θ using gradient optimization with an extra data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{S}$
2: **for** $t = 1$ **to** m **do**
3: Applying KNN based data construction to find the training data $(X_t, \mathbf{y}_t) = \{(\mathbf{x}_i, y_i)\}_{i=1}^{K_t}$ for $\mathbf{x}_\star^t$
4: Using θ to specify $G(\theta)$, $Q(\theta)$, $R(\theta)$ in (4.18-4.19) and (4.21) for state space model construction
5: Kalman Predict Step : Equation (4.23-4.24)
6: Kalman Update Step : Equation (4.27-4.29)
7: **end for**

The second step is to update the predictive estimation with the tth observed output $\mathbf{y}_t$,

$$p(f_t^c|\mathbf{y}_{1:t}, X_{1:t}, \mathbf{x}_\star^{1:t}, \theta) = \frac{p(\mathbf{y}_t|f_t^c, \theta, X_t, \mathbf{x}_\star^t)p(f_t^c|\mathbf{y}_{1:t-1}, X_{1:t}, \mathbf{x}_\star^{1:t}, \theta)}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \theta, X_{1:t}, \mathbf{x}_\star^{1:t})}, \tag{4.25}$$

where the likelihood distribution $p(\mathbf{y}_t|f_t^c, \theta, X_t, \mathbf{x}_\star^t)$ is directly from Equation 4.21, the predictive distribution $p(f_t^c|\mathbf{y}_{1:t-1}, X_{1:t}, \mathbf{x}_\star^{1:t}, \theta)$ is from Equation 4.22, and the marginal distribution is an analytical integral,

$$\begin{aligned}
&p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \theta, X_{1:t}, \mathbf{x}_\star^{1:t}) \\
&= \int p(\mathbf{y}_t|f_t^c, \theta, X_t, \mathbf{x}_\star^t)p(f_t^c|\mathbf{y}_{1:t-1}, X_{1:t}, \mathbf{x}_\star^{1:t}, \theta)df_t^c \\
&= \int \mathcal{N}(H_t f_t^c, R(\theta))\mathcal{N}(f_{t|t-1}^c, P_{t|t-1}^c)df_t^c \\
&= \mathcal{N}(H_t f_{t|t-1}^c, H_t P_{t|t-1}^c H_t^T + R(\theta)).
\end{aligned} \tag{4.26}$$

Finally we denote $p(f_t^c|\mathbf{y}_{1:t}, X_{1:t}, \mathbf{x}_\star^{1:t}, \theta) = \mathcal{N}(f_{t|t}^c, P_{t|t}^c)$, then put Equation 4.21, 4.22 and 4.26 into Equation 4.25, we obtain the update with the Kalman Gain Γ_t,

$$\Gamma_t = P_{t|t-1}^c H_t^T (H_t P_{t|t-1}^c H_t^T + R(\theta))^{-1}, \tag{4.27}$$

$$f_{t|t}^c = f_{t|t-1}^c + \Gamma_t(\mathbf{y}_t - H_t f_{t|t-1}^c), \tag{4.28}$$

$$P_{t|t}^c = P_{t|t-1}^c - \Gamma_t H_t P_{t|t-1}^c. \tag{4.29}$$

The whole Bayesian inference procedure is summarized in Algorithm 7. Similar to other GP approaches, we first initialize the hyperparameters with a small extra data set (Algorithm 7, Line 1). Then we perform KNN search to group data subsets and construct our SSM (Algorithm 7, Line 3-4). Finally we use Kalman filter to infer the latent function values (Algorithm 7, Line 5-6).

Since our KNN-KFGP allows each test point to find its strongly-correlated training subset, it is a suitable way to capture the local data property in order to deal with non-stationarity. Additionally, due to the fact our KNN-KFGP inherits the computational merits of Kalman

filter Gaussian process (KFGP) (Reece and Roberts, 2010), it is also a computationally efficient GP approach where the computation load is mainly governed by Kalman filter $O(mK^3)$ and KNN search $O(mn)$ after various pre-computations (Urtasun and Darrell, 2008). The K is the maximum value of $\{K_t\}_{t=1}^m$, m is the number of test inputs and n is the number of the training points.

4.3.3 Experiments

To show the effectiveness of our KNN-KFGP to handle nonstationarity, we perform a number of experiments on several simulation and real-world data sets in the following. For all the data sets, the accuracy is evaluated by the test Standardized Mean Square Error (SMSE) [3] and the test Mean Negative Log Probability (MNLP).

Mobile Robot Perception

The perception of the environment is a fundamental task for a mobile robot as it is essential to correctly interpret the sensor information in non-stationary environments to find the precise location. Here we consider a perception task to evaluate how well our KNN-KFGP deal with non-stationarity. Concretely, a mobile robot is located at $(0, 0)$. To find an exit, it uses its range sensors for obstacle detection. We simulate 200 training points where the input is a bearing angle in $[0, \pi]$ and the output is the corresponding distance measurement. The test inputs are from 0 to π with an interval $\pi/180$. The observations at different sides of the exit show the discontinuity of the measured data, which is a case of nonstationarity.

In this experiment, we compare our KNN-KFGP to the stationary GP, KFGP and local GP (LGP in Chapter 2) which uses local GP experts to capture nonstationarity (Urtasun and Darrell, 2008). In KFGP and KNN-KFGP, K is the size of the training subset for each test point. In LGP, K is the number of the local GP experts. We first learn the hyper-parameters using the gradient optimization with the full training set, and then run all the methods from $K = 1$ to 5. We found that the best performance of our KNN-KFGP is achieved at $K = 2$, while the best performance of KFGP and LGP is achieved at $K = 5$. This illustrates that the accuracy of our KNN-KFGP does not monotonically increase in the non-stationary case when K is increasing. The underlying reason is that, as K is getting larger, some irrelevant training points may be selected for each test point. That is also one reason why the fully-correlated stationary GP cannot correctly model non-stationarity.

We show the best results of all the methods in Figure 4.14 and Table 4.3. In Figure 4.14(a), the standard GP mistakenly predicts the exit as the obstacle even though the robot has detected the data on both sides of the exit. In Figure 4.14(b), the prediction of KFGP fails since the randomly selected training points in this case are disturbed and thus deteriorate the

3. It is also called as test Normalized Mean Square Error (NMSE).

Table 4.3: Accuracy evaluation for the robot perception data set and the land-surface precipitation data set.

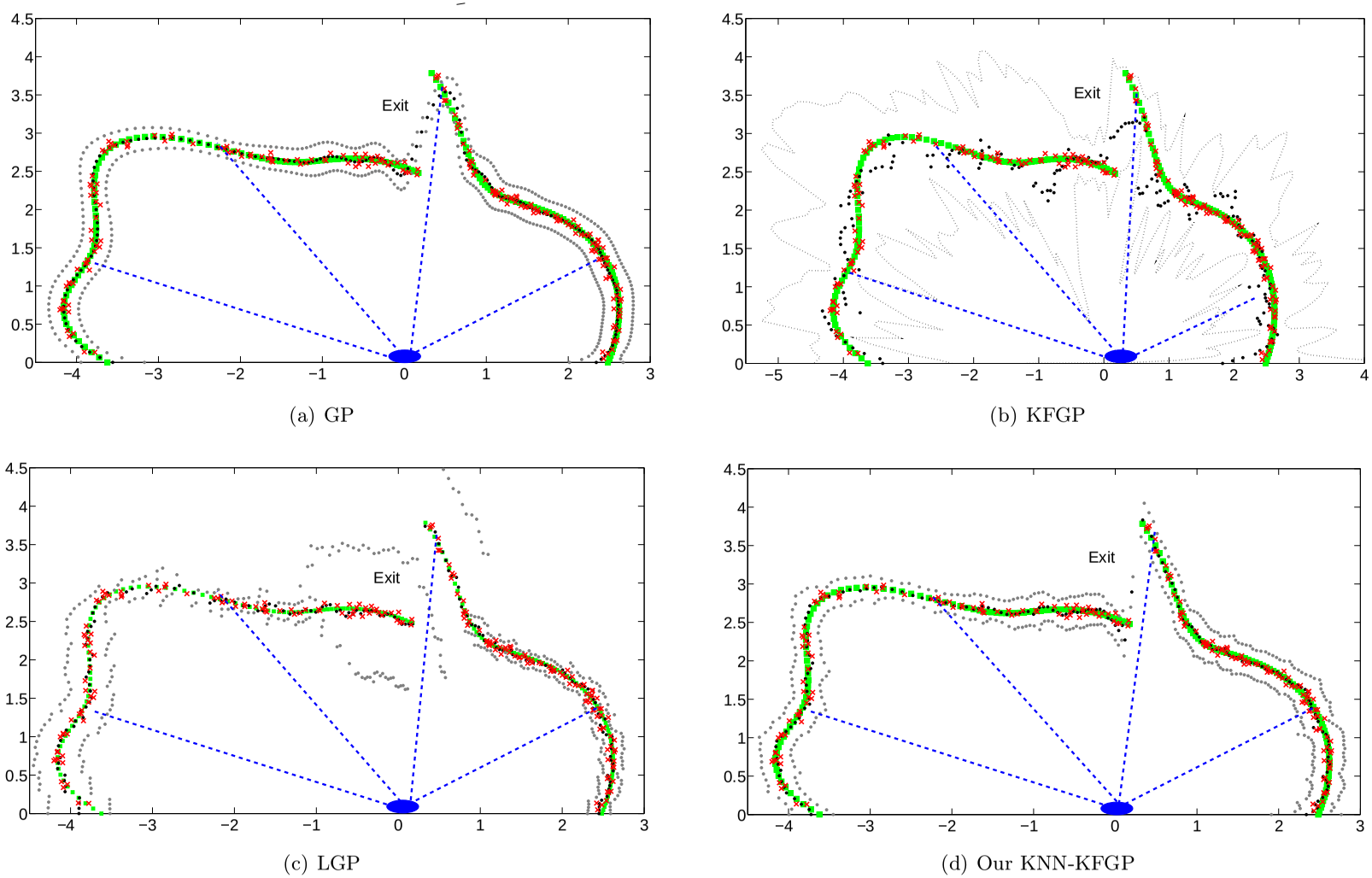
Robot Perception	SMSE	MNLP	Land Precipitation	SMSE	MNLP
GP	0.0130	-2.1775	GP	0.0128	7.6863
LGP	0.0093	-2.3216	LGP	0.0036	7.5979
KFGP	0.2218	0.0791	KFGP	1.2582	12.1196
KNN-KFGP	0.0057	-2.5060	KNN-KFGP	0.0027	7.1255

performance of KFGP. In Figure 4.14(c), LGP correctly finds the exit but the uncertainty at the exit is high. Finally, in Figure 4.14(d) our KNN-KFGP successfully models the exit since the training subset for each test is strongly correlated to that test. From Table 4.3, it is also shown that LGP and our KNN-KFGP are suitable for non-stationarity modeling, but our KNN-KFGP performs better with higher accuracy than LGP.

Global Land-Surface Precipitation

In practice, many environmental data sets usually exhibit non-stationarity (Paciorek and Schervish, 2003). Therefore, we choose a global land-surface precipitation (January 2010) data set to evaluate whether our KNN-KFGP can handle real-life non-stationarity [4]. There are 3,306 points in the training set. For each point, the input is the location and the output is the precipitation value. The test set is based on a regular grid with a spatial resolution of $2.5^o \times 2.5^o$ latitude by longitude. In this experiment, we also compare our KNN-KFGP to the stationary GP, KFGP and LGP where the meaning of K in these approaches is the same as the one in the robot perception experiment. Moreover, we train the hyper-parameters using the gradient optimization with a randomly selected training subset including 1,000 points, and then run all the methods from $K = 1$ to 10. We found that our KNN-KFGP shows the best performance when $K = 5$, KFGP and LGP represent the best performance when $K = 10$. The best results for all the approaches are illustrated in Figure 4.15 and Table 4.3. In Figure 4.15, the prediction of KFGP almost fails due to the random training set selection. In contrast, our KNN-KFGP flexibly learns the land-surface precipitation model. The underlying reason is that each test point is learned by using its strongly correlated training set and the predict-update KF mechanism preserves the accuracy. From Table 4.3, it is also shown that LGP and our KNN-KFGP may be considered as suitable approaches for this non-stationary case, but our KNN-KFGP performs better than LGP. Finally, it is worth mentioning that the running time of our KNN-KFGP is 475s which is much more efficient than GP (15374s).

4. The data set is available at http ://www.cgd.ucar.edu/cas/catalog/surface/precip/gpcc.html



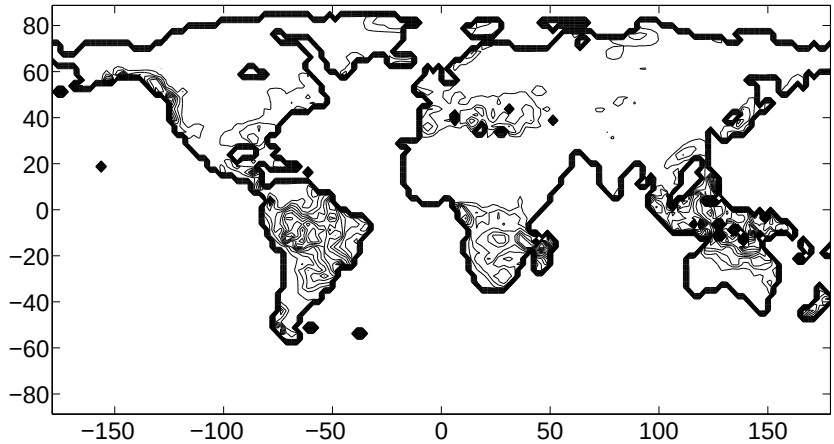
(a) GP

(b) KFGP

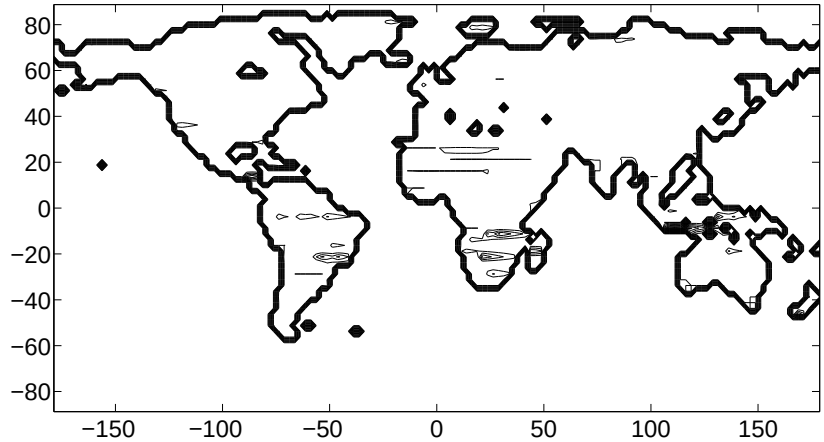
(c) LGP

(d) Our KNN-KFGP

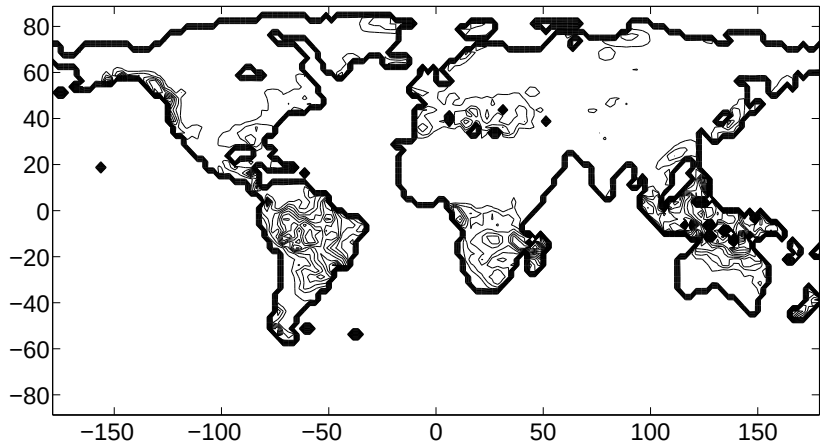
FIGURE 4.14: Mobile Robot Perception. A mobile robot (blue ellipse at (0,0)) uses its range sensors (four blue dashed lines) to obtain distance measurements (red crosses) for obstacle (green squares) detection. The prediction mean for all the methods are represented by black dots. The 95% confidence interval for KFGP are represented by grey dotted line, for other methods it is represented by grey dots.



(a) Observation



(b) KFGP



(c) Our KNN-KFGP

FIGURE 4.15: Global Land-Surface Precipitation. The contour in all three subplots represents the level of precipitation value. When the contours are concentrated, the precipitation value is high. Otherwise, the the precipitation value is low. The subplots of KFGP and KNN-KFGP show the predicted mean of the global land-surface precipitation.

4.3.4 Summary

In Wang and Chaib-draa (2013), we have proposed a novel K nearest neighbor based Kalman filter Gaussian process (KNN-KFGP) regression. Firstly, small training subsets, which are constructed by a test-driven KNN search procedure, are used to establish a GP based state space model. Then Kalman filter is applied to infer the latent function values in an efficient predict-update manner. Due to the fact that KNN search provides each test with its strongly correlated training set, our KNN-KFGP can more suitably capture non-stationarity than the stationary GP.

Since our contributions MPGP (Wang and Chaib-draa, 2012a) and KNN-KFGP (Wang and Chaib-draa, 2013) have opened a door to take advantage of Bayesian estimation to address the difficulties of GP in a dynamical filtering framework, in the following we further make use of the merits of *Bayesian Estimation for Gaussian Process* to deal with the temporal-related challenges in time-varying applications.

4.4 Particle Based Gaussian Processes for Time-Varying Applications

Even though the GP is a popular non-parametric model, the performance of GP is often deteriorated in time-varying applications where the data points are sequentially ordered and often exhibit temporal-related non-stationarity and heteroscedasticity. In order to interpret these challenging data properties correctly, we proposed two particle based GP approaches in Wang and Chaib-draa (2015a) where time-varying Bayesian inference is performed in a sequential Monte Carlo manner. By evaluating our approaches on four distinct time series (including both one-dimensional and multi-dimensional input cases), we showed that our particle based GPs outperform several benchmark GP approaches by capturing the time-varying data properties efficiently and accurately.

4.4.1 Fit Time-Varying Data with Gaussian Processes

Since we mainly focus on time-varying applications, we consider that the data points are sequentially-ordered $D_{1:T} = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$ where the output $y_t \in R$ is generated by a latent function of the input vector $\mathbf{x}_t \in R^{d_x}$ with an additive Gaussian noise $v_t^y \sim \mathcal{N}(0, \sigma_{y,t}^2)$,

$$y_t = f(\mathbf{x}_t) + v_t^y. \tag{4.30}$$

The t-th data point $D_t = (\mathbf{x}_t, y_t)$ is made available at time t. For simplicity, we denote $f(\mathbf{x}_t) = f_t$ and $f(\mathbf{x}_{1:T}) = f_{1:T}$.

As we already stated in Chapter 2, a popular nonparametric approach to infer $f_{1:T}$ is to put a Gaussian process (GP) prior over the latent function since the elegant Gaussian properties

of GPs make Bayesian inference analytically tractable (Rasmussen and Williams, 2006; Calderhead et al., 2008; Vaerenbergh et al., 2012). Specifically, we denote $f(\mathbf{x}) \sim \mathcal{GP}(0, k_\zeta(\mathbf{x}, \mathbf{x}'))$ where $k_\zeta(\mathbf{x}, \mathbf{x}') = \sigma^2 k_\ell(\mathbf{x}, \mathbf{x}')$ with $\zeta = [\sigma^2, \ell]^T$ that consists of an amplitude parameter σ^2 and a hyper-parameter vector ℓ of the unscaled covariance function $k_\ell(\mathbf{x}, \mathbf{x}')$. In this contribution, we choose a popular $k_\ell(\mathbf{x}, \mathbf{x}')$ that is the Automatic Relevance Determination (ARD) squared exponential kernel (Rasmussen and Williams, 2006), i.e.,

$$k_\ell(\mathbf{x}, \mathbf{x}') = \exp[-0.5(\mathbf{x} - \mathbf{x}')^T L^{-1} (\mathbf{x} - \mathbf{x}')],$$

L is diagonal with the ARD parameter vector $\ell = [l_1, ..., l_{d_x}]^T$.

To fit the temporal data, Bayesian inference with GPs addresses the joint posterior of latent function values and model parameters

$$p(f_{1:T}, \zeta, \sigma_y^2 | D_{1:T}) = p(f_{1:T} | \zeta, \sigma_y^2, D_{1:T}) p(\zeta, \sigma_y^2 | D_{1:T}), \tag{4.31}$$

in the following two steps where the variance of the noise $\sigma_{y,t}^2$ is assumed to be time-invariant σ_y^2. Firstly, ζ and σ_y^2 are inferred to fit the data. A popular method is to maximize $p(\zeta, \sigma_y^2 | D_{1:T})$ using a gradient based optimizer (Rasmussen and Williams, 2006). Secondly, $f_{1:T}$ is estimated with the learned ζ and σ_y^2. Due to $f(\mathbf{x}) \sim \mathcal{GP}(0, k_\zeta(\mathbf{x}, \mathbf{x}'))$, the marginal posterior $p(f_{1:T} | \zeta, \sigma_y^2, D_{1:T})$ can be expressed as a Gaussian distribution $\mathcal{N}(\hat{f}, \hat{P})$ (Rasmussen and Williams, 2006; Calderhead et al., 2008) with the mean,

$$\hat{f} = K_\zeta(\mathbf{x}_{1:T}, \mathbf{x}_{1:T}) [K_\zeta(\mathbf{x}_{1:T}, \mathbf{x}_{1:T}) + \sigma_y^2 I]^{-1} \mathbf{y},$$

where $\mathbf{y} = [y_1 \cdots y_T]^T$, and the covariance,

$$\hat{P} = \sigma_y^2 K_\zeta(\mathbf{x}_{1:T}, \mathbf{x}_{1:T}) [K_\zeta(\mathbf{x}_{1:T}, \mathbf{x}_{1:T}) + \sigma_y^2 I]^{-1},$$

where each entry of $K_\zeta(\mathbf{x}_{1:T}, \mathbf{x}_{1:T})$ is computed by using $k_\zeta(\mathbf{x}, \mathbf{x}')$.

However, this standard GP is often limited for time-varying applications (Vaerenbergh et al., 2012; Csató and Opper, 2002; Plagemann, 2008; Reece and Roberts, 2010; Hartikainen and Särkkä, 2010).

One important reason is that it is difficult for a standard GP to interpret time-varying data properties, such as temporal-related *non-stationarity* (the correlation between two temporal-linked function values often changes over time) and temporal-related *heteroscedasticity* (the observation noise often changes over time), by learning the static hyper-parameter vector of the stationary kernel ζ and the static variance of the observation noise σ_y^2 (Vaerenbergh et al., 2012; Paciorek and Schervish, 2003; Kersting et al., 2007). To capture the non-stationarity, one can directly use several non-stationary covariance functions with GPs (Rasmussen and Williams, 2006; Paciorek and Schervish, 2003). However, using the predefined non-stationary covariance functions requires prior knowledge of the non-stationarity that is often hard to

obtain (Adams and Stegle, 2008). To capture heteroscedasticity, one can use several heteroscedastic GP approaches by putting another GP prior over the noise variance (Kersting et al., 2007; Lázaro-Gredilla and Titsias, 2011). However, the computation burden of these methods is often heavy when the number of data points is a few thousands. Additionally, a sparse pseudo-input GP, which was originally proposed for computation reduction, can also model heteroscedasticity (Snelson and Ghahramani, 2005). However, this approach may be deteriorated when the size of the pseudo set is large and/or the input space is high dimensional (Snelson and Ghahramani, 2005).

Another important reason is that a standard GP deals with the temporal-ordered data points in a batch way. As a result, it is not suitable to use GPs to make inference and prediction for data streams. Several online variants of GPs have been investigated for temporal-related applications (Csató and Opper, 2002; Reece and Roberts, 2010; Hartikainen and Särkkä, 2010; Nguyen-Tuong et al., 2008; Ko and Fox, 2008; Turner et al., 2010). However, these approaches are not suitable to capture the temporal-related non-stationarity and heteroscedasticity due to the time-invariant, offline-learned GP hyper-parameters. Recently, a kernel recursive least squared tracker was introduced to interpret the time-varying properties in an online fashion (Vaerenbergh et al., 2012). Its performance is however limited when the temporal data points are highly non-stationary and heteroscedastic, since in this method the time-varying properties are achieved by learning a temporal-related scale parameter instead of GP hyper-parameters. Likewise, a marginalized particle GP was developed in Wang and Chaib-draa (2012a) to learn GP hyper-parameters directly in a recursive Bayesian manner. In this approach, however, the hyper-parameters are assumed to be time-invariant (the transition of hyper-parameters is 'artificial'), and thus its performance is also reduced when the data sets exhibit time-varying properties.

In the following contribution (Wang and Chaib-draa, 2015a), we propose two novel particle based Gaussian process approaches to interpret sequentially-ordered, time-varying data in a unifying, online Bayesian framework.

4.4.2 Gaussian Process Based State Space Model

To interpret the time-varying properties in the sequentially-ordered data, we propose to incorporate GP into state space model (SSM) which consist of the observation model and the transition model.

Observation Model

The observation model is the same as (4.30). Note that the time-varying noise variance $\sigma^2_{y,t}$ is the key to capture the temporal-related *heteroscedasticity*.

