Pattern Recognition Letters 45 (2014) 85–91

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Recursive Gaussian process: On-line regression and learning ☆

Marco F. Huber *

AGT International, Hilpertstr. 35, Darmstadt, Germany

ARTICLE INFO

Article history:
Received 6 December 2013
Available online 18 March 2014

Keywords:
Gaussian processes
Parameter learning
Kalman filtering

ABSTRACT

Two approaches for on-line Gaussian process regression with low computational and memory demands are proposed. The first approach assumes known hyperparameters and performs regression on a set of basis vectors that stores mean and covariance estimates of the latent function. The second approach additionally learns the hyperparameters on-line. For this purpose, techniques from nonlinear Gaussian state estimation are exploited. The proposed approaches are compared to state-of-the-art sparse Gaussian process algorithms.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

Gaussian processes (GPs) allow non-parametric learning of regression functions from noisy data and can be considered Gaussian distributions over functions conditioned on the data [13]. Unfortunately, due to their non-parametric nature, GPs require computations that scale with $\mathcal{O}(n^3)$ for training, where n is the number of data points.

In order to reduce the computational load, sparse approximations have been proposed in the recent years. In [12] a unifying framework for so-called *active set* approaches has been derived. Here, instead of processing the entire training data set, only a subset of the data points—the active set with $s \ll n$ data points—is used. This framework comprises for instance the subset of regressors [16], sparse on-line GP (SOGP, [2]), or sparse pseudo-input GP (SPGP, [17]). Thanks to the sparse representation, the computational load is reduced to $\mathcal{O}(s^2 \cdot n)$ or even to $\mathcal{O}(s^3)$ (see the approach proposed in [4]).

GP regression can also be sped up by partitioning the training data into separate data sets, where for each data set a separate GP is learned (see e.g., [19,11]). For calculating the GP prediction, the results of the separate GPs are combined. In contrast to active set methods, partitioning approaches make use of the entire training data.

Most of the above approximations assume that the whole data set is available a priori and thus, training can be performed *off-line* in a batch mode. Only a few sparse approaches have been proposed that allow sequential training of GPs for data that arrives *on-line*, i.e., streaming data. In [2] for instance, a score value is assigned to each element of the active set. If a new data point arrives, it is added to the active set, while an element with the lowest score is eliminated. For specific kernel functions, the approach proposed in [3] transforms GP regression into a Kalman state estimation problem that merely scales with $\mathcal{O}(n)$. Unfortunately, this approach so far is only applicable for one-dimensional inputs.

The approaches proposed in this paper allow for both a sparse representation *and* on-line processing. For this purpose, the regression function is represented by means of a finite set of *basis vectors*. Training with incoming data, i.e., updating mean and covariance estimates featured by the basis vectors (Section 3) as well as simultaneously learning hyperparameters (Section 4), is performed recursively via Bayesian state estimation techniques. After updating the newly arrived data points can be discarded, while the joint Gaussian state estimate of the regression function and the hyperparameters is sufficient for prediction.

2. Problem formulation

For GP regression, it is assumed that a set of data $\mathcal{D} = \{(\underline{x}_1, y_1), \ldots, (\underline{x}_n, y_n)\}$ is drawn from the noisy process

$$y_i = g(\underline{x}_i) + \epsilon, \tag{1}$$

where $\underline{x}_i \in \mathbb{R}^d$ are the inputs, $y_i \in \mathbb{R}$ are the observations or outputs, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is zero-mean Gaussian noise with variance σ^2. For brevity reasons, $\mathbf{X}_\mathcal{D} = [\underline{x}_1, \ldots, \underline{x}_n]$ are all inputs and $\underline{y} = [y_1, \ldots, y_n]^\mathrm{T}$ are the corresponding observations in the following.

A GP is used to infer the latent function $g(\cdot)$ from the data $\mathcal{D}$. The GP is completely defined by a mean function $m(\underline{x}) \triangleq \mathrm{E}\{g(\underline{x})\}$ specifying the expected output value, and a positive semi-definite covariance function $k(\underline{x}, \underline{x}') \triangleq \mathrm{cov}\{g(\underline{x}), g(\underline{x}')\}$, which specifies the

☆ This paper has been recommended for acceptance by S. Todorovic.
* Tel.: +49 151 460 5170; fax: +49 6151 460 5190.
E-mail address: marco.huber@ieee.org

http://dx.doi.org/10.1016/j.patrec.2014.03.004
0167-8655/© 2014 Elsevier B.V. All rights reserved.

covariance between pairs of inputs and is often called a *kernel*. Typical examples are the zero mean function $m(\underline{x}) = 0$ and the squared exponential (SE) kernel

$$k(\underline{x}, \underline{x}') = \alpha^2 \cdot \exp\left(-\frac{1}{2}(\underline{x} - \underline{x}')^{\mathrm{T}} \mathbf{\Lambda}^{-1} (\underline{x} - \underline{x}')\right). \tag{2}$$

In (2) $\mathbf{\Lambda} = \mathrm{diag}(l_1, l_2, \ldots, l_d)$ is a diagonal matrix of the characteristic length-scales l_i for each input dimension and α^2 is the variance of the latent function g. Such parameters of the mean and covariance functions together with the noise standard deviation σ are called the *hyperparameters* of the GP. In the following, all hyperparameters are collected in the vector $\underline{\eta} \in \mathbb{R}^r$, e.g., $\underline{\eta} = [\alpha, l_1, \ldots, l_d, \sigma]^{\mathrm{T}}$ comprising the parameters of the SE kernel (2) and the noise standard deviation. It is worth mentioning that the approach proposed in this paper holds for *arbitrary* mean and covariance functions.

For any finite set of inputs a GP provides a multivariate Gaussian distribution of the outputs. For example, the distribution of the function value $g_* = g(\underline{x}_*)$ for an arbitrary test input $\underline{x}_*$ is a univariate Gaussian with mean and variance

$$\begin{aligned} \mu_g(\underline{x}_*) &= \mathrm{E}\{g_*\} = m_* + \underline{k}_*^{\mathrm{T}} \mathbf{K}_x^{-1} \left(\underline{y} - \underline{m}\right), \\ \sigma_g^2(\underline{x}_*) &= \mathrm{var}\{g_*\} = k_{**} - \underline{k}_*^{\mathrm{T}} \mathbf{K}_x^{-1} \underline{k}_*, \end{aligned} \tag{3}$$

respectively. Here, $\mathrm{var}\{\cdot\}$ is the variance, $\mathbf{K}_x \triangleq \mathbf{K} + \sigma^2 \mathbf{I}$, $m_* \triangleq m(\underline{x}_*)$, $\underline{m} \triangleq m(\mathbf{X}_{\mathcal{D}}), \underline{k}_* \triangleq k(\mathbf{X}_{\mathcal{D}}, \underline{x}_*), k_{**} \triangleq k(\underline{x}_*, \underline{x}_*)$, and $\mathbf{K} \triangleq k(\mathbf{X}_{\mathcal{D}}, \mathbf{X}_{\mathcal{D}})$ is the kernel matrix.

For GP prediction, i.e., for calculating the distribution for a given set of test inputs according to (3), it is necessary to calculate the kernel matrix $\mathbf{K}$, to invert the matrix $\mathbf{K}_x$, and to multiply $\mathbf{K}_x$ with $\underline{k}_*$. Both the kernel matrix calculation and the multiplication scale with $\mathcal{O}(n^2)$, while the inversion even scales with $\mathcal{O}(n^3)$. Thus, for large data sets $\mathcal{D}$, storing the kernel matrix and solving all calculations is prohibitive. The following recursive GP approach aims at performing all calculations computationally very efficient on a set of $m \ll n$ so-called *basis vectors*.