Transition Model

GP Based State Transition To reflect the temporal-ordered data characteristics, we propose to establish the transition between f_{t-1} and f_t. Due to $f(\mathbf{x}) \sim \mathcal{GP}(0, k_\zeta(\mathbf{x}, \mathbf{x}'))$, the joint distribution $p(f_t, f_{t-1}|\mathbf{x}_t, \mathbf{x}_{t-1}, \zeta)$ is Gaussian

$$\mathcal{N}(\mathbf{0}, \begin{bmatrix} k_\zeta(\mathbf{x}_t, \mathbf{x}_t) & k_\zeta(\mathbf{x}_t, \mathbf{x}_{t-1}) \\ k_\zeta(\mathbf{x}_{t-1}, \mathbf{x}_t) & k_\zeta(\mathbf{x}_{t-1}, \mathbf{x}_{t-1}) \end{bmatrix})$$

Then $p(f_t|f_{t-1}, \mathbf{x}_t, \mathbf{x}_{t-1}, \zeta)$ is also Gaussian with the mean

$$\frac{k_\zeta(\mathbf{x}_t, \mathbf{x}_{t-1})}{k_\zeta(\mathbf{x}_{t-1}, \mathbf{x}_{t-1})} f_{t-1} \equiv g(\ell) f_{t-1} \tag{4.32}$$

where $g(\ell) = k_\ell(\mathbf{x}_t, \mathbf{x}_{t-1})/k_\ell(\mathbf{x}_{t-1}, \mathbf{x}_{t-1})$, and covariance

$$k_\zeta(\mathbf{x}_t, \mathbf{x}_t) - \frac{k_\zeta^2(\mathbf{x}_t, \mathbf{x}_{t-1})}{k_\zeta(\mathbf{x}_{t-1}, \mathbf{x}_{t-1})} \equiv \sigma^2 q(\ell) \tag{4.33}$$

where $q(\ell) = k_\ell(\mathbf{x}_t, \mathbf{x}_t) - k_\ell^2(\mathbf{x}_t, \mathbf{x}_{t-1})/k_\ell(\mathbf{x}_{t-1}, \mathbf{x}_{t-1})$.

In fact, $p(f_t|f_{t-1}, \mathbf{x}_t, \mathbf{x}_{t-1}, \zeta) = \mathcal{N}(g(\ell) f_{t-1}, \sigma^2 q(\ell))$ represents the dynamical transition model between f_t and f_{t-1} with an additive Gaussian noise $v^f \sim \mathcal{N}(0, \sigma^2 q(\ell))$

$$f_t = g(\ell) f_{t-1} + v^f \tag{4.34}$$

However, as we mentioned before, GP hyper-parameter vector $\zeta = [\sigma^2, \ell]^T$ is often initialized using gradient-based optimization where the learned ζ is locally optimal and does not adjust over time (Rasmussen and Williams, 2006). To capture the time-varying *non-stationarity*, we propose to learn GP hyper-parameters over time (based on the transition (4.34)).

Transition Model (I) We first propose to learn σ^2 (the amplitude parameter of GP covariance function) over time.

One important reason is that σ^2 is encoded in the transition noise v^f of (4.34). By learning σ^2 over time, we can adaptively adjust the transition noise to achieve the non-stationary transition between f_{t-1} and f_t.

Another important reason is that σ^2 is linearly-separable with $q(\ell)$ in the transition noise v^f of (4.34). This fact can achieve an efficient sequential Bayesian inference framework (we will discuss later on) by using a powerful particle learning approach (Carvalho et al., 2010).

Hence, based on (4.34) with the above-mentioned reasons, we propose our transition model (I) with a time-varying σ_t^2 of the transition noise $v_{I,t}^f \sim \mathcal{N}(0, \sigma_t^2 q(\ell))$

$$f_t = g(\ell) f_{t-1} + v_{I,t}^f \tag{4.35}$$

	SSM (I)	SSM (II)
OM	(4.30) with $\sigma_{y,t}^2$	(4.30) with $\sigma_{y,t}^2$
TM	(4.35) with σ_t^2 and ℓ	(4.36-4.38) with σ_t^2, ℓ_t, $\sigma_{y,t}^2$ and τ_t

TABLE 4.4: Our State Space Models. OM : Observation Model. TM : Transition Model.

Transition Model (II) Even though (4.35) allows us to represent non-stationarity by learning a time-varying σ_t^2, its model flexibility may be reduced due to the fact that ℓ of GP, which is associated with $g(\ell)$ and $q(\ell)$, is fixed after initialization. Furthermore, ℓ in this paper is the ARD parameter vector which is used to weight the importance of different input dimensions. In the multi-dimensional-input temporal applications, the fixed ℓ may not capture the time-varying importance of different input dimensions and thus decrease the performance of transition model (I).

To this end, we propose to learn all the time-varying GP hyper-parameters $\zeta_t = [\sigma_t^2, \ell_t]^T$ by augmenting a small dynamic of $\phi_t = \log([\zeta_t, \sigma_{y,t}^2]^T)$ on (4.34)

$$\phi_t = \phi_{t-1} + \mathbf{v}_t^{\phi} \tag{4.36}$$

$$f_t = g(\ell_t) f_{t-1} + v_{II,t}^f \tag{4.37}$$

where $v_{II,t}^f \sim \mathcal{N}(0, \sigma_t^2 q(\ell_t))$, $\mathbf{v}_t^{\phi} \sim \mathcal{N}(\mathbf{0}, diag[\exp(\tau_t)])$ with a parameter vector $\tau_t = \tau$, and $diag[\cdot]$ denotes a diagonal matrix operation. Additionally, since τ is unknown, we propose to design an artificial dynamic (Liu and West, 2001)

$$\tau_t = b\tau_{t-1} + (1-b)\bar{\tau}_{t-1} + v_{t-1}^{\tau} \tag{4.38}$$

where $b = (3\delta - 1)/(2\delta)$, δ is a discount factor (typically around 0.95-0.99), $\bar{\tau}_{t-1}$ is the mean at $t-1$, $v_{t-1}^{\tau} \sim \mathcal{N}(0, (1-b^2)\Sigma_{t-1})$, and Σ_{t-1} is the variance matrix at $t-1$.

Summary Based on (4.34), our SSMs are outlined in Table 4.4 with different transition models. Due to the different model assumptions in our two SSMs, we reorganize the target posterior of GP, i.e.,

$$p(f_{1:T}, \zeta, \sigma_y^2 | D_{1:T}), \quad \zeta = [\sigma^2, \ell]^T$$

to take the time-varying characteristics into account. For SSM (I), our goal of Bayesian inference is

$$p(f_{1:T}, \theta_{1:T}, \ell | D_{1:T}), \quad \theta_t = [\sigma_t^2, \sigma_{y,t}^2]^T$$

For SSM (II), our goal of Bayesian inference is

$$p(f_{1:T}, \phi_{1:T}, \tau_{1:T} | D_{1:T}), \quad \phi_t = \log([\zeta_t, \sigma_{y,t}^2]^T)$$

4.4.3 Particle Based Bayesian Inference

Since the parameters and latent function values are coupled in our SSMs, Bayesian inference for $p(f_{1:T}, \theta_{1:T}, \ell | D_{1:T})$ and $p(f_{1:T}, \phi_{1:T}, \tau_{1:T} | D_{1:T})$ is analytically intractable.

Hence we propose to design sequential Monte Carlo approaches to infer the latent function values and model parameters over time. Specifically, we take advantage of the conditionally linear structures in our two SSMs to preserve the efficiency and accuracy by using particle learning (Carvalho et al., 2010) and Rao-Blackwellized particle filtering (Doucet et al., 2000a).

Particle Learning (PL)

For SSM (I), we factorize the target posterior as follows

$$p(f_{1:T}, \theta_{1:T}, \ell | D_{1:T}) = p(f_{1:T}, \theta_{1:T} | \ell, D_{1:T}) p(\ell | D_{1:T})$$

Same as the standard GP, we learn the static ℓ by using the gradient optimization for $p(\ell | D_{1:T})$. Hence we focus on $p(f_{1:T}, \theta_{1:T} | \ell, D_{1:T})$. Since the data points are sequentially ordered, we mainly work on the posterior of forward filtering $p(f_t, \theta_{1:t} | \ell, D_{1:t})$.

Given the learned ℓ, our SSM (I) becomes a linear structure with the unknown but linearly-separable θ_t. Hence we propose an inference framework based on particle learning (PL) (Carvalho et al., 2010) which allows us to update sufficient statistics of f_t and θ_t to preserve the efficiency of Bayesian inference. To this end, $p(f_t, \theta_{1:t} | \ell, D_{1:t})$ is factorized as follows

$$p(f_t, \theta_{1:t} | \ell, D_{1:t}) = p(\theta_{1:t} | \ell, D_{1:t}, f_t) p(f_t | \ell, D_{1:t})$$

in order to achieve the *resampling-propagate* mechanism of PL.

We first work on **how to update** $p(f_t | \ell, D_{1:t})$. For convenience, we denote state sufficient statistics as $S_t^f = (f_{t|t}, P_{t|t})$ where $f_{t|t}$ and $P_{t|t}$ are the posterior mean and variance of f_t. According to the fact that $p(f_t | \ell, D_{1:t}) = \int p(f_t | S_t^f) p(S_t^f | \ell, D_{1:t}) dS_t^f = \mathbb{E}[p(f_t | S_t^f) | \ell, D_{1:t}]$, thus we are more interested in how to update $p(S_t^f | \ell, D_{1:t})$ since we can directly obtain state estimation after getting the particle approximation of $p(S_t^f | \ell, D_{1:t})$.

Based on Bayesian recursion, $p(S_t^f | \ell, D_{1:t})$ is proportional to the integral

$$\int \int p(S_t^f | S_{t-1}^f, \ell, D_{1:t}, \theta_{t-1}) p(y_t | S_{t-1}^f, \ell, \mathbf{x}_{1:t}, \theta_{t-1}) p(S_{t-1}^f, \theta_{t-1} | \ell, D_{1:t-1}) dS_{t-1}^f d\theta_{t-1}$$

Given the particles of $p(S_{t-1}^f, \theta_{t-1} | \ell, D_{1:t-1})$, we can compute the weights of the particles by $p(y_t | S_{t-1}^f, \ell, \mathbf{x}_{1:t}, \theta_{t-1})$ that is Gaussian

$$\mathcal{N}(g(\ell) f_{t-1|t-1}, g^2(\ell) P_{t-1|t-1} + \sigma_{t-1}^2 q(\ell) + \sigma_{y,t-1}^2) \tag{4.39}$$

According to the normalized weight we resample new particles at $t-1$. After resampling, the new particles at $t-1$ are propagated to $p(S_t^f|S_{t-1}^f, \ell, D_{1:t}, \theta_{t-1})$ that is Kalman filter

$$\begin{aligned}
f_{t|t-1} &= g(\ell) f_{t-1|t-1} \\
P_{t|t-1} &= g^2(\ell) P_{t-1|t-1} + \sigma_{t-1}^2 q(\ell) \\
f_{t|t} &= f_{t|t-1} + (y_t - f_{t|t-1}) P_{t|t-1} / (P_{t|t-1} + \sigma_{y,t-1}^2) \\
P_{t|t} &= P_{t|t-1} - P_{t|t-1}^2 / (P_{t|t-1} + \sigma_{y,t-1}^2)
\end{aligned}$$

We now work on **how to update** $p(\theta_{1:t}|\ell, f_t, D_{1:t})$. After obtaining the particle approximation of f_t, we take advantage of conjugacy to recursively update the parameter sufficient statistics S_t^θ due to the conditionally-linear structure in our SSM (I). Suppose that the parameter distributions at t are $p(\sigma_{y,t}^2|\ell, f_t, D_{1:t}) = \mathcal{IG}(0.5\alpha_t^y, 0.5\beta_t^y)$, $p(\sigma_t^2|\ell, f_t, D_{1:t}) = \mathcal{IG}(0.5\alpha_t^f, 0.5\beta_t^f)$, where $\mathcal{IG}$ represents the inverse Gamma distribution Storvik (2002), then $S_t^\theta = (\alpha_t^y, \beta_t^y, \alpha_t^f, \beta_t^f)$ can be recursively updated as follows [5]

$$\begin{aligned}
\alpha_t^y &= \alpha_{t-1}^y + 1, \quad \beta_t^y = \beta_{t-1}^y + (y_t - f_t)^2 \\
\alpha_t^f &= \alpha_{t-1}^f + 1, \quad \beta_t^f = \beta_{t-1}^f + (f_t - g(\ell) f_{t-1})^2 / q(\ell)
\end{aligned}$$

The computation load of the forward filtering in PL based GP is mainly $O(N_p T + d_x T)$ where N_p is the number of the particles, d_x is the dimension of the input and T is the total number of time steps. $N_p T$ is the main computation of the above-mentioned particle based sampling mechanism, $d_x T$ is the main computation of vector multiplication in (4.32) and (4.33) for all the time steps.

Rao-Blackwellized Particle Filtering

Different from our SSM (I), Equations (4.36-4.38) in our SSM (II) are highly coupled. However, we can see that, given model parameters, the transition between f_t and f_{t-1} in (4.37) is still a linear structure which still allows us to update the sufficient statistics of f_t to keep the efficiency of Bayesian inference. This refers to the idea of Rao-Blackwellized particle filtering (RBPF) (Doucet et al., 2000a; Schön et al., 2005; Wang and Chaib-draa, 2012a). As before, we mainly work on the posterior of forward filtering $p(f_t, \phi_{1:t}, \tau_{1:t}|D_{1:t})$ that can be factorized as :

$$p(f_t, \phi_{1:t}, \tau_{1:t}|D_{1:t}) = p(f_t|\phi_{1:t}, \tau_{1:t}, D_{1:t}) p(\phi_{1:t}, \tau_{1:t}|D_{1:t}).$$

Bayesian inference is then performed in an *importance sampling-resampling* framework.

We first work on **how to update** $p(f_t|\phi_{1:t}, \tau_{1:t}, D_{1:t})$. Given the particles of $\phi_{1:t}$ and $\tau_{1:t}$, the transition for latent function values is linear. Hence, $p(f_t|\phi_{1:t}, \tau_{1:t}, D_{1:t})$ can be inferred using

5. The updates are based on $p(\sigma_y^2|\ell, f_{1:t}, D_{1:t}) \propto p(y_t|f_t, \sigma_y^2) p(\sigma_y^2|\ell, f_{1:t-1}, D_{1:t-1})$ and $p(\sigma^2|\ell, f_{1:t}, D_{1:t}) \propto p(f_t|f_{t-1}, \sigma^2, \ell) p(\sigma^2|\ell, f_{1:t-1}, D_{1:t-1})$

Kalman filter

$$
\begin{aligned}
f_{t|t-1} &= g(\ell_t) f_{t-1|t-1} \\
P_{t|t-1} &= g^2(\ell_t) P_{t-1|t-1} + \sigma_t^2 q(\ell_t) \\
f_{t|t} &= f_{t|t-1} + (y_t - f_{t|t-1}) P_{t|t-1} / (P_{t|t-1} + \sigma_{y,t}^2) \\
P_{t|t} &= P_{t|t-1} - P_{t|t-1}^2 / (P_{t|t-1} + \sigma_{y,t}^2)
\end{aligned}
$$

Then we work on **how to update** $p(\phi_{1:t}, \tau_{1:t}|D_{1:t})$. This posterior $p(\phi_{1:t}, \tau_{1:t}|D_{1:t})$ is proportional to

$$
p(y_t|y_{1:t-1}, \phi_{1:t}, \tau_{1:t}, \mathbf{x}_{1:t}) p(\phi_t|\phi_{t-1}, \tau_t) p(\tau_t|\tau_{t-1}) p(\phi_{1:t-1}, \tau_{1:t-1}|D_{1:t-1}).
$$

Given the particles of $\phi_{1:t-1}$ and $\tau_{1:t-1}$, we can first sample particles for τ_t using $p(\tau_t|\tau_{t-1})$ of Equation (4.38) and then sample particles for ϕ_t using $p(\phi_t|\phi_{t-1}, \tau_t)$ of Equation (4.36). Then the particles of sufficient statistics of the latent function values, the particles of $\phi_{1:t}$ and the particles of $\tau_{1:t}$ are weighted by

$$
p(y_t|y_{1:t-1}, \phi_{1:t}, \tau_{1:t}, \mathbf{x}_{1:t}) = \mathcal{N}(f_{t|t-1}, P_{t|t-1} + \sigma_{y,t}^2). \tag{4.40}
$$

Finally, according to the normalized weights, all the particles are resampled for the next step.

The computation load of the forward filtering in RBPF based GP is mainly $O(N_p T + N_p d_x T)$ where N_p is the number of the particles, d_x is the dimension of the input and T is the total number of time steps. Note that the computational difference between RBPF based GP and PL based GP is the vector multiplication ($N_p d_x T$ vs. $d_x T$) in (4.32) and (4.33). In the PL based GP, ℓ is fixed after initialization. Hence, this multiplication is only implemented once at each time step. On the contrary, N_p particles of ℓ_t is sampled at each step in the RBPF based GP. Hence, for each time step, the multiplication should be operated for all the particles.

Backward Smoothing

So far we have established our PL based GP (PL-GP) and RBPF based GP (RBPF-GP) to obtain $p(f_t, \theta_{1:t}|\ell, D_{1:t})$ for SSM (I) and $p(f_t, \phi_{1:t}, \tau_{1:t}|D_{1:t})$ for SSM (II). Our target posteriors $p(f_{1:T}, \theta_{1:T}|\ell, D_{1:T})$ and $p(f_{1:T}, \phi_{1:T}, \tau_{1:T}|D_{1:T})$ refer to backward smoothing in which the posterior is evaluated given the entire data set. Fortunately, backward smoothing in (Godsill et al., 2004) can be straightforwardly implemented to our PL, RBPF based GP approaches with a backward sequential pass. Note that the computation complexity of the particle-based sampling mechanism in backward smoothing for both PL based GP and RBPF based GP will be $N_p^2 T$ instead of $N_p T$ in forward filtering (Godsill et al., 2004).

4.4.4 Experiments

Data Sets

To show that our two particle based GP approaches perform well in time-varying applications, we evaluated them on four time series : synthetic data, motor [6], heart rate [7], S&P financial data [8] where the synthetic data, motor and heart rate data sets are one-dimensional input cases ; S&P data set is a multi-dimensional input case. Specifically, the synthetic set consists of 1000 time/observation (input/output) pairs which are generated from $y_t = -30 + \mathcal{N}(0,1)$ $(t = 0.01 : 0.01 : 2)$, $y_t = 50\sin(0.5\pi t) + \mathcal{N}(0,9)$ $(t = 2.01 : 0.01 : 5)$, $y_t = 20\cos(\pi t + 0.5\pi) + \mathcal{N}(0,100)$ $(t = 5.01 : 0.01 : 10)$. The motor consists of 94 time/acceleration (input/output) pairs. Heart rate consists of 300 time/heart rate (input/output) pairs. S&P consists of 500 points which are weekly recorded from 2000-01-03 to 2009-08-03. The 4-D inputs are respectively time, S&P 100 index, S&P 400 Midcap index, S&P 500 index ; the output is volatility S&P 500.

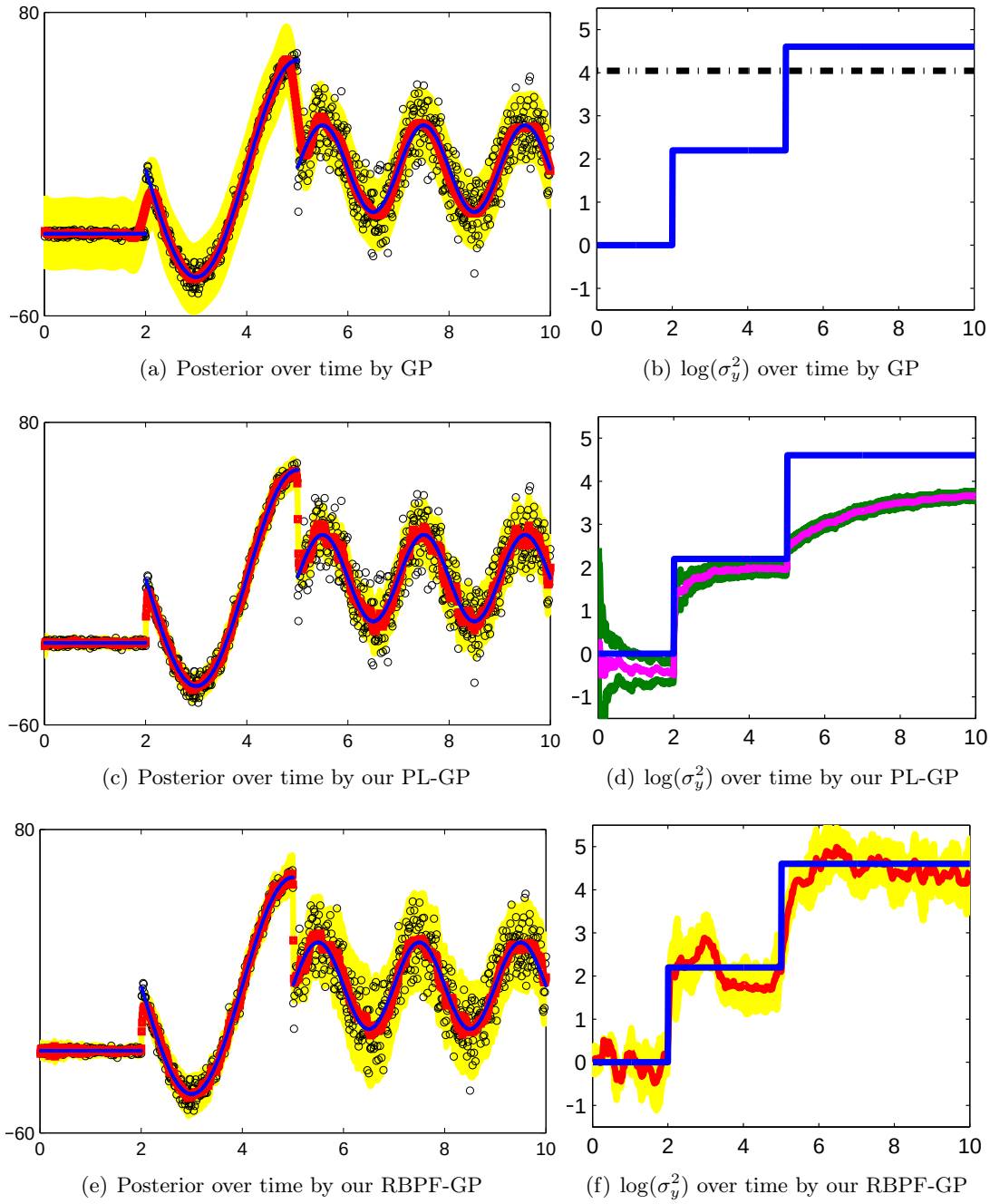
Posterior Evaluation

Our first experiment is to visualize the posterior surface of our methods to evaluate whether the time-varying data properties can be correctly modeled. Hence, we compare the posterior surface of our PL-GP (with backward smoothing), RBPF-GP (with backward smoothing) to the batch GP for all the data sets. For each data set, we subtract the mean of the data points, and for all the methods we use the same initial ζ that is learned by using gradient based optimization with all the data points. The number of particles in both PL-GP and RBPF-GP is 200.

The synthetic data exhibits both time-varying non-stationarity and heteroscedasticity. In Figure 4.16, it is shown that the three-noise-level heteroscedasticity in this synthetic data is mistakenly interpreted by GP due to the homoscedastic noise assumption in GP. Moreover, the discontinuity around $t = 2$ and $t = 5$, which is an important form of time-varying non-stationarity, is mistakenly interpreted by the GP due to its stationary assumption. On the contrary, both non-stationarity and heteroscedasticity are suitably interpreted by our PL-GP and RBPF-GP.

Since the ground truth of the latent function values are known in the synthetic data, we evaluated whether the latent function values can be successfully inferred by our method. We compared our method to several benchmark GP variants that are often used to capture the time-varying non-stationarity and heteroscedasticity. NNGP is a GP with non-stationary neural network covariance function (Rasmussen and Williams, 2006). SPGP is sparse pseudo-input GP (Snelson and Ghahramani, 2005), which is originally used to reduce computation

6. Motor data is available at http ://www.stat.cmu.edu/∼ larry/all-of-statistics/=data/motor.dat
7. Heart rate data is available at http ://www-psych.stanford.edu/∼andreas/Time-Series/SantaFe.html
8. S&P data is available at http ://finance.yahoo.com/stock-center/



(a) Posterior over time by GP

(b) $\log(\sigma_y^2)$ over time by GP

(c) Posterior over time by our PL-GP

(d) $\log(\sigma_y^2)$ over time by our PL-GP

(e) Posterior over time by our RBPF-GP

(f) $\log(\sigma_y^2)$ over time by our RBPF-GP

FIGURE 4.16: Posterior Evaluation (Synthetic Data). In the 1st column, we show the posterior surface with 95% confidence interval (red line with yellow interval) given observations (black circles) and ground truth function (blue line). In the 2nd column, we show the posterior estimation of $\log(\sigma_y^2)$ over time by GP (black doted line), our PL-GP (pink line with green lines), our RBPF-GP (red line with yellow interval) given ground truth $\log(\sigma_y^2)$ (blue line).

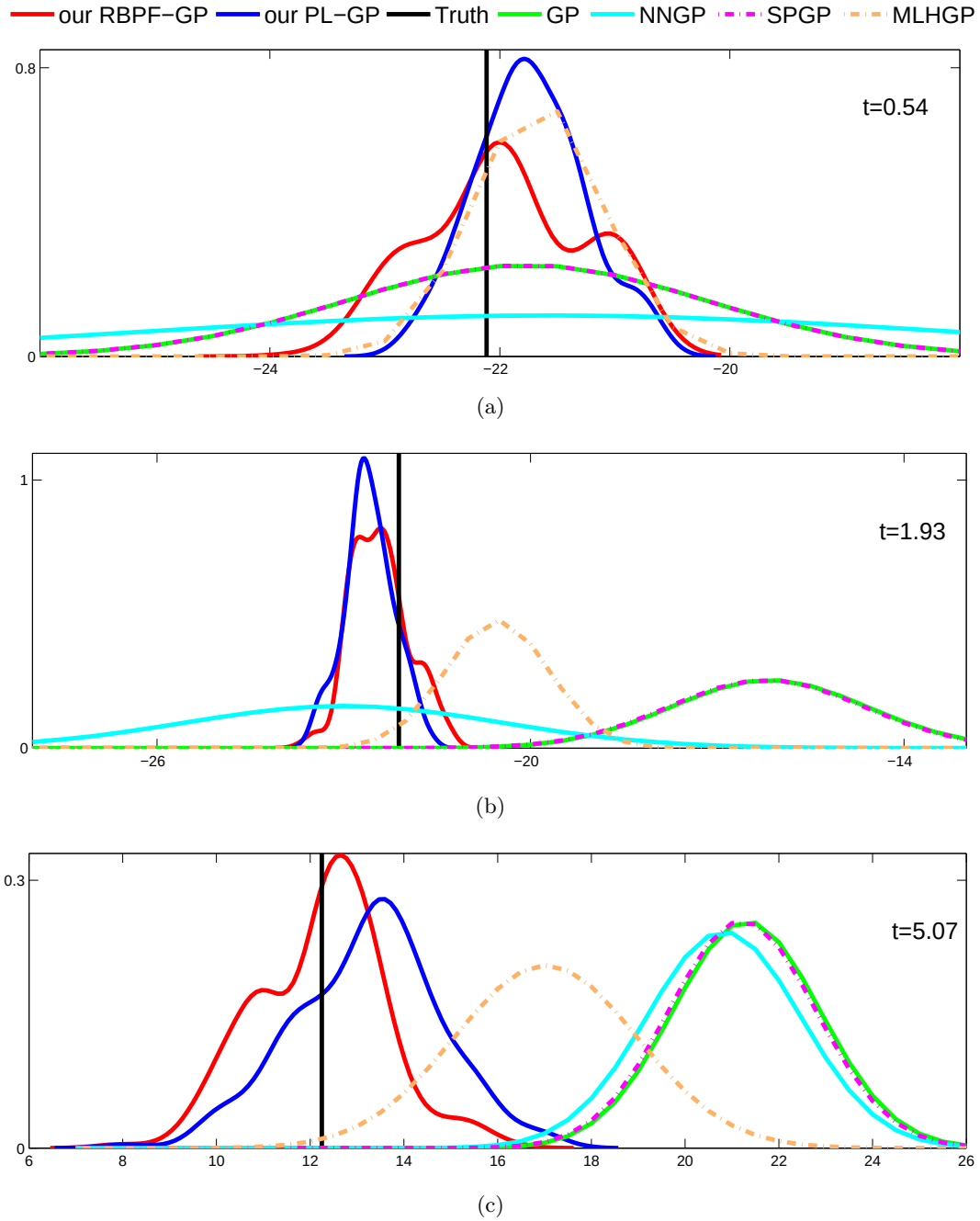


FIGURE 4.17: Backward smoothing posterior comparison of f_t at t =0.54 (top), 1.93 (middle), 5.07 (bottom).

Table 4.5: Accuracy & Efficiency of Posterior Evaluation for Synthetic Data.

	GP	NNGP	SPGP	MLHGP	our PL-GP	our RBPF-GP
MSE	9.43	7.12	9.40	7.48	**6.10**	**5.80**
RunningTime(s)	149.3	79.9	49.4	353.9	**27.6**	56.7

burden, but can achieve the heteroscedasticity. MLHGP is the most likely heteroscedastic GP (Kersting et al., 2007).

In Figure 4.17, we showed the posteriors of f_t at $t = 0.54$, 1.93, 5.07. Since the time period around $t = 0.54$ mainly reflects the heteroscedasticity, we can see that GP, NNGP and SPGP have flat-shaped posteriors (reflect high uncertainty) in order to keep the consistency of the observation noise. MLHGP, our PL-GP and RBPF-GP can deal with heteroscedasticity, hence the posterior for these approaches have peak-shaped posteriors (reflect low uncertainty). Furthermore, the time period around $t = 1.93$ and $t = 5.07$ mainly reflect the non-stationarity, we can see that the latent function values are mistakenly inferred by GP, SPGP. MLHGP and NNGP attempt to correct the non-stationarity, however the performance is still limited. On the contrary, our PL-GP and RBPF-GP successfully capture the time-varying properties with low uncertainty.

Moreover, we used mean squared error (MSE) and running time to compare our methods to other GP approaches with regard to accuracy and efficiency of posterior evaluation. It is shown in Table 4.5 that our PL-GP and RBPF-GP outperform other GP methods with higher accuracy and efficiency. Furthermore, as expected, the accuracy of our RBPF-GP is higher than our PL-GP but the computation efficiency is lower than our PL-GP. The reason is that RBPF-GP learns all the model parameters $\zeta_t = [\sigma_t^2, \ell_t]$ and $\sigma_{y,t}^2$ over time, but PL-GP learns σ_t^2 and $\sigma_{y,t}^2$ over time with the initialized ℓ.

The resulting posterior evaluation for the other three real-world data sets are respectively shown in Figure 4.18 - 4.20. The motor data (Figure 4.18) mainly exhibits the time-varying heteroscedasticity that consists of a flat low noise region, a curve region and flat high noise region (Rasmussen and Ghahramani, 2001). To keep the consistency, GP fails to capture the low-noise region. On the contrary, our PL-GP and RBPF-GP successfully explain all three noise regions. More interestingly, our RBPF-GP correctly represents the decreasing noise in the flat high noise region (Kersting et al., 2007).

The heart rate data mainly exhibits time-varying non-stationarity that is reflected by the sudden signal shifts. We showed the posterior of the sudden shift region (from 150-th to 230-th step) in Figure 4.19. To capture this sudden change (around 180-th), GP has to increase noise level and thus introduces unnecessary uncertainty in other regions. On the contrary, our

PL-GP and RBPF-GP correctly capture the sudden shift with compact confidence intervals.

S&P data is time-varying heteroscedastic and non-stationary. Moreover, the input here is a multi-dimensional vector which is not only related to the time. All these facts make Bayesian inference quite challenging. We showed the output posterior along with the time axis (from 200-th to 350-th step) in Figure 4.20. GP mistakenly interprets this region with a large noise level. On the contrary, our PL-GP reduces the effect of non-stationarity and heteroscedasticity by learning σ_t^2 and $\sigma_{y,t}^2$ over time. Furthermore, as expected, our RBPF-GP obtains the best posterior in this multi-dimensional input case due to the fact that our RBPF-GP learns all the time-varying model parameters including ℓ_t which is associated with the importance of different input dimensions in ARD covariance function.

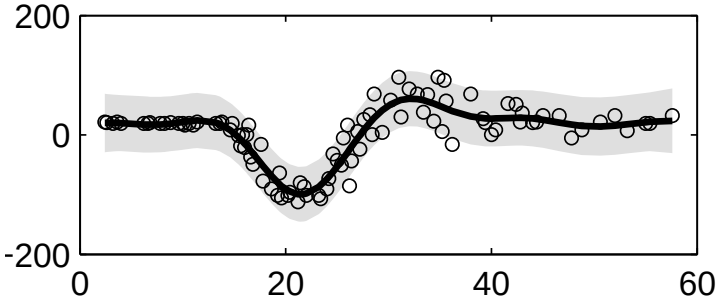
Prediction Comparison

Since our PL-GP and RBPF-GP are online GP approaches, in our second experiment we quantitatively evaluated the accuracy of our PL-GP and RBPF-GP for online prediction (based on Eq.4.39 and 4.40). The evaluation criterion is mean negative log probability (MNLP) that penalizes both uncertainty and inconsistency (Deisenroth et al., 2009).

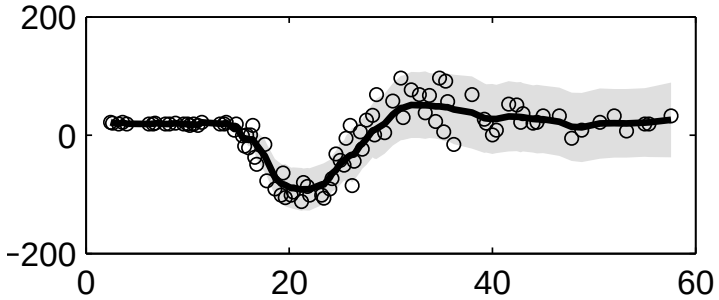
We compared our approaches with several online GP approaches : autoregressive GP (ARGP) trained on pairs of (y_t, y_{t+1}) ; high-order ARGP with neural network kernel (NHARGP) where the order is 10 for synthetic and motor, 20 for heart and SP data ; Kalman filtering-Temporal GP (KFTGP) (Hartikainen and Särkkä, 2010) ; Local GP (Nguyen-Tuong et al., 2008) ; Sparse online GP (SOGP) (Csató and Opper, 2002) ; Kernel recursive least-squares tracker (KRLST) (Vaerenbergh et al., 2012) ; GP particle filter (GP-PF) (Ko and Fox, 2008) ; Marginalized particle GP (MPGP) (Wang and Chaib-draa, 2012a). For LGP the prediction at the current step was based on all the local experts generated until this step (due to online construction of GP experts in (Nguyen-Tuong et al., 2008)) ; for SOGP and KRLST, the prediction for the current step was based on all the data until this step without sparsification. The reason is we tended to use the best results of these online GPs. Additionally, for GP-PF and MPGP, the latent state was f_t. For all the methods, GP hyper-parameters was the same and initialized using the first 300/50/200/200 data points for synthetic/motor/heart/SP. The evaluation of online prediction was based on the rest data points for all the data sets. Finally, for all the data sets and all the methods, we ran 20 times to obtain the average MNLP.

As our methods are particle based approaches, we firstly compared our PL-GP and RBPF-GP to other particle based methods, GP-PF and MPGP, by evaluating MNLP as a function of the number of the particles. It is shown in Figure 4.4.4 and Table 4.6 that our methods outperform GP-PF and MPGP by learning the time-varying model parameters.

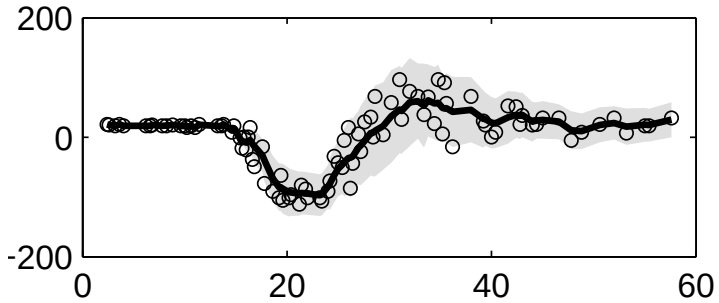
Moreover, the comparison with all the online GP methods was shown in Table 4.6. We can see that, for all the data sets, our PL-GP and RBPF-GP outperform all the other state-of-the-



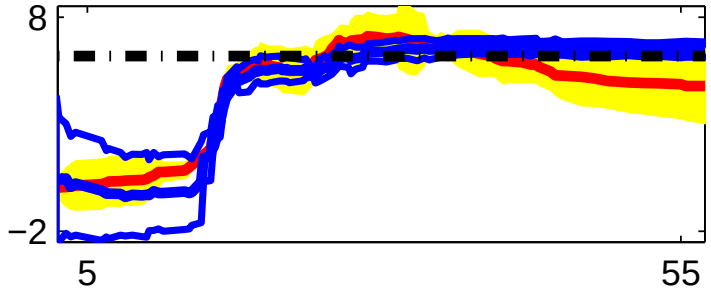
(a) Posterior over time by GP



(b) Posterior over time by our PL-GP

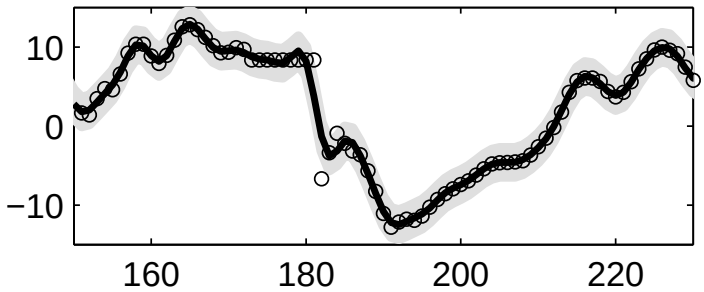


(c) Posterior over time by our RBPF-GP

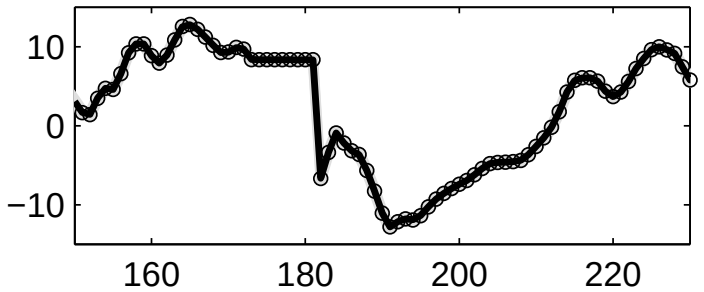


(d) $\log(\sigma_y^2)$ over time

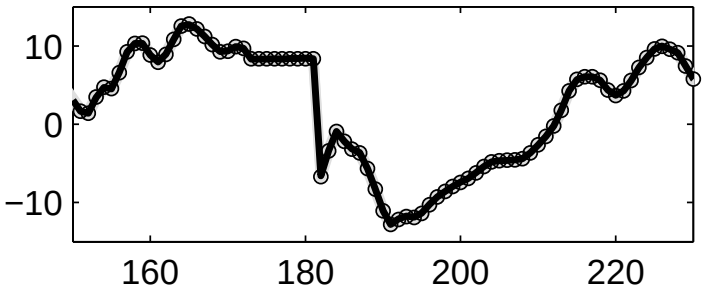
FIGURE 4.18: Posterior Evaluation (Motor Data). The first three rows are the posterior surfaces over time of GP, our PL-GP, our RBPF-GP where we show the posterior surface with 95% confidence interval (black line with grey interval) given observations (circles). The last row is the posterior estimation of $\log(\sigma_y^2)$ over time where the results of GP is the black dashed line. The posterior with 95% confidence interval for our PL-GP / our RBPF-GP are respectively blue lines / red lines with yellow intervals.



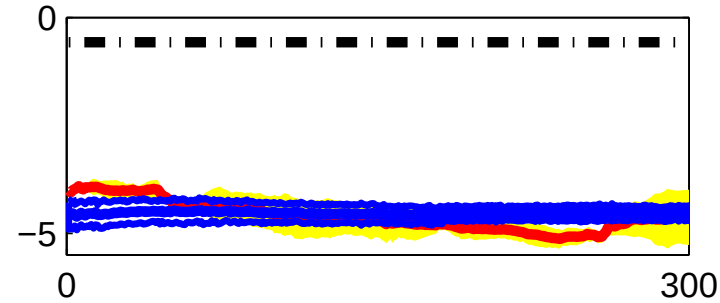
(a) Posterior over time by GP



(b) Posterior over time by our PL-GP

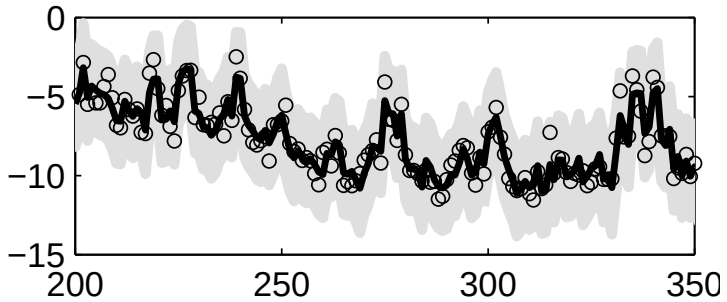


(c) Posterior over time by our RBPF-GP

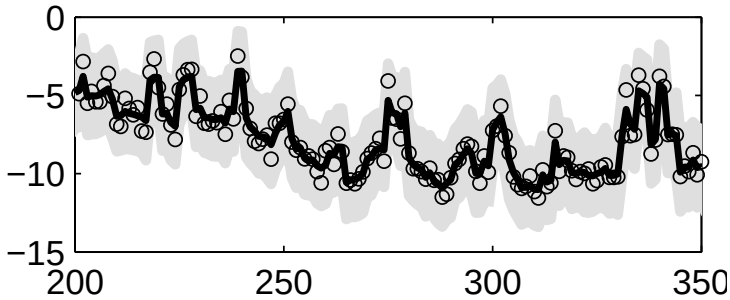


(d) $\log(\sigma_y^2)$ over time

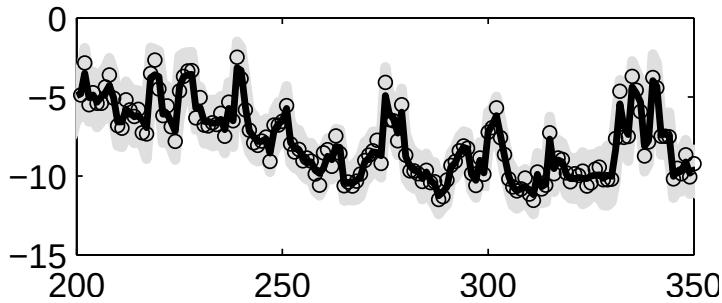
FIGURE 4.19: Posterior Evaluation (Heart Data). The first three rows are the posterior surfaces over time of GP, our PL-GP, our RBPF-GP where we show the posterior surface with 95% confidence interval (black line with grey interval) given observations (circles). The last row is the posterior estimation of $\log(\sigma_y^2)$ over time where the results of GP is the black dashed line. The posterior with 95% confidence interval for our PL-GP / our RBPF-GP are respectively blue lines / red lines with yellow intervals.



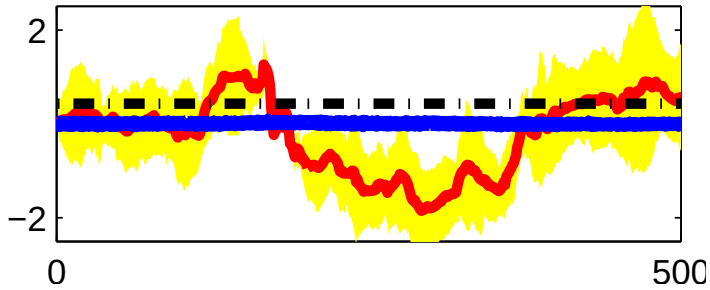
(a) Posterior over time by GP



(b) Posterior over time by our PL-GP



(c) Posterior over time by our RBPF-GP



(d) $\log(\sigma_y^2)$ over time

FIGURE 4.20: Posterior Evaluation (SP Data). The first three rows are the posterior surfaces over time of GP, our PL-GP, our RBPF-GP where we show the posterior surface with 95% confidence interval (black line with grey interval) given observations (circles). The last row is the posterior estimation of $\log(\sigma_y^2)$ over time where the results of GP is the black dashed line. The posterior with 95% confidence interval for our PL-GP / our RBPF-GP are respectively blue lines / red lines with yellow intervals.

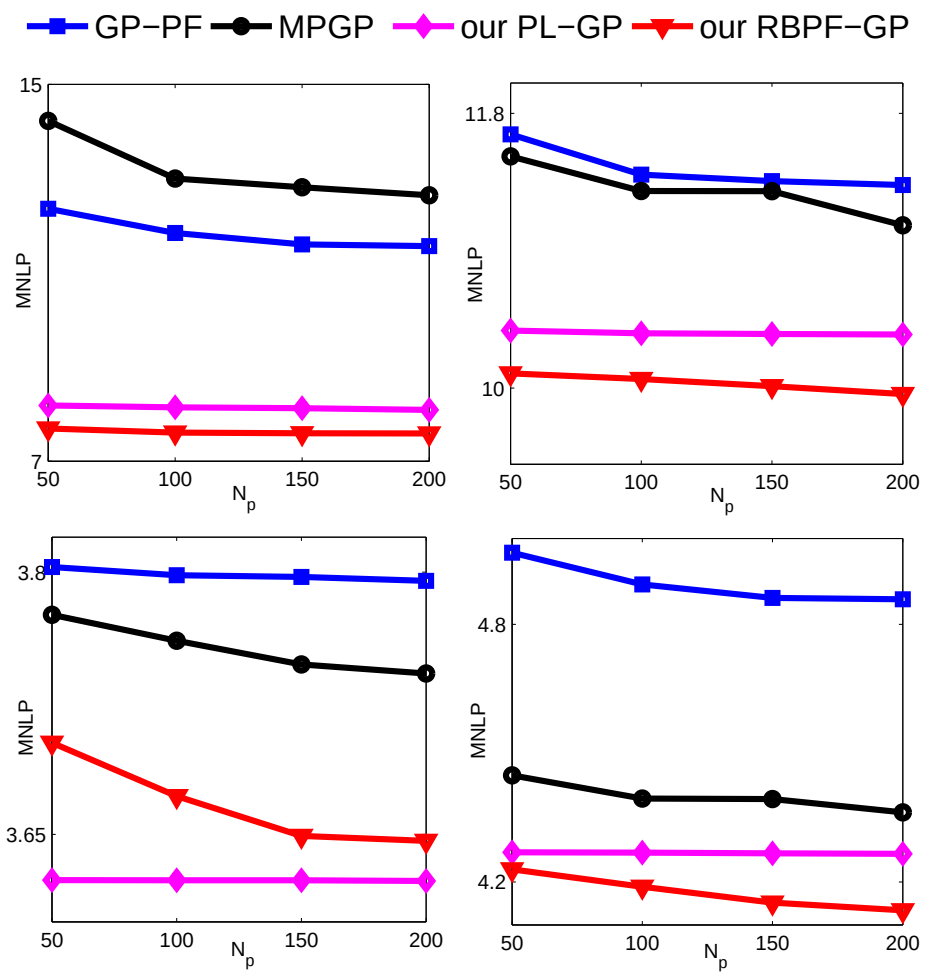


FIGURE 4.21: MNLP as a function of the number of particles. The plots are respectively the results for synthetic (1st row, left), motor(1st row, right), heart (2nd row, left) and SP data (2nd row, right).

art online GPs since the strong heteroscedasticity and non-stationarity in these time-varying applications reduce the prediction accuracy of most of the online GPs. It is worth mentioning that, to some degree, NHARGP, LGP, KRLST and MPGP can deal with time-varying data properties. However, compared to our PL-GP and RBPF-GP, their accuracy is still limited.

4.4.5 Summary

In Wang and Chaib-draa (2015a), we have proposed two particle based GPs for time-varying applications. By designing two novel SSMs based on GP, we used PL and RBPF to capture the time-varying non-stationarity and heteroscedasticity in a recursive Bayesian framework. The experiments showed that our approaches outperform several benchmark GP variants. Furthermore, our PL-GP is more computationally efficient than RBPF-GP, but our RBPF-GP is more accurate than PL-GP, especially in multi-dimensional input cases.

Table 4.6: Accuracy (MNLP) Evaluation for Online Prediction.

Method	Syn	Motor	Heart	SP	Method	Syn	Motor	Heart	SP
ARGP	14.33	11.61	3.757	6.61	GP-PF$_{200}$	11.56	11.33	3.795	4.85
NHARGP	9.14	10.43	**3.64**	5.38	MPGP$_{50}$	14.22	11.52	3.776	4.45
KFTGP	13.53	10.95	3.748	5.54	MPGP$_{200}$	12.64	11.07	3.742	4.36
LGP	10.27	10.63	3.720	4.54	PL-GP$_{50}$	**8.18**	**10.37**	**3.624**	**4.27**
SOGP	13.27	10.94	3.751	4.80	PL-GP$_{200}$	**8.08**	**10.35**	**3.623**	**4.26**
KRLST	**8.12**	10.38	3.730	4.71	RBPF-GP$_{50}$	**7.69**	**10.10**	3.702	**4.23**
GP-PF$_{50}$	12.35	11.66	3.803	4.97	RBPF-GP$_{200}$	**7.58**	**9.96**	**3.646**	**4.13**

4.5 Conclusion

In this chapter, we mainly worked on the direction - *Bayesian Estimation for Gaussian Processes* where we proposed three contributions for GPs to address real-world challenges by using several benchmark techniques in sequential Bayesian estimation. Specifically, the marginalized particle GP (Wang and Chaib-draa, 2012a) provided us with a computationally efficient GP method where hyperparameter learning and latent function estimation are accurately performed in a marginalized particle filter. Then, the KNN based Kalman filter GP (Wang and Chaib-draa, 2013) was designed for nonstationary phenomena. Finally, two sequential Monte Carlo based GP approaches (Wang and Chaib-draa, 2015a) were proposed for time-varying applications so that time-varying nonstationarity and heteroscedasticity of the temporal-ordered data points can be recursively interpreted in an online fashion.

Due to the fact that the influence between *Bayesian Estimation* and *Gaussian Processes* is bidirectional, in the next chapter we will take advantage of *Dynamical Interaction Between Gaussian Processes and Bayesian Estimation* to address several difficulties which potentially exist in both *Bayesian Estimation* and *Gaussian Processes*.

Chapitre 5

Dynamical Interaction Between Gaussian Processes and Bayesian Estimation

In many real-world applications, model learning and state estimation are strongly connected (Doucet et al., 2000b; Liu and West, 2001; Wang et al., 2008; Deisenroth et al., 2009; Ko and Fox, 2009). On one hand, model learning requires training data points which are often the estimated latent states. On the other hand, state estimation are based on the learned model. This fact tells us that, if we plan to achieve desirable performance from either the learning or estimation perspective, we have to address the challenges from both perspectives. This is our motivation to perform *Dynamical Interaction Between Gaussian Processes and Bayesian Estimation*. In the following we propose two contributions which aim to interactively address the difficulties in both GP learning and sequential Bayesian estimation

1. The first contribution is an online GP particle filter framework (Wang et al., 2014) where a Gaussian process dynamical model is refined online during tracking. It is a novel approach for both modeling and tracking because of the interaction between the GP and particle filter. Moreover, the resulting online GP particle filters can capture the multi-modality in dynamical systems by using a GP mixture representation for the state space model, and successfully track different types of human motions.

2. The second contribution is a heteroscedastic deep GP (HDGP) framework (Wang and Chaib-draa, 2015b) in which we address a heteroscedastic multi-output GP regression task by sharing a deep GP structure between signal and noise in the observation layer. By using *Dynamical Interaction Between Gaussian Processes and Bayesian Estimation*, we then propose a sequential Monte Carlo inspired inference mechanism to infer the latent states and update our HDGP in an efficient and accurate recursive Bayesian fashion.

5.1 Bayesian Filtering with Online Gaussian Process Latent Variable Models

In Wang et al. (2014), we present a novel online Gaussian process (GP) particle filter framework where the prediction and observation models in the state space model are learned in an online fashion. It is a non-parametric approach to Bayesian filtering, which is able to handle multi-modal distributions over both models by employing a mixture representation with GP based components. Moreover, to cope with the increasing complexity of the estimation process, we explore two computationally efficient GP variants, sparse online GP (Csató and Opper, 2002) and local GP (Urtasun and Darrell, 2008), which help to manage computation requirements for each mixture component. Our experiments demonstrate that our approach can track human motion much more accurately than existing approaches that learn the prediction and observation models offline and do not update these models with the incoming data stream.

5.1.1 Background and Previous Work

In this contribution, we are interested in *modeling* and *tracking* human motion which are two challenging problems involving high dimensional data.

In the context of modeling, dimensionality reduction techniques are widely employed to avoid the curse of dimensionality. Linear approaches such as principle component analysis (PCA) are popular as they are simple to use. However, they often fail to capture complex dependencies due to their assumption of linearity. Non-linear dimensionality reduction techniques that attempt to preserve the local structure of the manifold (*e.g.*, Isomap (Tenenbaum et al., 2000; Jenkins and Matarić, 2004), LLE (Roweis and Saul, 2000; Lee and Elgammal, 2010)) can capture more complex dependencies, but often suffer when the manifold assumptions are violated, *e.g.*, in the presence of noise.

Probabilistic latent variable models have the advantage of being able to take the uncertainties into account when learning the latent representations. Perhaps the most successful model for human motion is the Gaussian process latent variable model (GPLVM) (Lawrence, 2005), where the non-linear mapping between the latent space and the high dimensional space is modeled with a GP. This provides powerful prior models, which have been employed for character animation (Wang et al., 2008; Urtasun et al., 2008; Levine et al., 2012) and human body tracking (Urtasun et al., 2005; Moon and Pavlovic, 2006; Urtasun et al., 2006).

In the context of tracking, one is interested in Bayesian state estimation of a dynamic system. The most commonly used technique is Bayesian filtering, which recursively estimates the posterior probability of the states of the system. The two key components in the filter are the prediction model, which describes the temporal evolution of the process, as well as the observation model which links the state and the observation. A parametric form is typically

employed for both models.

Ko and Fox (2008) introduced the GP-BayesFilter, which defines the prediction and observation models by non-parametric GPs in order to deal with the case that accurate parametric models are difficult to obtain. Its main limitation, however, resides in the fact that it requires the ground truth of the latent states (as GPs are supervised), which are typically not available. Two extensions were introduced to learn the hidden states of the training set via a non-linear latent variable model (Ko and Fox, 2009) or a sparse pseudo-input GP regression (Turner et al., 2010). But these approaches cannot exploit the incoming stream of data available in the online setting as the latent space is learned offline. Furthermore, only unimodal prediction and observation models can be captured due to the fact that the models learned by GPs are nonlinear but Gaussian.

In this contribution (Wang et al., 2014), we extend the previous non-parametric GP-BayesFilter by performing *Dynamical Interaction Between Gaussian Process and Bayesian Estimation* to learn the latent space in an online fashion as well as to handle multi-modal distributions for both the prediction and observation models. We demonstrate the effectiveness of our approach on a wide variety of motions, and show that our approach performs better than existing algorithms.

5.1.2 Gaussian Process Dynamical Model

The Gaussian Process Latent Variable Model (GPLVM) is a probabilistic dimensionality reduction technique, which places a GP prior on the observation model (Lawrence, 2005). Wang et al. (2008) proposed the Gaussian Process Dynamical Model (GPDM), which enriches the GPLVM to capture temporal structure by incorporating a GP prior over the dynamics in the latent space. In this case, the state space model is

$$\begin{aligned} \mathbf{x}_t &= f_{\mathbf{x}}(\mathbf{x}_{t-1}) + \eta_{\mathbf{x}}, \\ \mathbf{y}_t &= f_{\mathbf{y}}(\mathbf{x}_t) + \eta_{\mathbf{y}}, \end{aligned}$$

where $\mathbf{y} \in \mathbb{R}^{D_{\mathbf{y}}}$ represents the observation and $\mathbf{x} \in \mathbb{R}^{D_{\mathbf{x}}}$ the latent state, with the dimensionality $D_{\mathbf{y}} \gg D_{\mathbf{x}}$. The noise processes are assumed to be Gaussian $\eta_{\mathbf{x}} \sim \mathcal{N}(0, \sigma_{\mathbf{x}}^2 I)$ and $\eta_{\mathbf{y}} \sim \mathcal{N}(0, \sigma_{\mathbf{y}}^2 I)$. The nonlinear functions $f_{\mathbf{x}}^i$ and $f_{\mathbf{y}}^i$ have GP priors, *i.e.*, $f_{\mathbf{x}}^i \sim \mathcal{GP}(0, k_x(\mathbf{x}, \mathbf{x}'))$ and $f_{\mathbf{y}}^i \sim \mathcal{GP}(0, k_y(\mathbf{x}, \mathbf{x}'))$ where $k_x(\cdot,\cdot)$ and $k_y(\cdot,\cdot)$ are the kernel functions. For simplicity, we denote the hyperparameters of the kernel functions by θ.

Let $\mathbf{x}_{1:T_0} = (\mathbf{x}_1, \cdots, \mathbf{x}_{T_0})$ be the latent space coordinates from time $t = 1$ to time $t = T_0$. GPDM is typically learned by minimizing the negative log posterior $-\log(p(\mathbf{x}_{1:T_0}, \theta | \mathbf{y}_{1:T_0}))$ with respect to $\mathbf{x}_{1:T_0}$, and θ (Wang et al., 2008). After $\mathbf{x}_{1:T_0}$ and θ are obtained, a standard GP prediction is used to construct the model $p(\mathbf{x}_t | \mathbf{x}_{t-1}, \theta, \mathcal{X}_{T_0})$ and $p(\mathbf{y}_t | \mathbf{x}_t, \theta, \mathcal{Y}_{T_0})$ with data sets $\mathcal{X}_{T_0} = \{(\mathbf{x}_{k-1}, \mathbf{x}_k)\}_{k=2}^{T_0}$ and $\mathcal{Y}_{T_0} = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^{T_0}$. Tracking ($t > T_0$) is then performed

assuming the model is fixed and can be done using particle filtering (Ko and Fox, 2009). The major drawback of this approach is that GPDM is learned offline and thus it is not able to adapt to new incoming observations during tracking. As shown in our experimental evaluation, this results in poor performance when the training set is small.

5.1.3 Online Gaussian Process Particle Filter

To take the incoming data stream into account in order to improve the performance of both learning and filtering, we propose an *Online GP Particle Filter* framework to refine the state space model during tracking, *i.e.*, the prediction $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ and observation $p(\mathbf{y}_t|\mathbf{x}_t)$ models are updated online in the particle filtering framework. Furthermore, to deal with multi-modality and the significant amount of uncertainty that can be present, we propose to represent the prediction and observation models by a mixture model. For each mixture component, we will investigate two different GP variants.

The whole framework is summarized in Algorithm 8. Concretely, let the prediction and observation models at $t-1$ be

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{x}_{t-1}, \Theta_{t-1,M}) &= \frac{1}{R_M}\sum_{i=1}^{R_M} p(\mathbf{x}_t|\mathbf{x}_{t-1}, \Theta^i_{t-1,M}), \\ p(\mathbf{y}_t|\mathbf{x}_t, \Theta_{t-1,O}) &= \frac{1}{R_O}\sum_{i=1}^{R_O} p(\mathbf{y}_t|\mathbf{x}_t, \Theta^i_{t-1,O}), \end{aligned}$$

where $\Theta^i_{t-1,M}$ and $\Theta^i_{t-1,O}$ denote the parameters of the i-th component, $\Theta_{t-1,M} = \{\Theta^i_{t-1,M}\}_{i=1}^{R_M}$ and $\Theta_{t-1,O} = \{\Theta^i_{t-1,O}\}_{i=1}^{R_O}$ are the parameters of all components. At the t-th time step, we run a standard particle filter to obtain a number of weighted particles (Algorithm 8, Line 4-8). The latent space representation at time t can be obtained by resampling the weighted particles (Algorithm 8, Line 9). Then, for simplicity, we assign each particle to the most likely mixture component of $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \Theta_{t-1,M})$ and $p(\mathbf{y}_t|\mathbf{x}_t, \Theta_{t-1,O})$ to capture the multi-modality of the prediction and observation models (Algorithm 8, Line 10-13). Finally, we infer the latent state (the mean of the assigned particles (Doucet et al., 2000b; Cappé et al., 2007)) and use this estimated state to update the corresponding components parameters, $\Theta^i_{t,M}$ for the prediction (or motion) model and $\Theta^i_{t,O}$ for the observation model. (Algorithm 8, Line 14-24).

What remains is to specify how the parameters of individual components are represented and updated (lines 18 and 23 in Algorithm 8). As noted above, we aim to use a GP model for each mixture component. However, a standard GP implementation would require $O(t^3)$ operations and $O(t^2)$ memory. As t grows linearly over time, the particle filter will quickly become too computationally and memory intensive. Thus a primary challenge is how to efficiently update the GP mixture components in the prediction and observation models.

In order to efficiently update $\Theta^i_{t,M}$ and $\Theta^i_{t,O}$ in an online manner, we consider two fast GP-

Algorithm 8 Online GP-Particle Filter.

Initialize model parameters Θ based on $\mathbf{y}_{1:T_0}$
Initialize particle set $\mathbf{x}_{T_0}^{(1:N_P)}$ based on $\mathbf{y}_{1:T_0}$
for $t = T_0 + 1$ **to** T **do**
  for $i = 1$ **to** N_p **do**
    $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \Theta_{t-1,M})$
    $\hat{w}_t^{(i)} = p(\mathbf{y}_t|\mathbf{x}_t^{(i)}, \Theta_{t-1,O})$
  end for
  Normalize weights $w_t^{(i)} = \hat{w}_t^{(i)}/(\sum_{i=1}^{N_p} \hat{w}_t^{(i)})$
  Resample particle set with probabilities $w_t^{(1:N_p)}$
  for $i = 1$ **to** N_p **do**
    $\eta_M^i = \arg\max_j p(\mathbf{x}_t^{(i)}|\mathbf{x}_{t-1}^{(i)}, \Theta_{t-1,M}^j)$
    $\eta_O^i = \arg\max_j p(\mathbf{y}_t|\mathbf{x}_t^{(i)}, \Theta_{t-1,O}^j)$
  end for
  for $j = 1$ **to** R_M **do**
    $n_{t-1}^j = \sum_{i=1}^{N_p} \delta(\eta_M^i = j)$
    $\bar{\mathbf{x}}_{t-1}^j = \frac{1}{n_{t-1}^j} \sum_{i=1}^{N_p} \delta(\eta_M^i = j)\mathbf{x}_{t-1}^{(i)}$
    $\bar{\mathbf{x}}_t^j = \frac{1}{n_{t-1}^j} \sum_{i=1}^{N_p} \delta(\eta_M^i = j)\mathbf{x}_t^{(i)}$
    Update $\Theta_{t,M}^j$ with $(\bar{\mathbf{x}}_{t-1}^j, \bar{\mathbf{x}}_t^j)$
  end for
  for $j = 1$ **to** R_O **do**
    $n_{t-1}^j = \sum_{i=1}^{N_p} \delta(\eta_O^i = j)$
    $\bar{\mathbf{x}}_t^j = \frac{1}{n_{t-1}^j} \sum_{i=1}^{N_p} \delta(\eta_O^i = j)\mathbf{x}_t^{(i)}$
    Update $\Theta_{t,O}^j$ with $(\bar{\mathbf{x}}_t^j, \mathbf{y}_t)$
  end for
end for

based strategies : Sparse Online GP (SOGP) (Csató and Opper, 2002) and Local GPs (LGP) (Urtasun and Darrell, 2008) where the reduction in memory and/or computation is achieved by an online sparsification and a local experts mechanism respectively. The specific contents of $\Theta_{t,M}^i$ or $\Theta_{t,O}^i$ vary depending on the strategy used. In the case of SOGP it will contain a number of computed quantities which are associated with the active set while for LGP it will simply be the set of all training points. The detailed review of these two GP variants can be found in Chapter 2.

5.1.4 Experiments

To evaluate the effectiveness of our approach, we choose 4 different motions, *i.e.*, walking, golf swing, swimming as well as exercises (composed of side twist and squat). The data consists of motion capture from the CMU dataset [1], where each observation is a 62 dimensional vector

1. CMU Mocap Database is available at http ://mocap.cs.cmu.edu/.

containing the 3D rotations (degree) of all joints. We normalize the data to be zero-mean and subsample the observations to reduce the correlation between consecutive frames. We use a frequency of 12 frames/s for walking and swimming, 24 frames/s for golf swing and 30 frames/s for the exercise motion. We compute all results averaged over 3 trials and report the average root mean squared error (RMSE) between ground truth and our predicted output, as our measure of performance.

In all our experiments, the dimensionality of the latent space is set to be 3 as is common for human motion models (Wang et al., 2008). We use PCA to initialize the latent space, and use K-means to obtain the data points used for the mixture components. We choose the compound kernel function $k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp(-0.5 \parallel \mathbf{x} - \mathbf{x}' \parallel^2 / \gamma^2) + l^2 \mathbf{x}^T \mathbf{x}'$ for both prediction and observation mappings. Unless otherwise stated, we use 50 particles, a training set of size of 20/30/50/450 and 2/2/5/5 mixture components for walking/golf/swimming/exercise motions respectively. For LGP, the number of local GP experts is 2/2/2/5, and the size of each local expert is 5/8/5/20. For SOGP, the size of the active set is 20/5/50/20. Note that the parameter values were chosen to balance computational cost with the prediction accuracy and in our experiments we demonstrate the robustness of our approach to these parameters.

Comparison to State-of-the-art

We compare our approaches to two baselines : The first is the approach of Ko and Fox (2009) where a GPDM is learned offline with gradient descent (Wang et al., 2008) before performing particle filtering for state estimation. The second baseline is similar, but learns the GPDM offline using stochastic gradient descent (Yao et al., 2011). We tested the baselines in two different settings. First, only the initial training set is available to learn the prediction and observation models. Second, all the data (including future streamed examples) are used to learn the prediction and observation models. We call the latter setting as the oracle for Ko and Fox (2009).

Number of Particles : We evaluate how the accuracy changes as a function of the number of particles, N_p. As expected, the RMSE of the prediction is reduced in all the methods when the number of particles increases. As shown in Figure 5.1, our approaches are superior to the baselines. Importantly, we outperformed the oracle baseline as we are able to represent multi-modal distributions effectively. This is particularly important in the exercise sequence as the dynamics are clearly multimodal due to the different motions which are performed in this sequence. Furthermore, our LGP variant outperforms SOGP. This may be due to the fact that SOGP has a fixed capacity while LGP is able to leverage more training data when making predictions.

Influence of Noise : In this experiment we evaluate the robustness of all approaches to additive Gaussian noise in the observations. Figure 5.2 shows that our SOGP and LGP particle

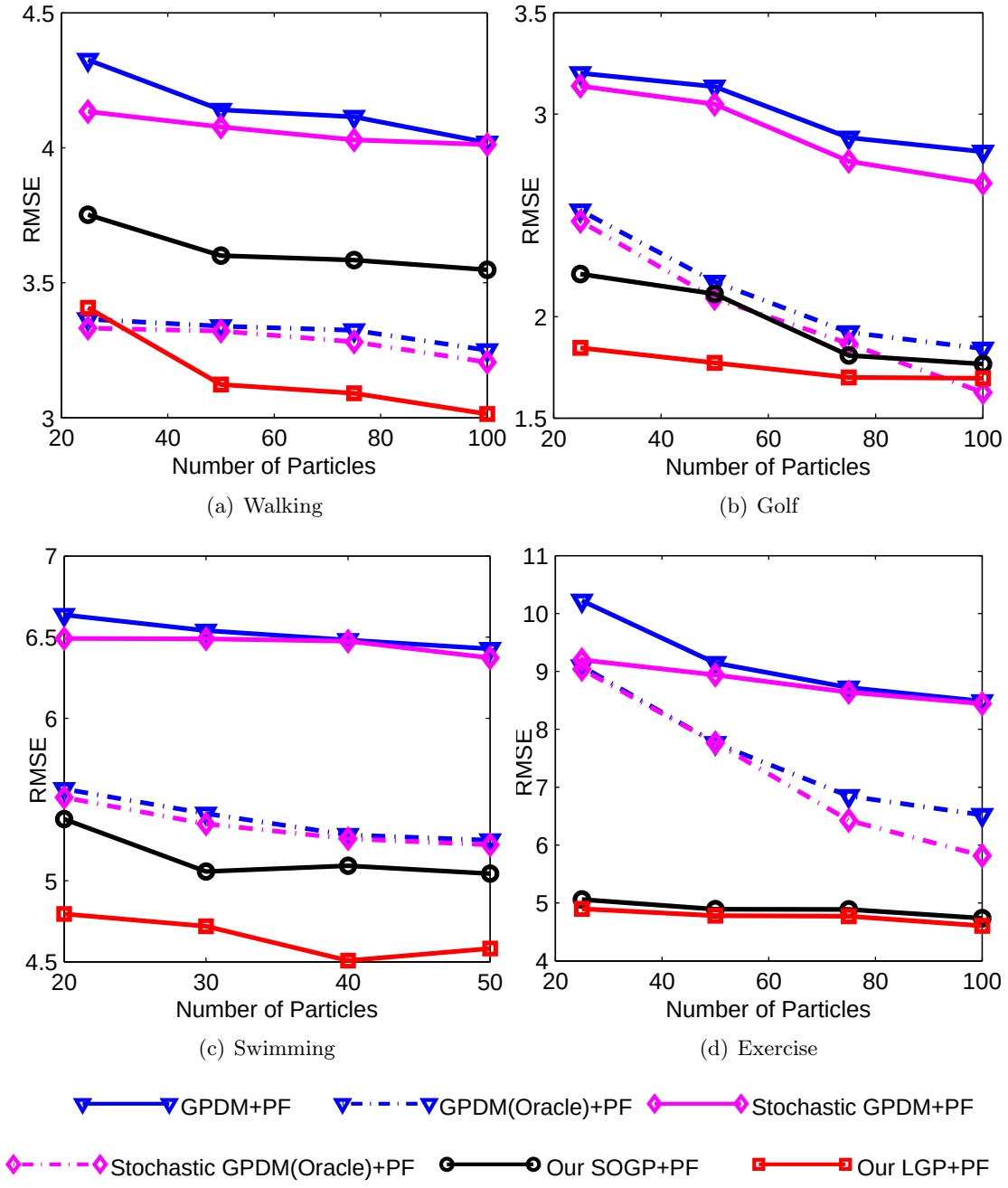
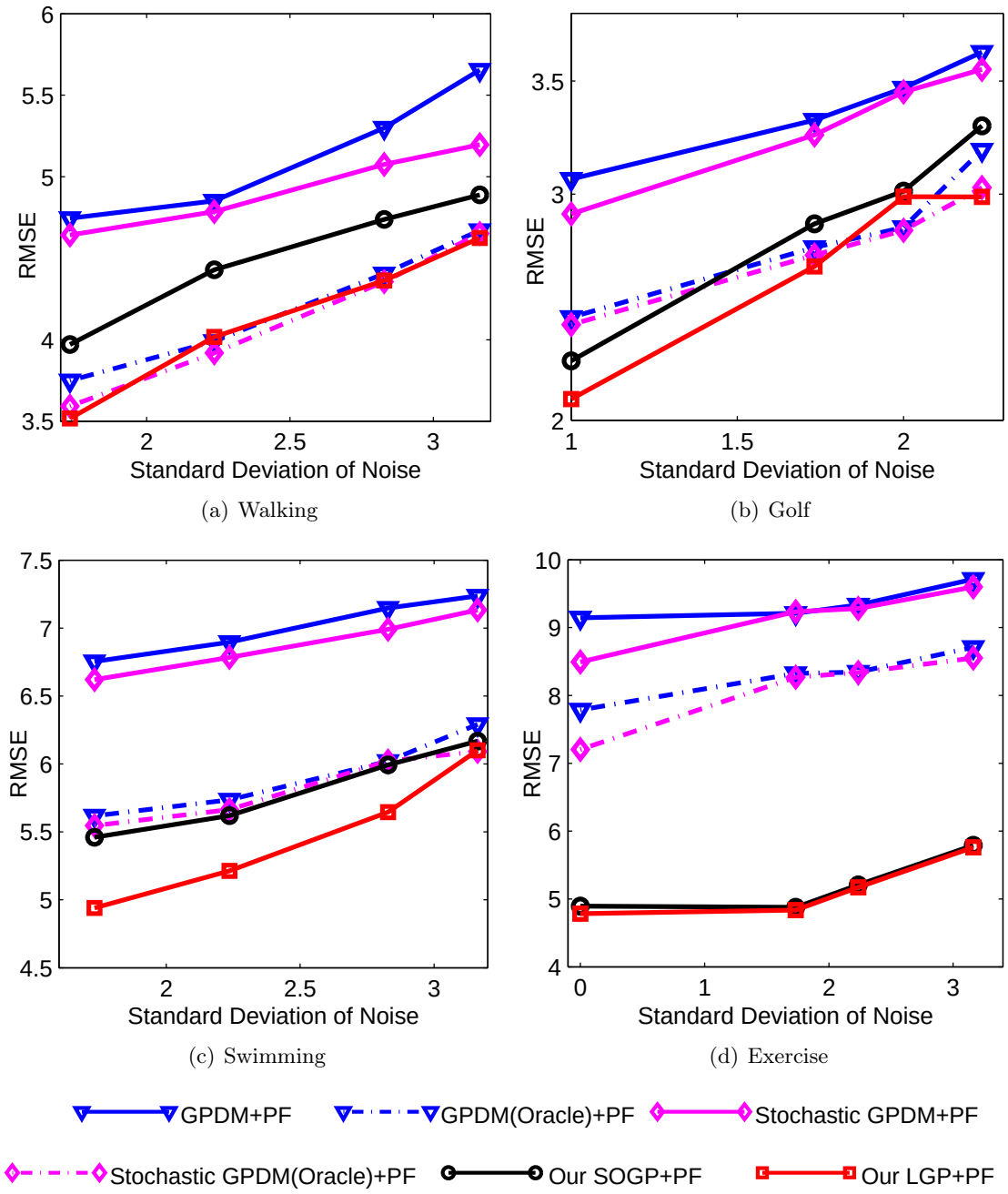
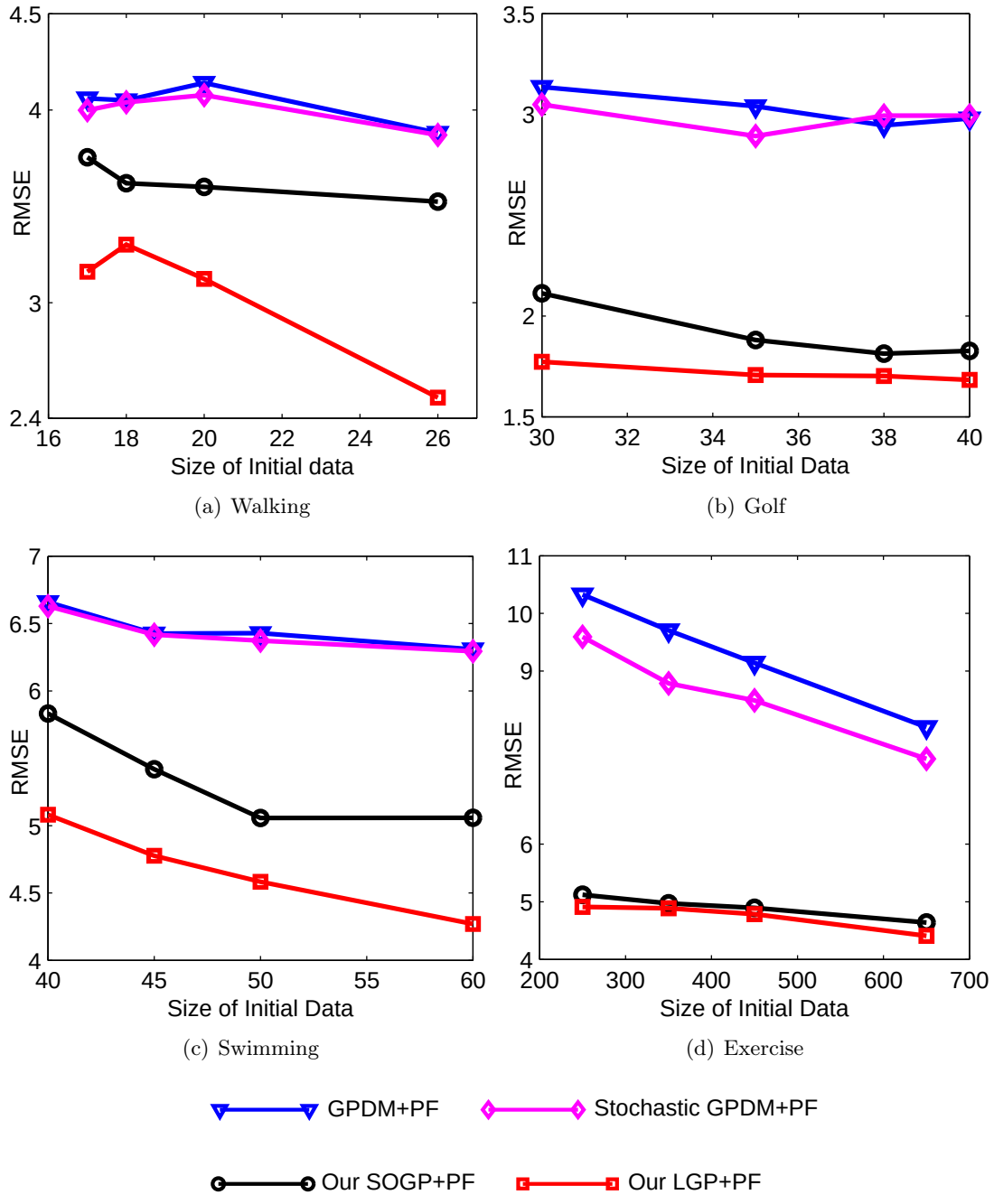


FIGURE 5.1: Root mean squared error (RMSE) as a function of the number of particles in the particle filter.



(a) Walking (b) Golf (c) Swimming (d) Exercise

FIGURE 5.2: Root mean squared error (RMSE) as a function of the standard deviation of noise added to the observations.



(a) Walking

(b) Golf

(c) Swimming

(d) Exercise

FIGURE 5.3: Root Mean Squared Error (RMSE) as a function of the number of initial training points.

filter outperform the baselines, particularly in the exercise sequence which contains multi-modality.

Size of Training Set : We next evaluate how the accuracy depends on the size of the initial training set, T_0. Figure 5.3 clearly indicates that our methods perform well even when the training set is very small. In contrast, the two baselines require bigger training sets to achieve comparable performance. This is expected as the baselines do not update the latent space to take the incoming observations into account.

Qualitative Experiments

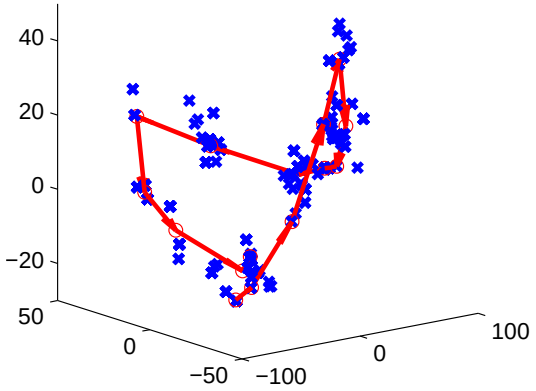
Figure 5.4 to 5.7 show the latent space of both SOGP and LGP filters when employing 50 particles for each time step (depicted in blue). From the 3D latent space and predicted skeletons, we find that the manifolds of both LGP and SOGP particle filters have a good representation of the high-dimensional human motion data.

Properties of Our Approaches

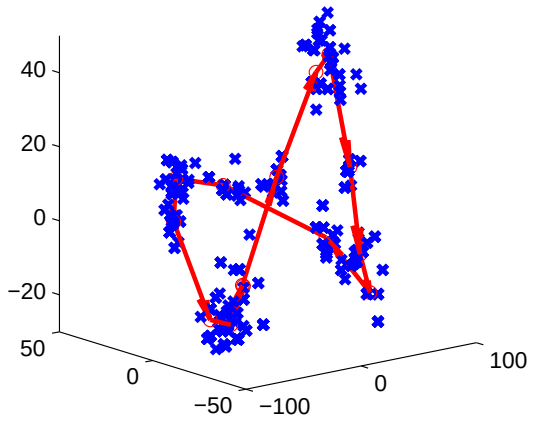
We discuss various aspects of our approaches and evaluate the influence of the parameters of SOGP and LGP particle filters. For LGP, due to the fact that the data sizes of walking, golf and swimming motions are small, we reduce the number of local experts to be able to increase the size of the local experts, or reduce the size of local experts to be able to increase the number of the local experts.

Computational Complexity : Overall the computational complexity of our method (Algoritm 8) is mainly determined by the complexity of constructing a prediction distribution for each components (lines 5-6 and 11-12) and model updates (line 18 and line 23). Specifically, for an individual component which is either SOGP or LGP, computing the prediction distribution is $O(N_A^2)$ or $O(M_{\mathbf{a}}M_{\mathbf{b}}^3 + TM_{\mathbf{a}}M_{\mathbf{b}})$ respectively where N_A is the size of active set, $M_{\mathbf{a}}$ is the number of local experts, $M_{\mathbf{b}}$ are the number of neighbors in the output space and $TM_{\mathbf{a}}M_{\mathbf{b}}$ comes from the KNN search. The model updates for the mixture components (lines 18 and 23) have a computational complexity of $O(N_A^2)$ and $O(1)$ for SOGP and LGP respectively.

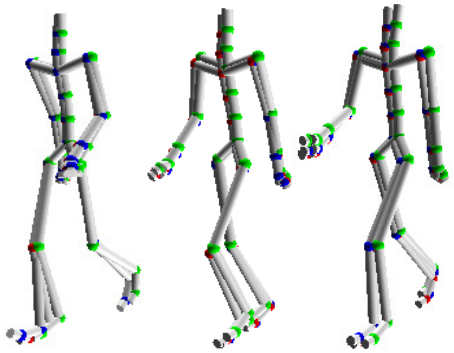
Number of Mixture Components : Figure 5.8 shows performance as a function of the number of mixture components, R_M and R_O, for both SOGP and LGP. For LGP+PF in walk/golf/swim/exercise, the number of local GPs are 1/1/2/5 and the size of each local GP are 3/3/5/20. In all cases, we set $R_M = R_O$ for convenience. Note that performance typically increases with the number of mixture components, for SOGP, but less so for LGP. Furthermore, while LGP generally outperforms SOGP, the difference quickly declines as the number of mixture components increases. Finally, our approaches outperform the baselines in which the model is not updated during filtering indicating that online model updating is important in practice.



(a) 3D Walk (Our SOGP+PF)

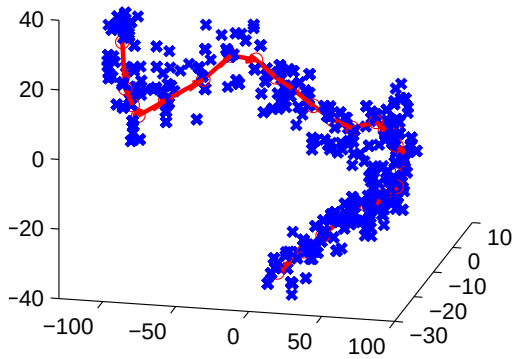


(b) 3D Walk (Our LGP+PF)

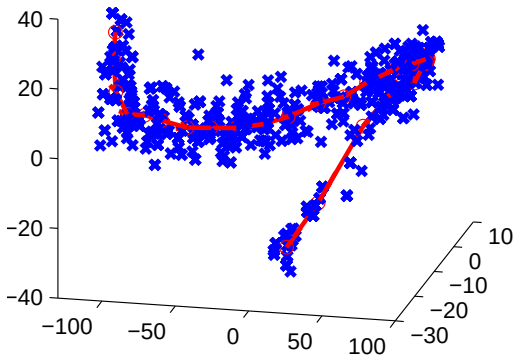


(c) Walk (t=23, 26, 29)

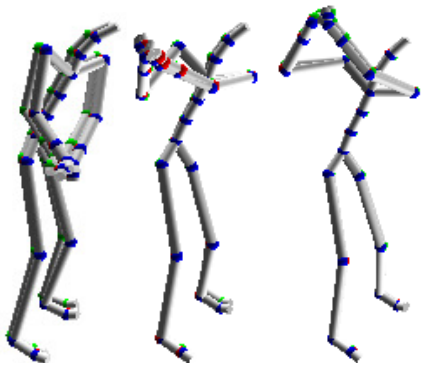
FIGURE 5.4: 3D latent spaces (Walking) while tracking. The first two rows depict the latent space learned by our SOGP variant and our LGP variant. In these two plots the red curve represents the predicted mean of the latent state sequence and the blue crosses are the particles at each step. The last row depicts the predicted skeletons, where ground truth is shown in green, our SOGP variant in blue and our LGP variant in red.



(a) 3D Golf (Our SOGP+PF)

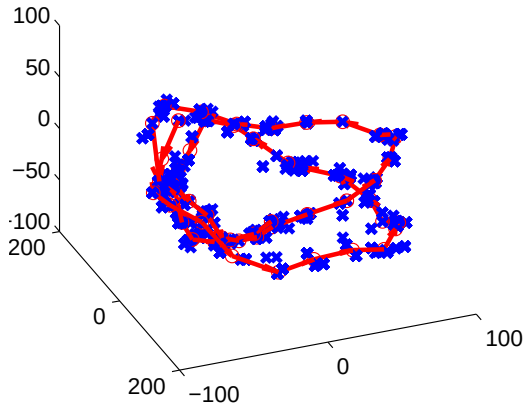


(b) 3D Golf (Our LGP+PF)

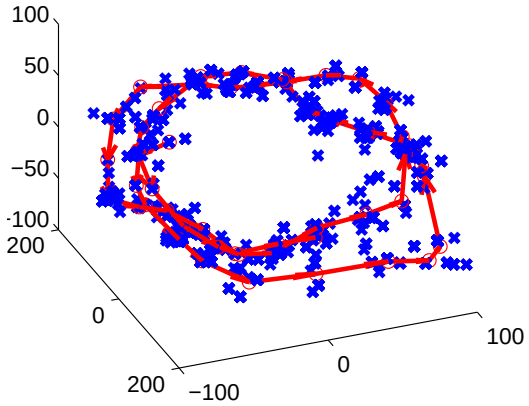


(c) Golf (t=33, 41, 54)

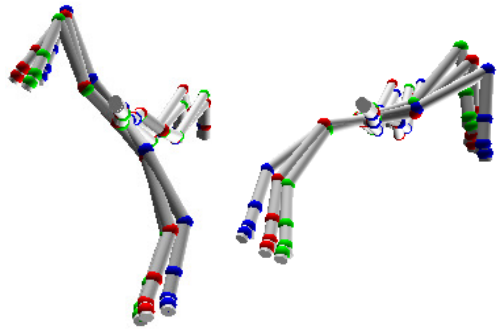
FIGURE 5.5: 3D latent spaces (Golf) while tracking. The first two rows depict the latent space learned by our SOGP variant and our LGP variant. In these two plots the red curve represents the predicted mean of the latent state sequence and the blue crosses are the particles at each step. The last row depicts the predicted skeletons, where ground truth is shown in green, our SOGP variant in blue and our LGP variant in red.



(a) 3D Swim (Our SOGP+PF)

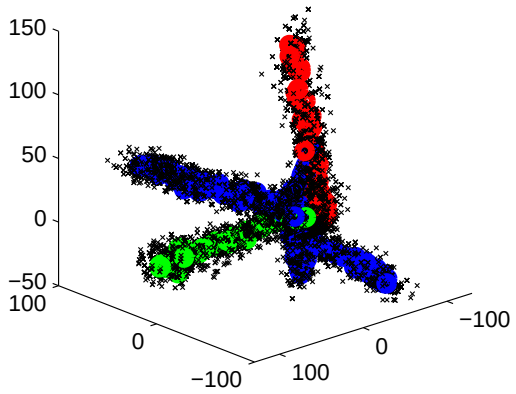


(b) 3D Swim (Our LGP+PF)

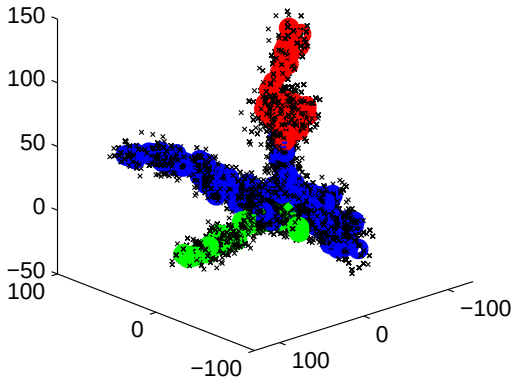


(c) Swim (t=71,79)

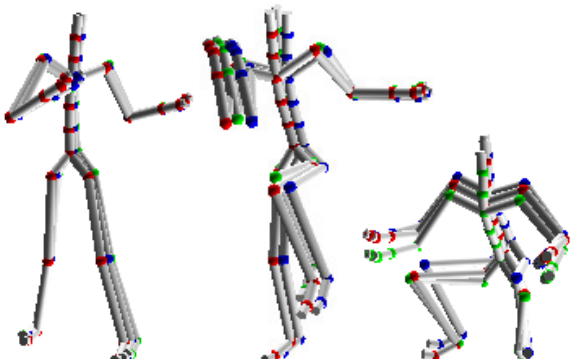
FIGURE 5.6: 3D latent spaces (Swimming) while tracking. The first two rows depict the latent space learned by our SOGP variant and our LGP variant. In these two plots the red curve represents the predicted mean of the latent state sequence and the blue crosses are the particles at each step. The last row depicts the predicted skeletons, where ground truth is shown in green, our SOGP variant in blue and our LGP variant in red.



(a) 3D Exercise (Our SOGP+PF)



(b) 3D Exercise (Our LGP+PF)



(c) Exercise (t=508,555,721)

FIGURE 5.7: 3D latent spaces (Exercise) while tracking. The first two rows depict the latent space learned by our SOGP variant and our LGP variant. In these two plots, the red, blue and green curves are the predicted mean of the latent state for three motions in the exercise sequence, and the black crosses are the particles at each step. The last row depicts the predicted skeletons, where ground truth is shown in green, our SOGP variant in blue and our LGP variant in red.

Active Set size in SOGP : To explore the effect of the size of the active set, N_A, we set the number of mixture components, R_M and R_O, to be 2/1/5/5 for walk/golf/swim/exercise, and use the same settings as before for the other parameters. Results are shown in Figure 5.9(a). As expected the performance improves when the size of the active set increases.

Number and Size of Local Experts in LGP : Figure 5.9(b) and 5.9(c) show the performance of our approach as a function of the number of local GP experts, $M_{\mathbf{a}}$, as well as their size, $M_{\mathbf{b}}$. For this experiment we set the number of mixture components, R_M and R_O, to be 1/2/1/5 and used the same settings as before for the other parameters except when evaluating the size of each local GP expert where we set the number of local GP experts to 5/2/5/5. As shown in Figure 5.9(b) and 5.9(c), even with the small number (size) of local GP experts, we still achieve good performance.

Handling Missing Data

We now evaluate the capabilities of our approaches to handle missing data. To do that, we assume that the initial set has no missing values, but a fixed set of joint angles are missing for all incoming frames. Our approach is able to cope with the missing data problem with only two small modifications. First, particles are weighted only based on the observed dimensions. Furthermore, when updating the prediction and observation models, we employ mean imputation for the missing observation dimensions. Figure 5.10 shows reconstructions of the missing dimensions for all our motions, which consist of the two legs for walking, the left arm for golf swing, swimming and exercise motions. We can see that our approach is able to reconstruct the missing parts well.

Finally, to evaluate the tracking performance as a function of the number of missing dimensions, we randomly generate the indices for the missing dimensions and use the same missing dimensions for all incoming frames. Figure 5.11 shows that, compared to the baselines, our approaches perform well even when the number of missing dimensions is 20 (1/3 of the skeleton) for all the motions. In addition, our LGP particle filter outperforms our SOGP variant since LGP could leverage more training data when making predictions.

5.1.5 Summary

In Wang et al. (2014), we have presented a novel non-parametric method to Bayesian filtering, where the observation and prediction models are learned in an online fashion by using a mixture model with GP components. We have demonstrated that our approaches can capture the multimodality accurately and efficiently due to the fact that two fast GP variants are used to update individual mixture components. By tracking different human motions and exploring the impact of various parameters on performance, we can see that our approaches are effective. Furthermore, our local GP particle filter proved superior to our SOGP variant, however the differences can be mitigated by using more mixture components in our SOGP particle filter.

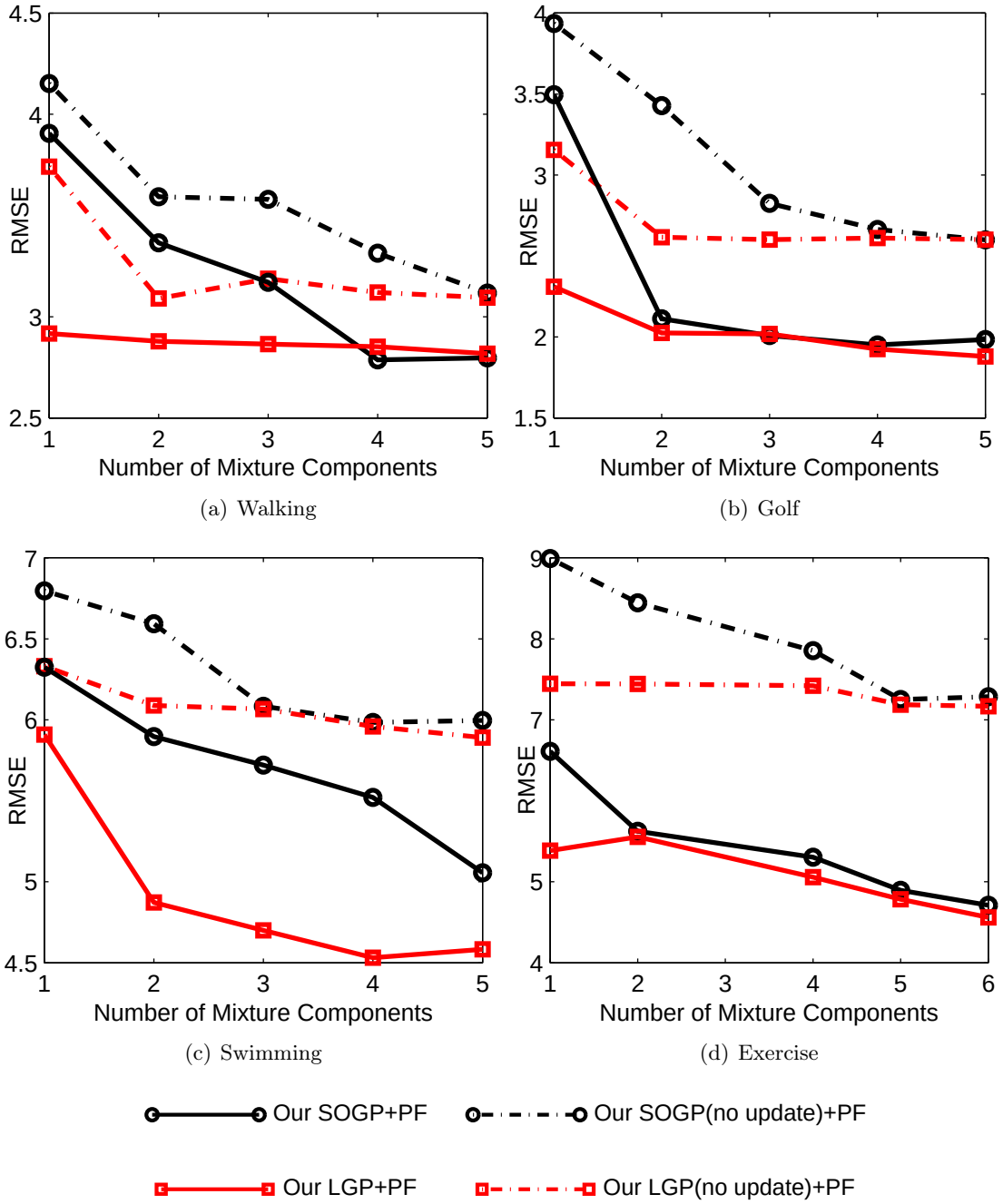


FIGURE 5.8: Root Mean Squared Error (RMSE) as a function of the number of mixture components.

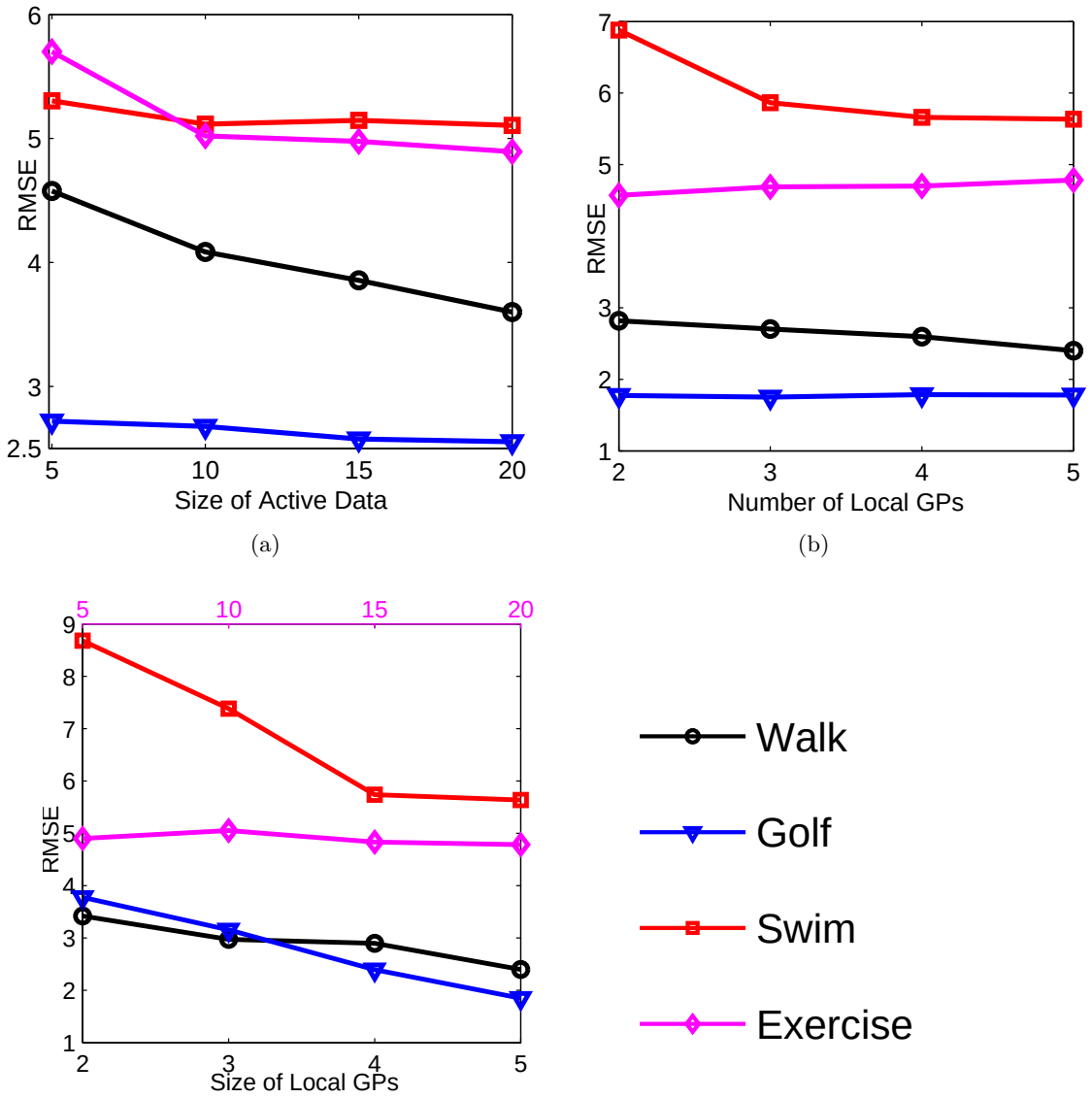


FIGURE 5.9: Root mean squared error (RMSE) as a function of the size of the active set in SOGP, the number of the local GP experts and the size of each local GP expert in LGP. In the subplot 5.9(c), the top x-axis is for exercise and the bottom one for the other motions.

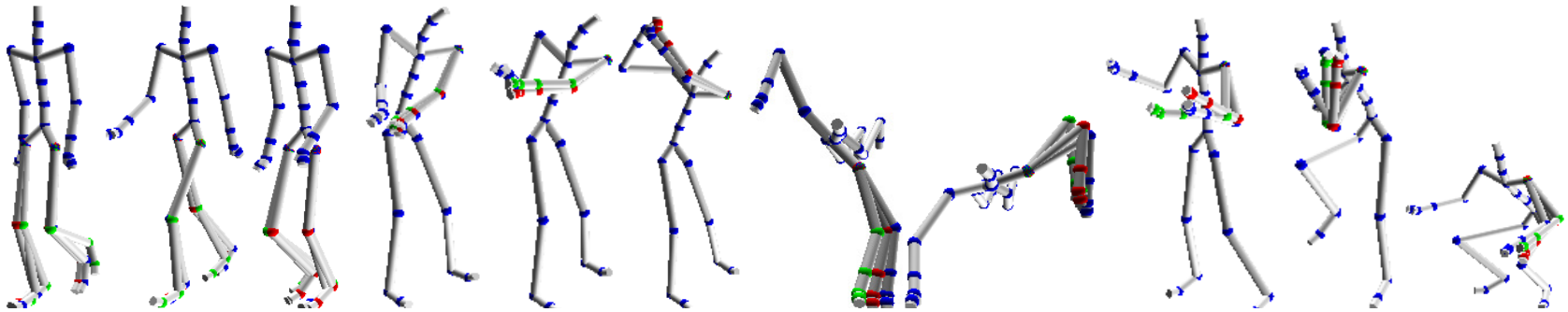


FIGURE 5.10: Predicted skeleton for missing parts (Walk : two legs ; Golf, Swim and Exercise : left arm). The ground truth is shown in green, our SOGP particle filter in blue and LGP particle filter in red. We show the predicted performance at t=24, 27, 32 for walk, t=34, 38, 50 for golf, t=71,79 for swim, t=470, 592, 711 for exercise.

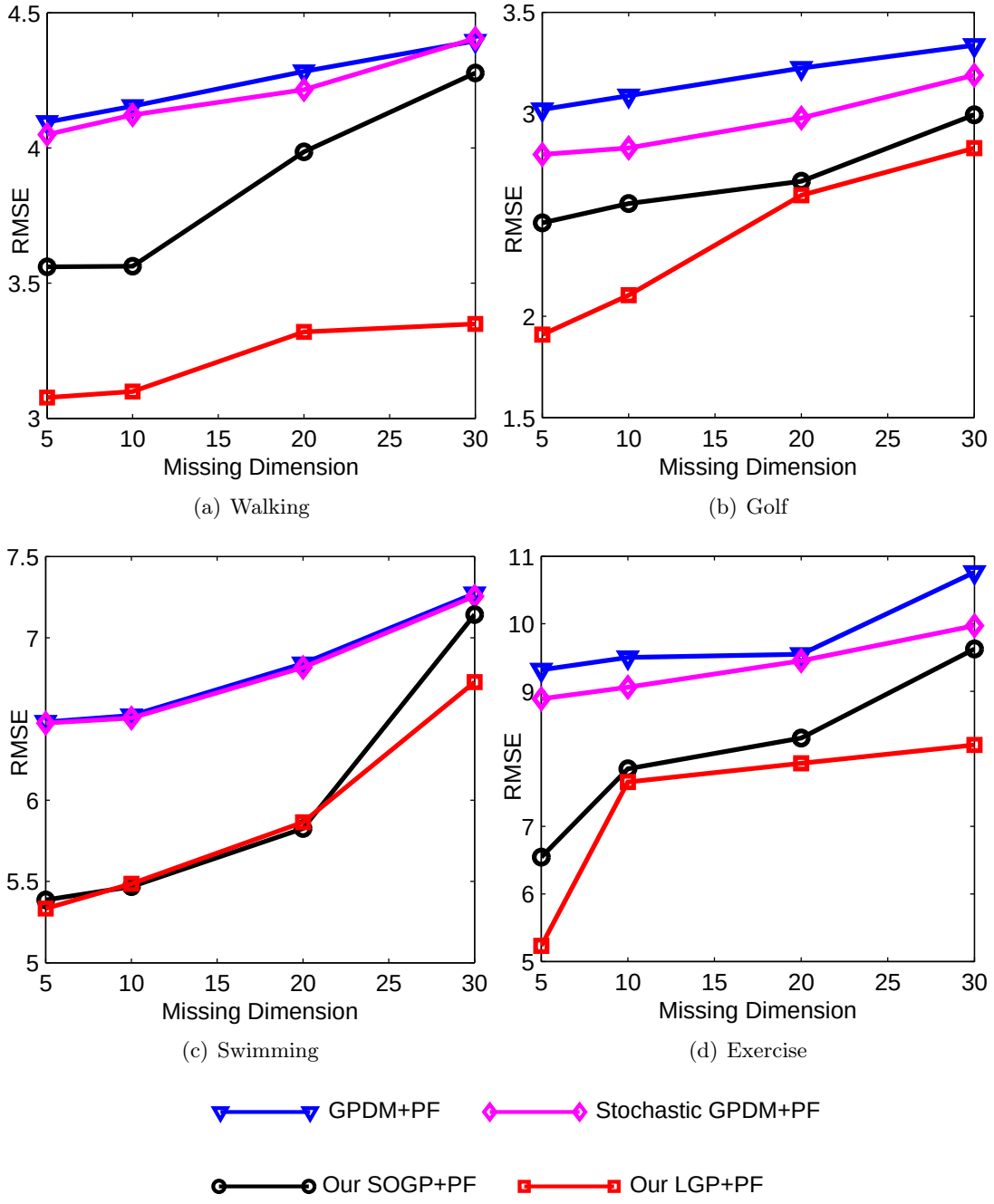


FIGURE 5.11: Root Mean Squared Error (RMSE) as a function of the number of the missing dimensions.

Finally, it is worth mentioning that our two resulting online GP particle filters can be treated as an online method for both GP modeling and sequential Bayesian estimation. This is credited to our novel *Dynamical Interaction Between Gaussian Processes and Bayesian Estimation*.

In the following, we will design a heteroscedastic deep GP network for which we present an efficient and accurate sequential inference framework by taking advantage of this novel dynamical interaction between GPs and Bayesian estimation.

5.2 Efficient Sequential Inference for Heteroscedastic Deep Gaussian Processes Regression

Many real-world applications often reflect input-dependent correlations between multiple output variables. Deep Gaussian process (Damianou and Lawrence, 2013) is a flexible deep neural network to capture such input-dependencies by modeling GP mappings in each layer. However, a deep GP is often limited when the observations are heteroscedastic, and also its network structure often makes the inference intractable when the size of the data sets increases.

In Wang and Chaib-draa (2015b), we propose a heteroscedastic deep GP (HDGP) which can be seen as a generalized version of both heteroscedastic GP and deep GP. By taking advantage of *Dynamical Interaction Between Gaussian Processes and Bayesian Estimation*, we design an efficient sequential inference mechanism for our HDGP to estimate the latent variables and update the model in a recursive Bayesian manner. We show that our HDGP outperforms several state-of-the-art GP variants on three real-world data sets.

5.2.1 Motivations

As we discussed in Section 4, a standard GP is often limited when the data sets reflect input-dependent properties such as non-stationarity and heteroscedasticity (Rasmussen and Williams, 2006; Schmidt and O'Hagan, 2003; Goldberg et al., 1998). To address such challenging data properties, several GP variants have been proposed by designing non-stationary covariance functions (Rasmussen and Williams, 2006; Schmidt and O'Hagan, 2003; Paciorek and Schervish, 2003), putting another GP prior on the log-noise term (Goldberg et al., 1998; Kersting et al., 2007; Lázaro-Gredilla and Titsias, 2011) or warping GPs with different nonlinear functions (Snelson et al., 2003; Adams and Stegle, 2008; Lázaro-Gredilla, 2012). However, all these methods are designed for a standard regression task with one-dimensional output. Hence their performance is often deteriorated for multi-output tasks as the correlations between outputs are ignored by using these GP variants independently for each output variable.

Over the past years, multi-output GPs have been significantly investigated (Boyle and Frean, 2004; Bonilla et al., 2007; Alvarez and Lawrence, 2008) as the dependencies between outputs can improve the prediction accuracy. However, in these multi-output GP approaches the corre-

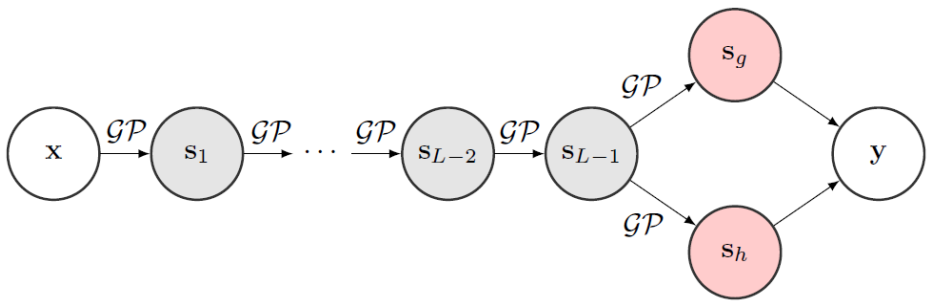


FIGURE 5.12: Heteroscedastic Deep Gaussian Processes Model.

lations between outputs are independent of the input space (Wilson et al., 2012). Consequentially their performance is often limited when the data sets reflect strong input-dependencies (Wilson et al., 2012; Damianou and Lawrence, 2013). Recently, a deep GP model has been proposed by Damianou and Lawrence (2013), based on the fact that a GP is a multi-layer perception with infinite units in the hidden layer (Neal, 1996). The resulting deep belief network, which is based on GP mappings in each layer, can capture the input-dependent correlations between outputs. However, its performance may be limited for heteroscedastic cases since heteroscedasticity is not explicitly modeled in deep GP. Moreover, its network structure often makes the inference intractable when the size of data sets increases.

In the following contribution (Wang and Chaib-draa, 2015b), we propose a novel heteroscedastic deep GP (HDGP) to interpret the input-dependent noise correlations between output variables. To achieve this goal, first we explicitly model heteroscedasticity by sharing a deep GP structure between signal and noise in the observation layer. We then design a sequential sampling inference framework, which is inspired by the interaction between GP modeling and Bayesian estimation, to filter out the latent states and update our HDGP in a recursive fashion. Finally, it is worth mentioning that the weighting mechanism in our inference framework can be straightforwardly used to handle missing data tasks in multi-output regression.

5.2.2 Heteroscedastic Deep Gaussian Processes

Given a number of input-output pairs $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ where $\mathbf{x}_n \in \mathbb{R}^{D_x}$ and $\mathbf{y}_n \in \mathbb{R}^{D_y}$, we propose the following heteroscedastic deep GP (HDGP) model that consists of **a deep GP structure** where

• Layer 1

$$\begin{aligned}
&\mathbf{s}_1 = \mathbf{f}_1(\mathbf{x}) + \mathbf{v}_1, \quad \mathbf{s}_1 \in \mathbb{R}^{D_1} \\
&\mathbf{f}_{1,d_1} \sim \mathcal{GP}(0, k_{\theta_{1,d_1}}) \\
&\mathbf{v}_{1,d_1} \sim \mathcal{N}(0, \sigma^2_{1,d_1}) \\
&d_1 = 1 \cdots D_1
\end{aligned}$$

...

• Layer $L-1$

$$\begin{aligned}
&\mathbf{s}_{L-1} = \mathbf{f}_{L-1}(\mathbf{s}_{L-2}) + \mathbf{v}_{L-1}, \quad \mathbf{s}_{L-1} \in \mathbb{R}^{D_{L-1}} \\
&\mathbf{f}_{L-1,d_{L-1}} \sim \mathcal{GP}(0, k_{\theta_{L-1,d_{L-1}}}) \\
&\mathbf{v}_{L-1,d_{L-1}} \sim \mathcal{N}(0, \sigma^2_{L-1,d_{L-1}}) \\
&d_{L-1} = 1 \cdots D_{L-1}
\end{aligned}$$

with a heteroscedastic observation layer

$$\mathbf{y} = \mathbf{s}_h + \mathbf{v}_y, \quad \mathbf{v}_y \sim \mathcal{N}(\mathbf{0}, e^{\mathbf{s}_g}\mathbf{I})$$

where the output of the deep GP, $\mathbf{s}_{L-1}$, is the input of both signal functions $\mathbf{s}_h$

$$\begin{aligned}
&\mathbf{s}_h = \mathbf{h}(\mathbf{s}_{L-1}) + \mathbf{v}_h, \quad \mathbf{s}_h \in \mathbb{R}^{D_y} \\
&\mathbf{h}_d \sim \mathcal{GP}(0, k_{\theta_{h,d}}) \\
&\mathbf{v}_{h,d} \sim \mathcal{N}(0, \sigma^2_{h,d}) \\
&d = 1 \cdots D_y
\end{aligned}$$

and noise functions $\mathbf{s}_g$ in this layer

$$\begin{aligned}
&\mathbf{s}_g = \mathbf{g}(\mathbf{s}_{L-1}) + \mathbf{v}_g, \quad \mathbf{s}_g \in \mathbb{R}^{D_y} \\
&\mathbf{g}_d \sim \mathcal{GP}(0, k_{\theta_{g,d}}) \\
&\mathbf{v}_{g,d} \sim \mathcal{N}(0, \sigma^2_{g,d}) \\
&d = 1 \cdots D_y
\end{aligned}$$

For convenience, we put all the hyper-parameter vectors θ_{i,d_i} $(i = 1, \cdots, L-1)$, $\theta_{h,d}$, $\theta_{g,d}$ of the covariance functions and all the variances of the noises σ^2_{i,d_i} $(i = 1, \cdots, L-1)$, $\sigma^2_{h,d}$, $\sigma^2_{g,d}$ into a model parameter vector that is denoted as Θ.

The graph model of our HDGP is shown in Figure 5.12. It is worth mentioning that our HDGP is a generalized version of both heteroscedastic GP and deep GP.

• When $\mathbf{x} = \mathbf{s}_{L-1}$ and $D_y = 1$, our HDGP is reduced to an one-layer structure which is the typical heteroscedastic GP in (Goldberg et al., 1998).

• When $\mathbf{s}_h = \mathbf{y}$, our HDGP is reduced to L-layer deep GP model in (Damianou and Lawrence, 2013). Furthermore, due to the fact that L-layer deep GP can be treated as a generalized version of warped GP ($L = 2$, $D_1 = 1$, $D_y = 1$) in (Lázaro-Gredilla, 2012) and GP product model ($L = 2$, $D_1 = 2$, $D_y = 1$, $\mathbf{h}(\mathbf{s}_1) = \mathbf{s}_{1,1}e^{\mathbf{s}_{1,2}}$) in (Adams and Stegle, 2008), hence our HDGP can be also seen as a generalized version of these variants of deep GP.

In the next section, we propose an efficient sequential inference mechanism for our HDGP, which is inspired by sequential Monte Carlo sampling (Doucet et al., 2000b; Chopin, 2002), to estimate the latent states and update the model in a recursive manner.

5.2.3 Sequential Inference for HDGP

As mentioned by Damianou and Lawrence (2013), it is often challenging to perform deep GP for large data sets. We propose an efficient sequential inference mechanism (Algorithm 9) by taking advantage of the interaction between Bayesian state estimation and model update.

Suppose that the training pairs $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$ are made available one at a time. For sequential Bayesian inference, we rewrite our HDGP model at the $(n-1)$-th step in a probabilistic manner,

• Deep GP Structure

$$p(\mathbf{s}_1|\mathbf{x}, M_1^{n-1}, \Theta) \tag{5.1}$$

$$\dots$$

$$p(\mathbf{s}_{L-1}|\mathbf{s}_{L-2}, M_{L-1}^{n-1}, \Theta) \tag{5.2}$$

• Heteroscedastic Observation Layer

$$p(\mathbf{s}_h|\mathbf{s}_{L-1}, M_h^{n-1}, \Theta) \tag{5.3}$$

$$p(\mathbf{s}_g|\mathbf{s}_{L-1}, M_g^{n-1}, \Theta) \tag{5.4}$$

$$p(\mathbf{y}|\mathbf{s}_h, \mathbf{s}_g) = \mathcal{N}(\mathbf{s}_h, e^{\mathbf{s}_g}\mathbf{I}) \tag{5.5}$$

where $M_{1:L-1}^{n-1}$, M_h^{n-1} and M_g^{n-1} are model quantities that we will discuss later on.

Sampling for State Estimation : An Approach Inspired from Sequential Monte Carlo

Firstly we infer the latent states at the n-th step by using our HDGP at the $(n-1)$-th step (5.1-5.5) and $(\mathbf{x}_n, \mathbf{y}_n)$. Even though the posterior,

$$p(\mathbf{s}_{1:L-1}^n, \mathbf{s}_h^n, \mathbf{s}_g^n|\mathbf{x}_n, \mathbf{y}_n, M_{1:L-1}^{n-1}, M_h^{n-1}, M_g^{n-1}, \Theta),$$

Algorithm 9 Sequential inference for HDGP.

1: **Input** :
2: The n-th data pair : $(\mathbf{x}_n, \mathbf{y}_n)$
3: The model quantities of HDGP at the $(n-1)$-th step : $M^{n-1}_{1:L-1}$, M^{n-1}_h, M^{n-1}_g
4: **Output** :
5: The latent states at the n-th step : $\hat{\mathbf{s}}^n_{1:L-1}$, $\hat{\mathbf{s}}^n_h$ and $\hat{\mathbf{s}}^n_g$
6: The model quantities of HDGP at the n-th step : $M^n_{1:L-1}$, M^n_h, M^n_g
7: $\cdots\cdots$ **Sequential Monte Carlo Inspired Sampling for State Estimation** $\cdots\cdots$
8: Sampling N_p Particles from (5.1-5.4) that are constructed by (2.15)

$$\mathbf{s}^n_1(1:N_p) \sim p(\mathbf{s}^n_1|\mathbf{x}_n, M^{n-1}_1, \Theta)$$
$$\mathbf{s}^n_i(1:N_p) \sim p(\mathbf{s}^n_i|\mathbf{s}^n_{i-1}(1:N_p), M^{n-1}_i, \Theta),\ (i=2,...L-1)$$
$$\mathbf{s}^n_h(1:N_p) \sim p(\mathbf{s}^n_h|\mathbf{s}^n_{L-1}(1:N_p), M^{n-1}_h, \Theta),\quad \mathbf{s}^n_g(1:N_p) \sim p(\mathbf{s}^n_g|\mathbf{s}^n_{L-1}(1:N_p), M^{n-1}_g, \Theta)$$

9: Weighting N_p Particles by (5.5) & Normalizing Weights

$$\mathbf{w}^n(1:N_p) = p(\mathbf{y}_n|\mathbf{s}^n_h(1:N_p), \mathbf{s}^n_g(1:N_p)),\quad \hat{\mathbf{w}}^n(j) = \mathbf{w}^n(j)/(\sum_{j=1}^{N_p}\mathbf{w}^n(j)),\ (j=1,...N_p)$$

10: State Estimation

$$\hat{\mathbf{s}}^n_i = \sum_{j=1}^{N_p}\hat{\mathbf{w}}^n(j)\mathbf{s}^n_i(j),\ (i=1,...L-1),\quad \hat{\mathbf{s}}^n_h = \sum_{j=1}^{N_p}\hat{\mathbf{w}}^n(j)\mathbf{s}^n_h(j),\quad \hat{\mathbf{s}}^n_g = \sum_{j=1}^{N_p}\hat{\mathbf{w}}^n(j)\mathbf{s}^n_g(j)$$

11: $\cdots\cdots$ **Sparse Online Gaussian Process (SOGP) for Model Update** $\cdots\cdots$
12: Updating $M^{n-1}_{1:L-1}$, M^{n-1}_h, M^{n-1}_g to $M^n_{1:L-1}$, M^n_h, M^n_g by using SOGP in Subsection 2.1.2 with $(\mathbf{x}_n, \hat{\mathbf{s}}^n_1)$, $\cdots$, $(\hat{\mathbf{s}}^n_{L-2}, \hat{\mathbf{s}}^n_{L-1})$, $(\hat{\mathbf{s}}^n_{L-1}, \hat{\mathbf{s}}^n_h)$, $(\hat{\mathbf{s}}^n_{L-1}, \hat{\mathbf{s}}^n_g)$

is analytically intractable, we can use Bayes rule to obtain that it is proportional to

$$\begin{aligned} &p(\mathbf{y}_n|\mathbf{s}^n_h, \mathbf{s}^n_g)p(\mathbf{s}^n_h|\mathbf{s}^n_{L-1}, M^{n-1}_h, \Theta)p(\mathbf{s}^n_g|\mathbf{s}^n_{L-1}, M^{n-1}_g, \Theta) \\ &\times p(\mathbf{s}^n_1|\mathbf{x}_n, M^{n-1}_1, \Theta)\textstyle\prod_{i=2}^{L-1} p(\mathbf{s}^n_i|\mathbf{s}^n_{i-1}, M^{n-1}_i, \Theta). \end{aligned} \tag{5.6}$$

Inspired by sequential Monte Carlo (Doucet et al., 2001; Cappé et al., 2007), we propose a particle-based-sampling mechanism (Algorithm 9, Line 8-10) to estimate the latent states at the n-th step. Given $\mathbf{x}_n$, the particles are propagated layer by layer (Algorithm 9, Line 8). Once the particles arrive at the observation layer, $\mathbf{y}_n$ is used to evaluate the weights of the particles (Algorithm 9, Line 9). Finally, the latent states can be estimated by using the weighted particles (Algorithm 9, Line 10).

After obtaining the estimated states, we can construct the data pairs at n-th step, i.e., $(\mathbf{x}_n, \hat{\mathbf{s}}^n_1)$, $\cdots$, $(\hat{\mathbf{s}}^n_{L-2}, \hat{\mathbf{s}}^n_{L-1})$, $(\hat{\mathbf{s}}^n_{L-1}, \hat{\mathbf{s}}^n_h)$ and $(\hat{\mathbf{s}}^n_{L-1}, \hat{\mathbf{s}}^n_g)$. These data pairs are then used to update our HDGP at the (n-1)-th step (5.1-5.4) to obtain our HDGP at n-th step (Algorithm 9, Line 12). However, a standard GP implementation requires $O(n^3)$ computation load, and it will

make sequential inference computationally unbounded since n grows over time. To efficiently update our HDGP in a sequential manner, we propose to use a well-known sparse online GP (SOGP) (Csató and Opper, 2002; Vaerenbergh et al., 2012) to reduce the computation burden by iteratively selecting an active set from the training set.

Sparse Online Gaussian Process

The model quantities of SOGP is associated with an active set (Csató and Opper, 2002; Vaerenbergh et al., 2012). Since the training pairs are available one at a time, we denote the model quantities of SOGP at the (n-1)-th step are $M_{SOGP}^{n-1} = \{\mathcal{D}_{n-1}, \mu_{n-1}, \Sigma_{n-1}, Q_{n-1}\}$ where $\mathcal{D}_{n-1}$ is the active set, $\mathcal{N}(\mu_{n-1}, \Sigma_{n-1})$ is the posterior over the latent function values of the active set, Q_{n-1} is the inverse kernel matrix of the active set. Due to the fact that all the GP models in our HDGP are SOGP, M_{SOGP}^{n-1} can be $M_{1:L-1}^{n-1}$, M_h^{n-1} or M_g^{n-1} as specified in (5.1-5.4).

Basically SOGP is associated with two crucial steps in our HDGP. Firstly, we draw particles from the (n-1)-th HDGP model (Algorithm 9, Line 8). This process is based on the predictive distribution of SOGP (Equation 2.15). Secondly, once we obtain the estimated latent states, we will use the new data points $(\mathbf{x}_n, \hat{\mathbf{s}}_1^n), \cdots, (\hat{\mathbf{s}}_{L-2}^n, \hat{\mathbf{s}}_{L-1}^n)$, $(\hat{\mathbf{s}}_{L-1}^n, \hat{\mathbf{s}}_h^n)$, $(\hat{\mathbf{s}}_{L-1}^n, \hat{\mathbf{s}}_g^n)$ to update the (n-1)-th model quantities $M_{1:L-1}^{n-1}$, M_h^{n-1}, M_g^{n-1} to the n-th model quantities $M_{1:L-1}^{n}$, M_h^{n}, M_g^{n} (Algorithm 9 Line 12). The detailed updating procedure can be found in Subsection 2.1.2.

Discussion

As all the GPs in our HDGP are SOGPs, the computation complexity of each GP mapping in our HDGP is governed by $O(N_{AC}^2)$ for each update step where N_{AC} is the size of the active set. Additionally, after obtaining our HDGP by passing all the training pairs through Algorithm 9, one may be interested in learning hyper-parameters Θ. It can be done straightforwardly because the active sets, which we infer from our sequential inference mechanism, can be directly used as the data sets to efficiently refine Θ with a standard gradient optimization of GP. In the experiment we follow the strategy in Csató and Opper (2002); Turner et al. (2010) to iteratively perform sequential inference and hyper-parameter learning in an EM fashion.

Finally, it is worth mentioning that our sequential inference mechanism allows us to easily deal with missing data tasks in multi-output regression. Suppose that several dimensions of $\mathbf{y}_n$ are missing, the only modification in Algorithm 9 is particle weighting (Line 9) in which the particles are weighted by using the observed dimensions of $\mathbf{y}_n$.

Prediction

After using all the training pairs to obtain our final HDGP with model quantities $M_{1:L-1}^{Final}$, M_h^{Final}, M_g^{Final}, we can make a prediction for a given test input $\mathbf{x}^\star$ straightforwardly by

passing $\mathbf{x}^\star$ into particle sampling mechanism (Algorithm 9, Line 8) where $\mathbf{x}_n$ is changed to $\mathbf{x}^\star$, and $M_{1:L-1}^{n-1}$, M_h^{n-1}, M_g^{n-1} are changed to $M_{1:L-1}^{Final}$, M_h^{Final}, M_g^{Final}. Once we obtain the particles $\mathbf{s}_h^\star(1:N_p)$ and $\mathbf{s}_g^\star(1:N_p)$, the predictive distribution of the output is a Gaussian mixture (refer to Equation 5.5),

$$\frac{1}{N_p}\sum_{j=1}^{N_p}\mathcal{N}(\mathbf{s}_h^\star(j), e^{\mathbf{s}_g^\star(j)}\mathbf{I}).$$

5.2.4 Experiments

To show that our HDGP can interpret the input-dependent properties efficiently and accurately, we evaluate our HDGP and its variants on three distinct data sets : motor (Rasmussen and Ghahramani, 2001), parkinsons [2] and flu data [3].

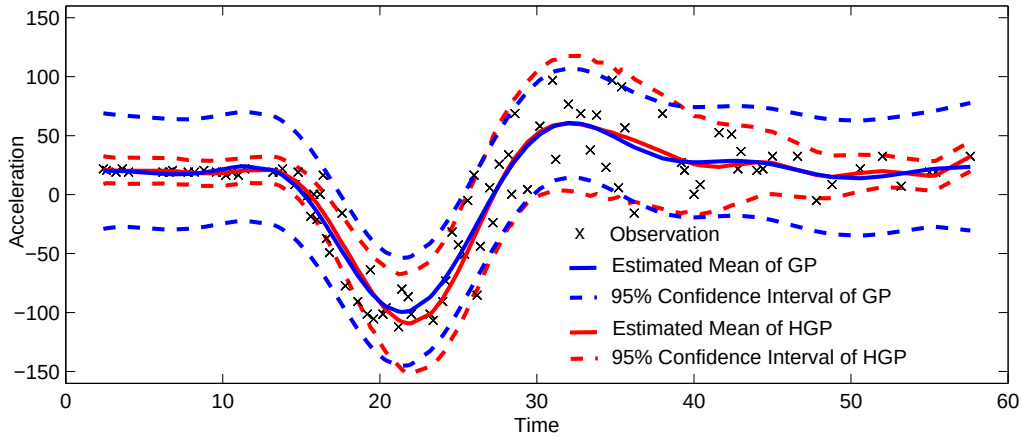
In all the experiments, we follow Damianou and Lawrence (2013) to use Automatic Relevance Determination (ARD) squared exponential kernel as the covariance functions in our HDGP, $k(\mathbf{x}, \mathbf{x}') = \sigma_{ker}^2 \exp[-0.5\sum_{i=1}^d c_i(\mathbf{x}_i - \mathbf{x}_i')^2]$ where the hyper-parameters are σ_{ker}^2, c_1, $\cdots$, c_d. Moreover, as suggested in (Rasmussen and Williams, 2006) we use normalized mean squared error (NMSE) & mean negative log probability (MNLP) as the criterion for prediction error, and use training time (second) as the criterion for training efficiency. Finally, we run all the methods 5 times and report the average results for comparison.

Motor Data : Heteroscedastic GP (HGP) View

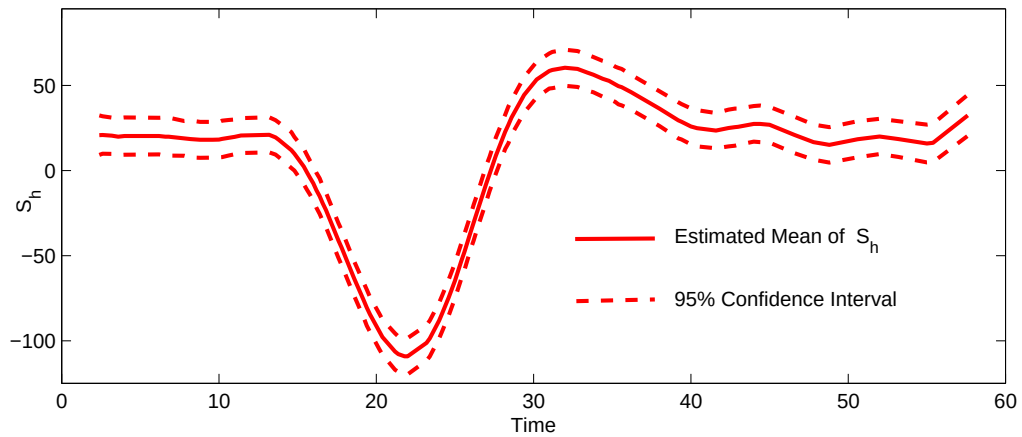
One main contribution of our HDGP is heteroscedasticity (i.e., our HDGP is a generalized version of HGP), so we start our experiments by evaluating our HDGP in a HGP view. Motor data, which consists of 94 time/acceleration (1-D input/1-D output) pairs, is a benchmark data set to evaluate heteroscedasticity (Rasmussen and Ghahramani, 2001). In this data set there are three noise level, i.e., a flat low noise region, a curved region and a flat high noise region.

Firstly, we evaluate the posterior of our HDGP using all the data points. Due to heteroscedasticity, we perform our sequential inference with HGP structure (it is one-layer HDGP, i.e., $L = 1$, $\mathbf{x} = \mathbf{s}_{L-1} \in \mathbb{R}$), and two-layer HDGP ($L = 2$, $\mathbf{x} = \mathbf{s}_{L-2} \in \mathbb{R}$, $\mathbf{s}_1 = \mathbf{s}_{L-1} \in \mathbb{R}$). We denote these two approaches as our HGP and HDGP. The resulting posteriors, which associates with latent state estimation, are shown in Figure 5.13. The number of particles N_p is 1000, the size of active set N_{AC} is the size of training set without sparsification. It is clearly shown that both of our HGP and HDGP outperform the standard GP, and successfully capture the challenging heteroscedastic noise.

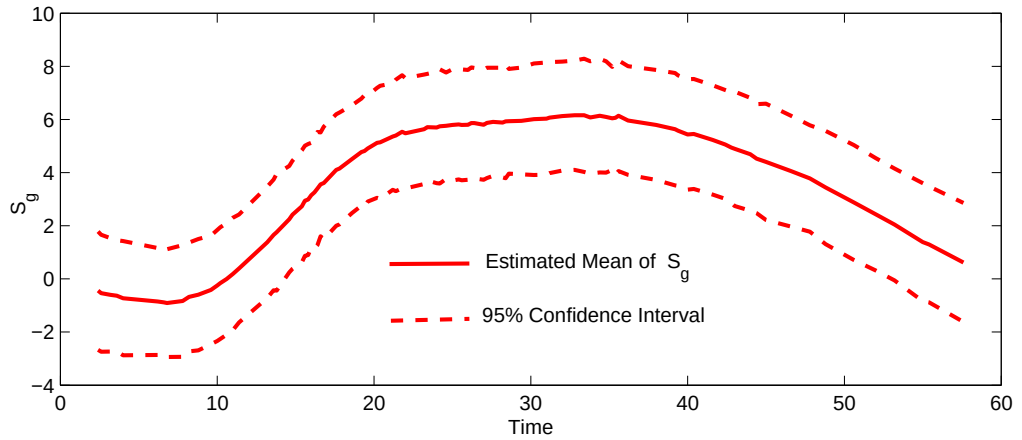
2. http ://archive.ics.uci.edu/ml/datasets.html
3. Google Flu, http ://www.google.org/flutrends



(a) our HGP vs. GP

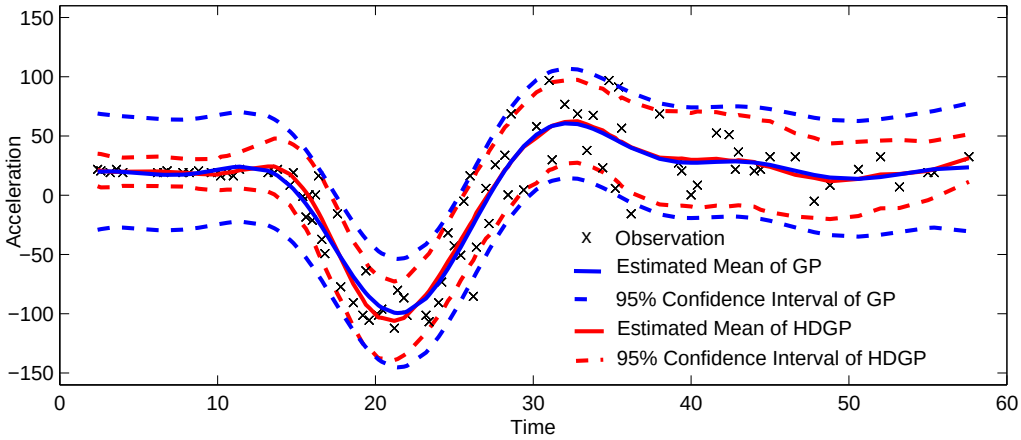


(b) $\mathbf{s}_h$ of our HGP

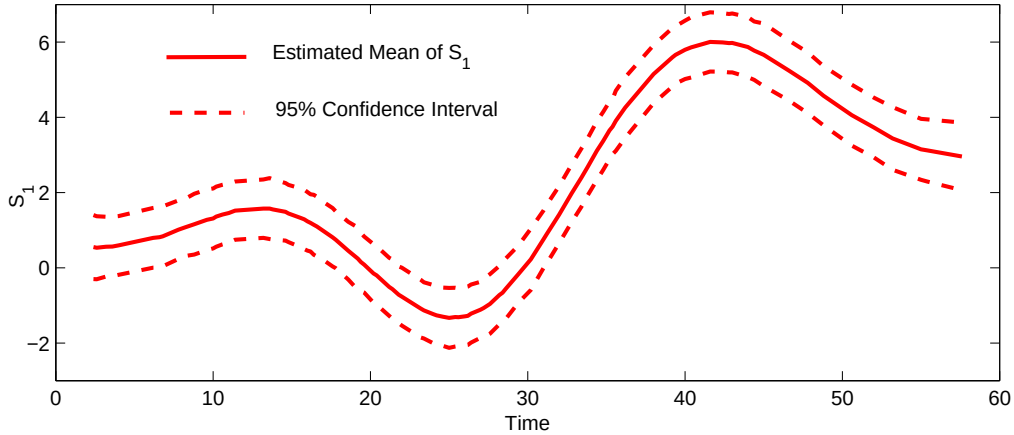


(c) $\mathbf{s}_g$ of our HGP

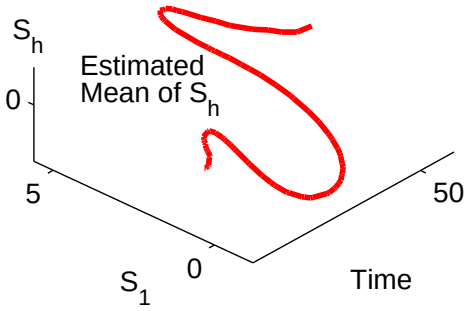
FIGURE 5.13: Posterior Evaluation (Motor Data) by our HGP. In Subplot 5.13(a), the observations are black crosses, the posterior mean of GP / our HGP are blue / red lines, 95% confidence interval of GP / our HGP are blue / red dashed lines.



(a) our HDGP vs. GP



(b) $\mathbf{s}_1$ of our HDGP



(c) $\mathbf{s}_h$ of our HDGP

(d) $\mathbf{s}_g$ of our HDGP

FIGURE 5.14: Posterior Evaluation (Motor Data) by our HDGP. In Subplot 5.14(a), the observations are black crosses, the posterior mean of GP / our HDGP are blue / red lines, 95% confidence interval of GP / our HDGP are blue / red dashed lines.

Secondly, we evaluate the predictive performance. In this experiment, we randomly select 60 data pairs to train the model and the rest 34 data points are used for test. Due to the fact that our HGP and HDGP are based on the particles, we firstly assess average prediction error & training efficiency as a function of N_p. It is shown in Figure 5.15 that average prediction error of both our HGP and HDGP is improved when N_p increases, and trends to converge when N_p is around 1000. Furthermore, our HDGP outperforms our HGP since in our HDGP here the nonlinear mapping $\mathbf{s}_1 = \mathbf{f}_1(\mathbf{x}) + \mathbf{v}_1$ is the input of observation layer (rather than directly using $\mathbf{x}$ as the input of observation layer in our HGP). The tradeoff in our HDGP is the longer training time because of modeling an extra nonlinear mapping $\mathbf{f}_1$.

Finally, we compare our HDGP to several benchmark GP approaches. The selection of GP methods is associated with the dimension of the output. Since this motor data is single-output, we choose single-output GPs for heteroscedasticity comparison.

• We select the standard GP with the ARD kernel (GP) as a base line. Furthermore, since this single-output motor data is heteroscedastic, we choose single-output GPs which can capture heteroscedasticity. Specifically, we select the standard GP with the non-stationary kernel compounded by ARD kernel, neural network kernel and linear kernel (NGP) (Rasmussen and Williams, 2006) ; warped GP (WGP) (Snelson et al., 2003) ; heteroscedastic GP (HGP) (Kersting et al., 2007).

• Since GP product model (GPPM) (Adams and Stegle, 2008) and HGP (Goldberg et al., 1998) are single-output variants of our HDGP. We thus perform our particle based inference on GPPM (our GPPM) and HGP (our HGP) for this single-output motor data.

• Due to the fact that our HDGP in this motor data is two-layer, we choose deep GP (DGP) (Damianou and Lawrence, 2013) with two layers for comparison. Additionally, we also implement our sequential inference mechanism for DGP with two layers (our DGP).

For our HDGP and its variants, N_p is 500 and we show the average results in Table 5.1. The predictive error of GP and NGP is high due to heteroscedasticity. HGP (Kersting et al., 2007) reduces the error by modeling the input-dependent noise. However, compared to our HGP it is still limited. This fact indicates that our sequential inference mechanism is more effective than the one of HGP in (Kersting et al., 2007). Similarly, our DGP outperforms DGP (Damianou and Lawrence, 2013). Moreover, our HDGP further improves the predictive accuracy of our DGP by modeling another GP for noise. Even though the extra GP mapping increases the training time of our HDGP, it is still more efficient (a shorter training time) than DGP in (Damianou and Lawrence, 2013). Hence, we can see that our HDGP with sequential inference mechanism is effective to capture the heteroscedasticity in terms of both accuracy and efficiency.

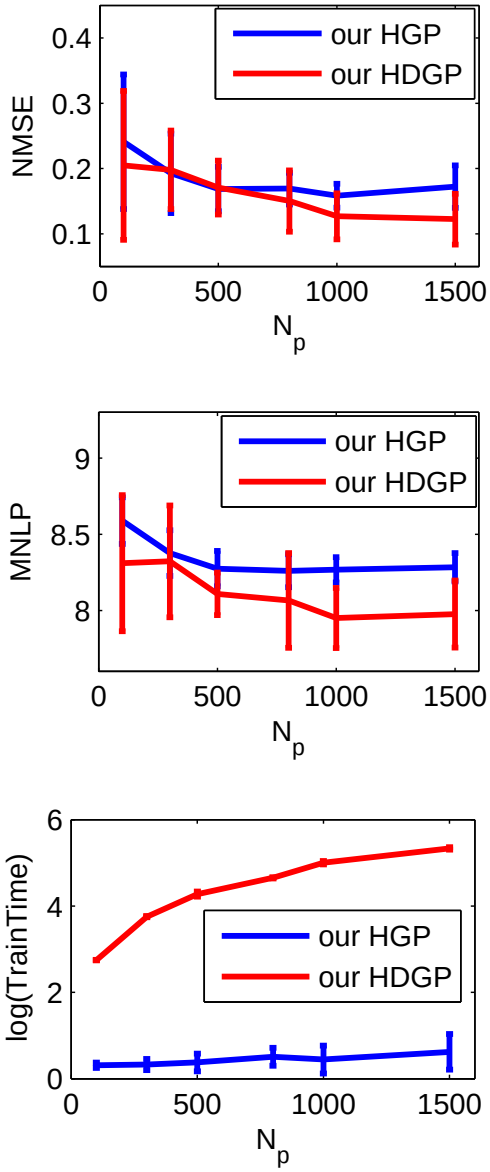


FIGURE 5.15: Predictive performance as a function of N_p (motor data).

Table 5.1: Average prediction error & training efficiency comparison for single-output motor data.

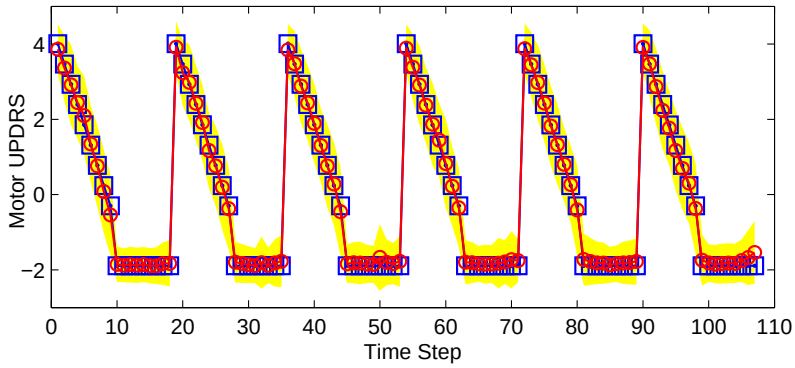
Method	NMSE	MNLP	TrainTime(s)	Method	NMSE	MNLP	TrainTime(s)
HGP	0.198	8.918	1.83	our HGP	**0.169**	**8.275**	1.49
DGP	0.193	9.028	100.1	our DGP	0.186	8.636	38.4
GP	0.202	9.078	1.03	our GPPM	0.180	8.566	1.50
NGP	0.202	9.077	2.26	our HDGP	**0.171**	**8.109**	72.1
WGP	0.196	8.619	4.05				

Parkinsons Data : Deep GP (DGP) View

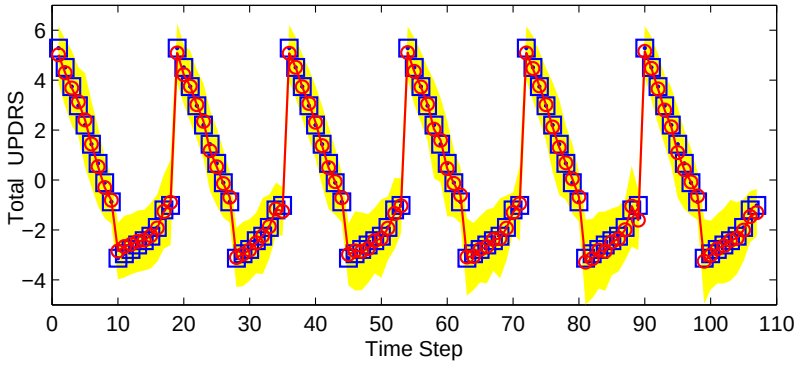
One main contribution of our HDGP is deep network structure (i.e., our HDGP is a generalized version of DGP), so we now evaluate our HDGP in a DGP view by using a parkinsons data set. This data set is a six-month biomedical voice recording from several people with early-stage Parkinson's disease where the input $\mathbf{x}$ is a 16-dimensional biomedical voice feature vector, the output $\mathbf{y}$ is a 2-dimensional score vector (motor UPDRS score and total UPDRS score). Since this data set mainly reflects input-dependent non-stationarity, it is a quite suitable set to evaluate if our sequential inference works well with deep GP structure ($\mathbf{y} = \mathbf{s}_h$).

Firstly, we evaluate the posterior using a six-month recording of one patient in which the number of input-output pairs is 107. The posterior surface of two-dimensional output is shown in Figure 5.16(a) and 5.16(b) where our DGP is a two-layer structure with a two-dimensional latent layer ($\mathbf{s}_1 = [\mathbf{s}_{1,1}, \mathbf{s}_{1,2}]^T$), the number of particles N_p is 1000, the size of active set N_{AC} is the size of training set without sparsification. It is shown that our DGP successfully captures the discontinuity (a type of non-stationarity). Additionally, we show the posterior density of the estimated $\mathbf{s}_1$ in Figure 5.16(c). We find that in the latent space there are two patterns, which is verified by the fact that for each output dimension there are two patterns.

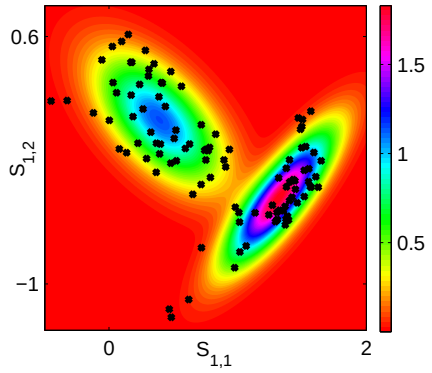
Secondly, we verify the effectiveness of our sequential inference framework for missing data. In this experiment, the fifth-month record of the first output and the second-month record of the second output in the above-mentioned data set are treated as missing data. For comparison, we choose the base line as independent GP (IGP) where each dimension of outputs is independently modeled by a standard GP. Furthermore, we select a multi-output GP (MulGP) variant (Bonilla et al., 2007), which is a well-known approach for missing data. The average predictive NMSE / MNLP for recovering the missing data are respectively 0.944 / 4.573 for IGP, 0.039 / 1.415 for MulGP and **0.015** / **0.522** for our DGP. In this case IGP is poor since the correlations between outputs have been ignored. MulGP improves the accuracy. However, compared to our DGP, its performance is still limited. The reason is that our DGP with sequential inference framework successfully captures the strong input-dependent correlations



(a) Our DGP for Motor UPDRS



(b) Our DGP for Total UPDRS



(c) Posterior for $\mathbf{s}_1$

FIGURE 5.16: Posterior Evaluation (Parkinsons Data). In Subplot 5.16(a) and 5.16(b), the observations are blue line with squares, the posterior mean of our DGP is red line with circles and 95% confidence interval is the yellow region. Subplot 5.16(c) shows the posterior density of our DGP for 2-D latent state $\mathbf{s}_1 = [\mathbf{s}_{1,1}, \mathbf{s}_{1,2}]^T$ where the estimated $\mathbf{s}_1$ are black circles.

Table 5.2: Average prediction error & training efficiency comparison for multi-output parkinsons data.

Park	NMSE	MNLP	TrainTime(s)
MulGP	0.180	8.19	385
DGP	0.492	6.49	1337
our DGP	**0.176**	**5.67**	**100**

between outputs.

Finally, we assess whether our DGP is accurate and computationally efficient for large data sets. We hence choose a six-month recording of another four people where the number of input-output pairs are 575. We randomly select 500 pairs for training and the rest 75 pairs for test. Note that the size of training pairs (500) is moderate, but it is large enough to show the computation efficiency between different approaches (as we show later on). In this experiment, we first evaluate the influence of N_P and N_{AC}. As shown in Figure 5.17, the average prediction accuracy of our DGP increases when N_P and/or N_{AC} increase. Furthermore, due to the fact that the parkinsons data is multi-output, we thus compare our DGP with multi-output GP variants including MulGP (Bonilla et al., 2007) and DGP (Damianou and Lawrence, 2013) where N_P of our DGP is 500, N_{AC} of both our DGP and DGP in (Damianou and Lawrence, 2013) is 100. In Table 5.2, our DGP outperforms both MulGP (Bonilla et al., 2007) and DGP (Damianou and Lawrence, 2013) in terms of both accuracy and efficiency.

Flu Data : HDGP View

After considering our HDGP from both HGP and DGP views, we now incorporate both contributions together to evaluate if our HDGP can correctly interpret both input-dependent signal and noise correlations between different outputs. To achieve this goal, we choose a flu data set which is composed of a record of the flu activity rate in Canada from 2003-11-09 to 2014-03-30. This data set is quite challenging as both non-stationary and heteroscedastic correlations exist. Furthermore, in this flu data there are 543 input-output pairs where the input is the time step, the output is a 9 dimensional vector that consists of the flu activity rate of 9 provinces in Canada (Alberta / British Columbia / Manitoba / New Brunswick / Newfoundland and Labrador / Nova Scotia / Ontario / Saskatchewan / Quebec).

Firstly, we evaluate the posterior of our HDGP where we choose a three-layer structure in which the latent states of the first two layers, $\mathbf{s}_1$ and $\mathbf{s}_2$, are three-dimensional; N_P is 500 and N_{AC} is 100. Figure 5.18 shows the posterior mean of our HDGP layer by layer. Additionally, in Figure 5.18(d) we show the correlations between nine dimensions of $\mathbf{s}_g$ (each dimension represents the noise level of the flu activity rate for one province) and the correlations between nine dimensions of $\mathbf{s}_h$ (each dimension represents the signal level of the flu activity rate for

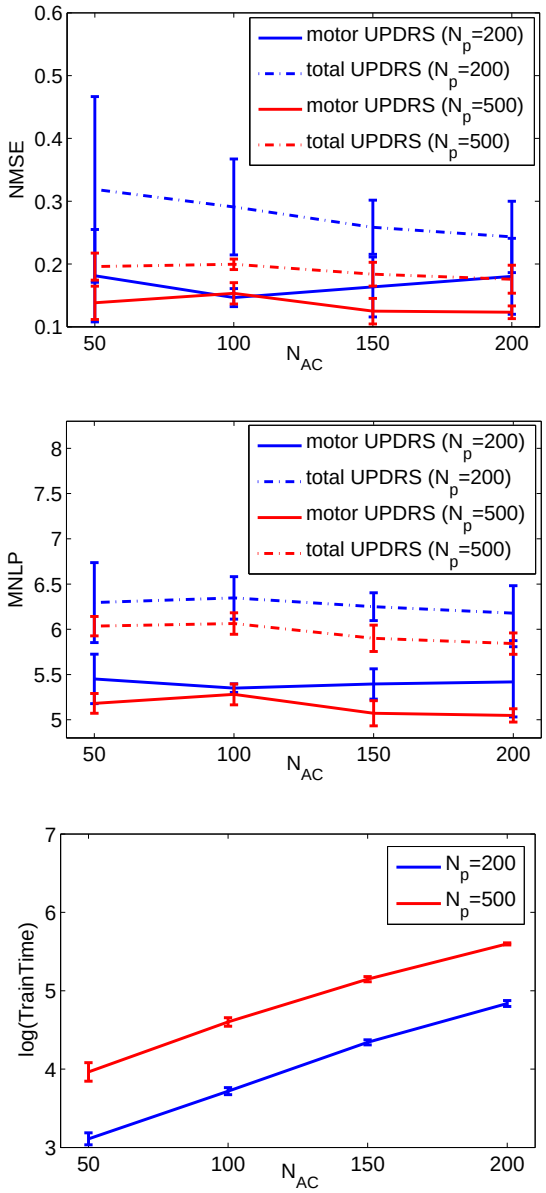


FIGURE 5.17: Predictive performance of our DGP as a function of N_{AC} and N_p (parkinsons data).

Table 5.3: Average prediction error & training efficiency comparison for multi-output flu data.

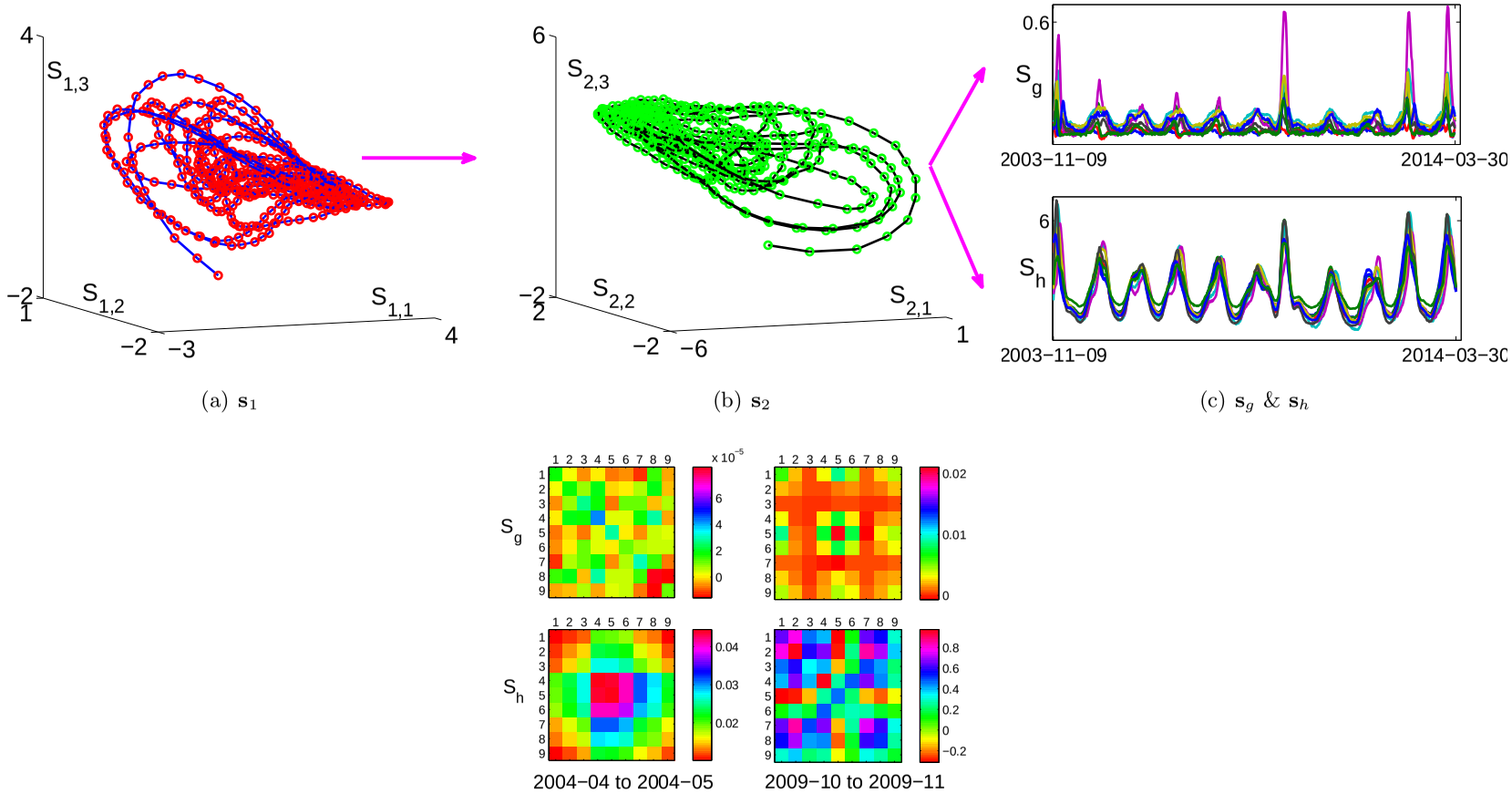
Flu	NMSE	MNLP	TrainTime(s)
MulGP	0.688	6.15	121
DGP	0.692	3.73	164
our DGP	**0.278**	**3.57**	**116**
our HDGP	**0.288**	**3.42**	210

one province) at two flu outbreak periods 2004-04 to 2004-05 (H7N3, bird flu) and 2009-10 to 2009-11 (H1N1, swine flu). Our HDGP successfully captures both types of correlations which are temporal-dependent (as the input is the time step).

Secondly, we assess our HDGP for missing data. In this experiment, we use the time period from 2007-09 to 2010-07 which contains the outbreak period of swine flu. There are total 150 data points where we consider the outputs of the first five provinces are missing from the 80th to the 100th data point, the outputs of the rest four provinces are missing from the 30th to the 50th data point. Similarly to parkinsons data, we compare our HDGP to independent GP (IGP) and MulGP in (Bonilla et al., 2007). The average predictive NMSE / MNLP for recovering the missing data are respectively 1.074 / 3.86 for IGP, 0.593 / 2.95 for MulGP and **0.556** / **2.73** for our HDGP. We can see that our HDGP outperforms both methods.

Thirdly, we use the same 150 data points from 2007-09 to 2010-07 to show the average accuracy and efficiency of our HDGP. We randomly select 50 points for training, the rest 100 points for test. Similar to the parkinsons data, the flu data is also multi-output. Hence we compare our DGP and HDGP to multi-output GP variants including MulGP (Bonilla et al., 2007) and DGP (Damianou and Lawrence, 2013) where N_p of our DGP and HDGP is 500 ; N_{AC} of DGP (Damianou and Lawrence, 2013), our DGP and our HDGP is 50 ; DGP Damianou and Lawrence (2013) is the same three-layer structure as our DGP. The results in Table 5.3 show both our DGP and HDGP outperform DGP in (Damianou and Lawrence, 2013).

Finally, we explore the robustness of our sequential inference framework for large data sets when varying the size of deep network structure. To this end, we perform our DGP and DGP in (Damianou and Lawrence, 2013) on the whole flu data (500 data pairs for training, 43 for test) where N_p of our DGP is 500 ; N_{AC} of DGP in (Damianou and Lawrence, 2013) and our DGP is 100. Table 5.4 shows that our sequential inference framework for DGP structure is more accurate and efficient than the inference framework in (Damianou and Lawrence, 2013).



(a) $\mathbf{s}_1$ (b) $\mathbf{s}_2$ (c) $\mathbf{s}_g$ & $\mathbf{s}_h$

(d) Correlations

FIGURE 5.18: Posterior Evaluation Layer by Layer (Flu Data). In Subplot 5.18(c), each dimension of $\mathbf{s}_g$ & $\mathbf{s}_h$ is plotted with respect to the time step. In Subplot 5.18(d), the 1st row shows the correlations between all the dimensions of $\mathbf{s}_g$ at the periods 2004-04 to 2004-05 and 2009-10 to 2009-11. The 2nd row shows the correlations between all the dimensions of $\mathbf{s}_h$ at the same periods.

Table 5.4: Average predictive performance of different DGP structures for multi-output flu data. (i/j) : DGP structure is i layers/j-dimension in each layer.

Flu	NMSE	MNLP	TrainTime(s)
DGP (3/3)	**1.1267**	9.5236	1924
DGP (3/5)	1.0933	8.3952	2183
DGP (5/3)	1.1353	8.7463	2725
DGP (5/5)	1.0748	7.9243	3769
our DGP (3/3)	**1.1269**	**5.5304**	**444**
our DGP (3/5)	**1.0156**	**5.3836**	**549**
our DGP (5/3)	**1.1317**	**5.4668**	**704**
our DGP (5/5)	**0.9670**	**5.4140**	**864**

5.2.5 Summary

In Wang and Chaib-draa (2015b), we have proposed a heteroscedastic deep GP (HDGP) which is a generalized version of heteroscedastic GP and deep GP. By performing *Dynamical Interaction Between Gaussian Processes and Bayesian Estimation*, we designed a sequential sampling inference framework for our HDGP to capture input-dependent correlations between outputs. Additionally, our HDGP can straightforwardly deal with missing data for multi-output regression. Our experiments have shown that our HDGP outperforms several benchmark GP variants.

5.3 Conclusion

In this chapter, we mainly work on *Dynamical Interaction Between Gaussian Processes and Bayesian Estimation* to address two challenging tasks that potentially exist in both GPs and Bayesian estimation.

Concretely, online GP particle filters (Wang et al., 2014) provide us with a novel approach for both modeling (GP) and tracking (Bayesian Estimation) where a GP dynamical model can be refined online during state tracking. The multi-modality is successfully captured by the GP mixture model representation in which we propose two fast online GP models for each mixture component to maintain the computation load. Our human motion tracking experiments have shown that our resulting framework can interpret the different types of human motion efficiently and accurately.

The second contribution in this chapter is a heteroscedastic deep GP (HDGP) model (Wang and Chaib-draa, 2015b) in which a deep GP structure is shared between signal and noise at the observation level. The resulting network structure does not only capture the input-dependent signal correlations between multi-output variables (inheriting from the deep GP (Damianou and Lawrence, 2013)) , but it also models the input-dependent noise correlations between

multi-output variables (explicitly establishing a deep GP structure for noise). Furthermore, inspired by *Dynamical Interaction Between Gaussian Processes and Bayesian Estimation*, we propose a sequential inference mechanism for our HDGP so that our HDGP can be implemented with high accuracy and efficiency.

In the next chapter, we conclude the thesis and propose some potential future works.

Chapitre 6

Conclusion

In this thesis, we worked on the problems of model learning and state estimation from a Bayesian perspective. By investigating *Interactions Between Gaussian Processes and Bayesian Estimation* which are based on the connections between Bayesian model (Gaussian processes) and Bayesian estimators (Kalman filter and Monte Carlo methods) in different directions, we demonstrated our resulting contributions can address a number of difficulties in modeling and estimation tasks with high efficiency and accuracy. In this chapter, we conclude this thesis with a summary of our contributions and an overview of future work.

6.1 Summary of the Contributions

In the following, we briefly summarize our contributions from three main aspects :

- *Gaussian Processes for Bayesian Estimation*
- *Bayesian Estimation for Gaussian Processes*
- *Dynamical Interaction Between Gaussian Processes and Bayesian Estimation*

6.1.1 Gaussian Processes for Bayesian Estimation

In *Gaussian Processes for Bayesian Estimation*, we mainly addressed the limitations in Bayesian estimation where the system model lacks in flexibility and/or is partially unknown. Due to the fact that parametric models are often insufficient in capturing the underlying flexibility of system models, we investigated on Gaussian processes (GPs) which are an elegant Bayesian nonparametric models. By incorporating GP models into Bayesian estimation approaches, we demonstrated that the estimation performance can be significantly improved.

In Wang and Chaib-draa (2012b), we proposed a novel adaptive nonparametric particle filter. Firstly, the particles, which are drawn from a learned proposal based on GP, are more likely located at the high probabilistic regions of the true posterior. Then, we combined this GP based proposal with a KLD-Sampling particle filter where the number of the particles is

adaptively learned. Our resulting framework improved the estimation performance with the adaptively-learned number of high-qualify particles. In the future, it would be interesting to further explore the sensitivity of model parameters (such as ε and δ that are used to learn the number of particles), and also show the effectiveness of our framework for mobile robot localization that is a fundamental task in autonomous robotics (Fox, 2001; Thrun et al., 2005).

In Wang and Barber (2014), we proposed a GP based Bayesian parameter estimation approach for ordinary differential equations (ODEs). In general, Bayesian numerical integration is a suitable way for parameter estimation in ODEs. However, explicit numerical integration is computationally prohibitive, even for small systems (Calderhead et al., 2008; Dondelinger et al., 2013). Several GP based approaches (Calderhead et al., 2008; Dondelinger et al., 2013) have been investigated to alleviate the computational burden, while these approaches are not based on natural generative models of the data and thus the estimation accuracy is limited. On the contrary, our GP-ODE model has a natural link to numerical integration by directly connecting GP derivatives to the observations, and is conceptually the closest match amongst competing GP approaches to Bayesian numerical integration. Since our GP-ODE model is a simple generative model, in the future it would be interesting to investigate alternative efficient approximation techniques other than Markov Chain Monte Carlo. For example, variational approximations are in principle possible to apply directly to the posterior (Bishop, 2006; Barber, 2012).

6.1.2 Bayesian Estimation for Gaussian Processes

In *Bayesian Estimation for Gaussian Processes*, we addressed a number of limitations in GPs. We investigated how to design an efficient and accurate estimation framework for GPs so that many challenging data properties can be correctly and flexibly modeled. Inspired by the strong connections between GPs and Kalman filter (Csató and Opper, 2002; Reece and Roberts, 2010; Vaerenbergh et al., 2012), we developed several computationally efficient and accurate Bayesian filtering frameworks for GPs.

In Wang and Chaib-draa (2012a), we proposed a novel marginalized particle filtering framework for GP regression. The small training set at each filtering iteration reduces the computational burden, while estimation performance is improved over iterations due to the fact that recursive filtering propagates the previous estimation to enhance the current estimation. Additionally, it provides us with a new online approach to learn GP hyperparameters. In the future, it would be interesting to show the effectiveness of our framework for non-Gaussian observations (such as student-t distribution for outlier robustness) due to the fact that the particle filter is an efficient and accurate technique for nonlinear and non-Gaussian systems (Doucet et al., 2000b; Cappé et al., 2007).

In Wang and Chaib-draa (2013), we proposed a novel KNN-KFGP for regression. First, small

training subsets, which are constructed by a test-driven KNN search, are used to establish a GP prior based state space model. Then a Kalman filter is applied to infer the latent function values in a computationally efficient predict-update manner. Our resulting framework is suitable to capture nonstationarities due to the fact that the KNN search assigns each test with its strongly correlated training subset. In the future, we will investigate online hyperparameter learning for our KNN-KFGP (instead of offline learning) to improve the flexibility of our model to process data streams.

In Wang and Chaib-draa (2015a), we proposed two particle based GP approaches for time-varying applications. By designing two novel state space models (SSMs), we used particle learning (PL) and Rao-Blackwellized particle filter (RBPF) to capture the time-varying non-stationarity and heteroscedasticity in a recursive Bayesian framework. Additionally, as in our RBPF-GP the hyperparameters are learned online, our RBPF-GP is more accurate than PL-GP (especially in multi-dimensional input cases) but the tradeoff is our PL-GP is more computationally efficient than our RBPF-GP. In the future, it would be interesting to incorporate other particle smoothing methods (for example, in Fearnhead et al. (2010) the smoothing computation is reduced to $O(N_p)$ where N_p is the number of the particles) to further improve the computational efficiency of our approaches.

6.1.3 Dynamical Interaction Between Gaussian Processes and Bayesian Estimation

In *Dynamical Interaction Between Gaussian Processes and Bayesian Estimation*, we mainly work on how to overcome the potential difficulties in both GP modeling and Bayesian estimation since model learning and state estimation are strongly correlated in many real-world applications. This tells us that, if we plan to achieve acceptable performances from either the learning or estimation perspective in these applications, we have to handle the challenges from both perspectives.

In Wang et al. (2014), we proposed a novel online GP particle filter to address modeling and tracking tasks for human motions. By using the interaction between GP modeling and particle filtering, a GP dynamical model is updated in an online fashion after filtering in order to improve the tracking performance. Moreover, our online GP particle filter can successfully capture multi-modality since the GP dynamical model is represented by GP mixture in which we explored two fast GP variants, sparse online GP and local GP, to maintain an acceptable computation efficiency. Finally, we demonstrated the effectiveness of our approach when tracking different human motions and explored the impact of various parameters on performance. In the future, we plan to implement our approaches for 3-D human motion tracking.

In Wang and Chaib-draa (2015b), we proposed a heteroscedastic deep GP (HDGP) which is a generalized version of heteroscedastic GP and deep GP. By designing a sequential sampling

inference framework, our HDGP successfully captures input-dependent signal and noise correlations between outputs. Furthermore, it is worth mentioning that our HDGP can straightforwardly deal with missing data for multi-output regression. Finally, note that the representational capacity of the deep network tends to capture fewer degrees of freedom as the number of layer increases (Duvenaud et al., 2014), it would be interesting to design an input-connected structure (inspired by Duvenaud et al. (2014)) for our HDGP to further improve the power of model interpretation in the future.

6.2 Future Research

Interactions Between Gaussian Processes and Bayesian Estimation in this thesis open a door to take advantage of the connections of Gaussian Processes and Bayesian Estimation in different directions to address the potential difficulties in Bayesian modeling and state estimation. In the following we briefly outline three main future directions for research.

Gaussian Processes with Student-t Noise

In a standard Gaussian process (GP) setting, the distribution of the observation noise is assumed to be Gaussian in order to make inference analytically tractable . However, in many real-life applications the observations often contain a number of outliers, and this fact will heavily deteriorate the performance of GPs (Rasmussen and Williams, 2006; Vanhatalo et al., 2009). The student-t distribution is a popular choice to achieve outlier-robustness (Rasmussen and Williams, 2006; Vanhatalo et al., 2009). Hence one can work on Gaussian Processes with Student-t noise (or more generally, the observation noise could be any other non-Gaussian distributions) based on our particle filtering based GP approaches (Wang and Chaib-draa, 2012a, 2015a). Technically, this can be done without difficulties since the particle filter is an efficient and accurate Bayesian estimation approach for nonlinear and non-Gaussian systems (Doucet et al., 2000b; Cappé et al., 2007).

Input-connected Deep Gaussian Processes

The recent developments in deep learning (Hinton et al., 2006; Larochelle et al., 2009; Bengio et al., 2013) brings neural network models again in popularity. Similar to a standard deep Gaussian processes model (Damianou and Lawrence, 2013), our heteroscedastic deep Gaussian process is a type of infinitely-wide, deep neural network. However, as mentioned in Duvenaud et al. (2014), the representational capacity of the network tends to capture fewer degrees of freedom as the number of layer increases. Duvenaud et al. (2014) proposed an input-connected architecture to increase the model flexibility where the outputs of each layer does not only depend on the outputs of the previous layer, but they also depend on the original input. Inspired by this idea, one can directly incorporate this input-connected structure into our heteroscedastic deep GP to further

improve the model representation. Additionally, our sequential inference framework will still work without difficulties due to its particle-based approximate mechanism.

Applications in Robotics and Computer Vision

Most parts of this thesis can be applied for machine learning and/or artificial intelligent applications. That is why it would be interesting to work on real applications in Robotics and Computer Vision. For example, our adaptive nonparametric particle filter (Wang and Chaib-draa, 2012b) could be implemented for mobile robot localization, and our online GP particle filter (Wang et al., 2014) could be used in real-world 3-D human motion tracking.

Bibliographie

Adams, R. P. and Stegle, O. (2008). Gaussian Process Product Models for Nonparametric Nonstationarity. In *International Conference on Machine Learning (ICML)*, pages 1–8.

Alvarez, M. and Lawrence, N. D. (2008). Sparse Convolved Gaussian Processes for Multi-output Regression. In *Neural Information Processing Systems (NIPS)*, pages 57–64.

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. (2003). An Introduction to MCMC for Machine Learning. *Machine Learning*, 50 :3–43.

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle Markov Chain Monte Carlo Methods. *Journal of the Royal Statistical Society : Series B*, 72(3) :269–342.

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

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation Learning : A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8) :1798–1828.

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

Bonilla, E. V., Chai, K. M. A., and Williams, C. K. I. (2007). Multi-task Gaussian Process Prediction. In *Neural Information Processing Systems (NIPS)*, pages 153–160.

Boyle, P. and Frean, M. (2004). Dependent Gaussian Processes. In *Neural Information Processing Systems (NIPS)*, pages 217–224.

Calderhead, B., Girolami, M., and Lawrence, N. D. (2008). Accelerating Bayesian Inference over Nonlinear Differential Equations with Gaussian Processes. In *Neural Information Processing Systems (NIPS)*, pages 217–224.

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

Carvalho, C. M., Johannes, M. S., Lopes, H. F., and Polson, N. G. (2010). Particle Learning and Smoothing. *Statistical Science*, 25(1) :88–106.

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

Chen, Z. (2003). Bayesian Filtering : From Kalman Filters to Particle Filters, and Beyond. *Statistics*, 182 :1–69.

Chopin, N. (2002). A Sequential Particle Filter Method for Static Models. *Biometrika*, 89(3) :539–551.

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

Damianou, A. C. and Lawrence, N. D. (2013). Deep Gaussian Processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 207–215.

de Freitas, N. (2002). Rao-Blackwellised particle filtering for fault diagnosis. In *IEEE Aerospace Conference Proceedings*, pages 1767–1772.

Deisenroth, M. P. (2010). *Efficient Reinforcement Learning Using Gaussian Processes*. PhD thesis, Karlsruhe Institute of Technology.

Deisenroth, M. P., Huber, M. F., and Hanebeck, U. D. (2009). Analytic Moment-based Gaussian Process Filtering. In *International Conference on Machine Learning (ICML)*, pages 225–232.

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

Dondelinger, F., Filippone, M., Rogers, S., and Husmeier, D. (2013). ODE Parameter Inference using Adaptive Gradient Matching with Gaussian Processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 216–228.

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

Doucet, A., de Freitas, N., Murphy, K., and Russell, S. (2000a). Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In *Uncertainty in Artificial Intelligence (UAI)*, pages 176–183.

Doucet, A., Godsill, S., and Andrieu, C. (2000b). On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and Computing*, 10 :197–208.

Duvenaud, D., Rippel, O., Adams, R. P., and Ghahramani, Z. (2014). Avoiding Pathologies in Very Deep Networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 202–210.

Fearnhead, P., Wyncoll, D., and Tawn, J. (2010). A Sequential Smoothing Algorithm with Linear Computational Cost. *Biometrika*, 97(2) :447–464.

Fox, D. (2001). KLD-Sampling : Adaptive Particle Filters. In *Neural Information Processing Systems (NIPS)*, pages 713–720.

Geman, S. and Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6 :721–741.

Gilks, W., Richardson, S., and Spiegelhalter, D. (1996). *Markov Chain Monte Carlo in Practice*. Chapman and Hall/CRC.

Godsill, S. J., Doucet, A., and West, M. (2004). Monte Carlo Smoothing for Nonlinear Time Series. *Journal of the American Statistical Association*, 99(465) :156–168.

Goldberg, P. W., Williams, C. K. I., and Bishop, C. M. (1998). Regression with Input-Dependent Noise : A Gaussian Process Treatment. In *Neural Information Processing Systems (NIPS)*, pages 493–499.

Hartikainen, J. and Särkkä, S. (2010). Kalman Filtering and Smoothing Solutions to Temporal Gaussian Process Regression Models. In *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 379–384.

Hastings, W. K. (1970). Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57 :97–109.

Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A Fast Learning Algorithm For Deep Belief Networks. *Neural Computation*, 18(7) :1527–1554.

Jazwinski, A. (1970). *Stochastic Processes and Filtering Theory*. Academic Press.

Jenkins, O. C. and Matarić, M. (2004). A Spatio-Temporal Extension to Isomap Nonlinear Dimension Reduction. In *International Conference on Machine Learning (ICML)*, pages 441–448.

Julier, S. J., Uhlmann, J. K., and Durrant-Whyte, H. F. (1995). A New Approach for Filtering Nonlinear Systems. In *American Control Conference*, pages 1628–1632.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME-Journal of basic Engineering*, 82(Series D) :35–45.

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

Kersting, K., Plagemann, C., Pfaff, P., and Burgard, W. (2007). Most Likely Heteroscedastic Gaussian Process Regression. In *International Conference on Machine Learning (ICML)*, pages 393–400.

Ko, J. (2011). *Gaussian Processes for Dynamic Systems*. PhD thesis, University of Washigton.

Ko, J. and Fox, D. (2008). GP-BayesFilters : Bayesian Filtering Using Gaussian Process Prediction and Observation Models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3471–3476.

Ko, J. and Fox, D. (2009). Learning GP-BayesFilters via Gaussian Process Latent Variable Models. In *Robotics : Science and Systems (RSS)*.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models : Principles and Technique*. MIT Press.

Kuss, M. (2006). *Gaussian Process Models for Robust Regression, Classification, and Reinforcement Learning*. PhD thesis, Technische Universität Darmstadt.

Larochelle, H., Bengio, Y., Louradour, J., and Lamblin, P. (2009). Exploring Strategies for Training Deep Neural Networks. *Journal of Machine Learning Research*, 1 :1–40.

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. (2012). Bayesian Warped Gaussian Processes. In *Neural Information Processing Systems (NIPS)*, pages 1619–1627.

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

Lázaro-Gredilla, M. and Titsias, M. K. (2011). Variational Heteroscedastic Gaussian Process Regression. In *International Conference on Machine Learning (ICML)*, pages 841–848.

Lee, C.-S. and Elgammal, A. (2010). Coupled Visual and Kinematics Manifold Models for Tracking. *International Journal of Computer Vision*, 87 :118–139.

Levine, S., Wang, J., Haraux, A., Popovic, Z., and Koltun, V. (2012). Continuous Character Control with Low-Dimensional Embeddings. In *SIGGRAPH*, page 28.

Li, P., Goodall, R., and 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. and West, M. (2001). Combined Parameter and State Estimation in Simulation-Based Filtering. In *Sequential Monte Carlo Methods in Practice*, pages 197–223.

MacKay, D. J. C. (1998). Introduction to Gaussian Processes. In *Neural Networks and Machine Learning*, pages 133–165.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21 :1087–1091.

Moon, K. and Pavlovic, V. (2006). Impact of Dynamics on Subspace Embedding and Tracking of Sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 198–205.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer-Verlag.

Neal, R. M. (1997). Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification. Technical report, Department of Statistics, University of Toronto.

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

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

Plagemann, C. (2008). *Gaussian Processes for Flexible Robot Learning*. PhD thesis, Albert-Ludwigs-Universität Freiburg.

Plagemann, C., Fox, D., and Burgard, W. (2007). Efficient Failure Detection on Mobile Robots Using Particle Filters with Gaussian Process Proposals. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2185–2190.

Quinonero-Candela, J. and Rasmussen, C. E. (2005). A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research*, 6 :1939–1959.

Ramsay, J., G.Hooker, Campbell, D., and Cao, J. (2007). Parameter Estimation for Differential Equations : A Generalized Smoothing Approach. *Journal of the Royal Statistical Society : Series B*, 69(5) :741–796.

Rasmussen, C. E. and Ghahramani, Z. (2001). Infinite Mixtures of Gaussian Process Experts. In *Neural Information Processing Systems (NIPS)*, pages 881–888.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

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

Roweis, S. and Saul, L. (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500) :2323–2326.

Schmidt, A. M. and O'Hagan, A. (2003). Bayesian Inference for Non-stationary Spatial Covariance Structure via Spatial Deformations. *Journal of the Royal Statistical Society : Series B*, 65(3) :743 – 758.

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

Seeger, M., Williams, C. K. I., and Lawrence, N. D. (2003). Fast Forward Selection to Speed Up Sparse Gaussian Process Regression. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 643–650.

Smola, A. J. and Bartlett, P. (2000). Sparse Greedy Gaussian Process Regression. In *Neural Information Processing Systems (NIPS)*, pages 619–625.

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

Snelson, E., Rasmussen, C. E., and Ghahramani, Z. (2003). Warped Gaussian Processes. In *Neural Information Processing Systems (NIPS)*, pages 337–344.

Solak, E., Murray-Smith, R., Leithead, W., Leith, D., and Rasmussen, C. (2002). Derivative Observations in Gaussian Process Models of Dynamic Systems. In *Neural Information Processing Systems (NIPS)*, pages 1057–1064.

Storvik, G. (2002). Particle Filters for State-Space Models With the Presence of Unknown Static Parameters. *IEEE Transactions on Signal Processing*, 50(2) :281–289.

Tenenbaum, J., de Silva, V., and Langford, J. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, pages 2319–2323.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. MIT Press.

Tresp, V. (2000). A Bayesian Committee Machine. *Neural Computation*, 12 :2719 – 2741.

Turner, R., Deisenroth, M. P., and Rasmussen, C. E. (2010). State-Space Inference and Learning with Gaussian Processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 868–875.

Turner, R. and Rasmussen, C. E. (2012). Model Based Learning of Sigma Points in Unscented Kalman Filtering. *Neurocomputing*, 80 :47–53.

Urtasun, R. and Darrell, T. (2008). Sparse Probabilistic Regression for Activity-independent Human Pose Inference. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8.

Urtasun, R., Fleet, D., and Fua, P. (2006). 3D People Tracking with Gaussian Process Dynamical Models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 238–245.

Urtasun, R., Fleet, D., Geiger, A., Popovic, J., Darrell, T., and Lawrence, N. (2008). Topologically-Constrained Latent Variable Models. In *International Conference on Machine Learning (ICML)*, pages 1080–1087.

Urtasun, R., Fleet, D., Hertzman, A., and Fua, P. (2005). Priors for People Tracking from Small Training Sets. In *IEEE International Conference on Computer Vision (ICCV)*, pages 403–410.

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

van der Merwe, R., Doucet, A., de Freitas, N., and Wan, E. (2000). Unscented Particle Filter. Technical report, Engineering Department, Cambridge University.

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

Vyshemirsky, V. and Girolami, M. (2008). Bayesian Ranking of Biochemical System Models. *Bioinformatics*, 24 :833–839.

Wang, J., Fleet, D., and Hertzmann, A. (2008). Gaussian Process Dynamical Models for Human Motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2) :283–298.

Wang, Y. and Barber, D. (2014). Gaussian Processes for Bayesian Estimation in Ordinary Differential Equations. In *International Conference on Machine Learning (ICML)*, pages 1485–1493.

Wang, Y., Brubaker, M. A., Chaib-draa, B., and Urtasun, R. (2014). Bayesian Filtering with Online Gaussian Process Latent Variable Models. In *Uncertainty in Artificial Intelligence (UAI)*, pages 849–857.

Wang, Y. and Chaib-draa, B. (2012a). A Marginalized Particle Gaussian Process Regression. In *Neural Information Processing Systems (NIPS)*, pages 1196–1204.

Wang, Y. and Chaib-draa, B. (2012b). An Adaptive Nonparametric Particle Filter for State Estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4355–4360.

Wang, Y. and Chaib-draa, B. (2013). A KNN based Kalman Filter Gaussian Process Regression. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1771–1777.

Wang, Y. and Chaib-draa, B. (2015a). Bayesian Inference for Time-Varying Applications : Particle Based Gaussian Process Approaches. In *Submission to AAAI Conference on Artificial Intelligence (AAAI)*.

Wang, Y. and Chaib-draa, B. (2015b). Efficient Sequential Inference for Heteroscedastic Deep Gaussian Processes Regression. In *Submission to International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Wilson, A. G., Knowles, D. A., and Ghahramani, Z. (2012). Gaussian Process Regression Networks. In *International Conference on Machine Learning (ICML)*, pages 599–606.

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

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