3. On-line regression

We will now summarize our approach proposed in [5], which focuses on performing on-line regression given a set of m basis vectors. Let us assume that the hyperparameters are already known and thus, have not to be learned from data. This assumption will be avoided in Section 4. These basis vectors are located at $\mathbf{X} \triangleq [\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_m]$ and store local estimates $\underline{g} \triangleq g(\mathbf{X})$ of the latent function $g(\cdot)$. Thus, the basis vectors can be considered an active set allowing a sparse GP representation. In contrast to most other active set approaches, the basis vectors are updated *on-line* with new observations $\underline{y}_t$ at inputs $\mathbf{X}_t \triangleq [\underline{x}_{t,1}, \ldots, \underline{x}_{t,n_t}]$ and time step $t = 0, 1, \ldots$, which makes this approach well suited for streaming data. Also off-line processing is possible by presenting the data in $\mathcal{D}$ in batches to the algorithm.

For all steps $t = 0, 1, \ldots$ it assumed that the basis vectors are fixed in number and location. Since $g(\underline{x})$ is assumed to be a GP, the initial distribution $p_0(\underline{g}) = \mathcal{N}\left(\underline{g}; \underline{\mu}_0^g, \mathbf{C}_0^g\right)$ of $\underline{g}$ for $t = 0$ is Gaussian with mean $\underline{\mu}_0^g \triangleq m(\mathbf{X})$ and covariance $\mathbf{C}_0^g \triangleq k(\mathbf{X}, \mathbf{X})$.

The goal is now to calculate the posterior distribution $p\left(\underline{g}|\underline{y}_{1:t}\right)$, with $\underline{y}_{1:t} \triangleq \left(\underline{y}_1, \ldots, \underline{y}_t\right)$, recursively by updating the prior distribution of $\underline{g}$ from the previous step $t-1$

$$p\left(\underline{g}|\underline{y}_{1:t-1}\right) = \mathcal{N}\left(\underline{g}; \underline{\mu}_{t-1}^g, \mathbf{C}_{t-1}^g\right)$$

with the new observations $\underline{y}_t$.

One might think of exploiting (3) for incorporating the new observations. This however, is not suitable for recursive processing for mainly three reasons. First, (3) merely allow a prediction for given inputs and no incorporation of new information. Second, (3) operates directly on the data $\mathcal{D}$. To allow recursive processing with constant time and memory, not the data $\mathcal{D}$ but a distribution $p\left(\underline{g}|\underline{y}_{1:t-1}\right)$ sparsely representing $\mathcal{D}$ needs to processed. Third, no correlation or cross-covariance between $\mathbf{X}$ and $\mathbf{X}_t$ is provided, which however is of paramount importance for updating $p\left(\underline{g}|\underline{y}_{1:t-1}\right)$. Instead, for deriving a recursive algorithm, the desired posterior distribution is expanded according to

$$p\left(\underline{g}|\underline{y}_{1:t}\right) = \int \underbrace{c_t \cdot p\left(\underline{y}_t|\underline{g}, \underline{g}_t\right) \cdot \overbrace{p\left(\underline{g}_t|\underline{g}\right) \cdot p\left(\underline{g}|\underline{y}_{1:t-1}\right)}^{=p\left(\underline{g}, \underline{g}_t|\underline{y}_{1:t-1}\right) \text{ (inference)}}}_{=p\left(\underline{g}, \underline{g}_t|\underline{y}_{1:t}\right) \text{ (update)}} \mathrm{d}\underline{g}_t \tag{4}$$

in two processing steps: (*inference*) calculating the joint prior $p\left(\underline{g}, \underline{g}_t|\underline{y}_{1:t-1}\right)$ given the prior $p\left(\underline{g}|\underline{y}_{1:t-1}\right)$, which provides the required correlation information between $\mathbf{X}$ and $\mathbf{X}_t$, and (*update*) updating the joint prior with the observations $\underline{y}_t$. The second step follows from applying Bayes' law and integrating out $\underline{g}_t \triangleq g(\mathbf{X}_t)$, where c_t is a normalization constant. The integration is required for maintaining a constant number of basis vectors.

3.1. Inference

In order to determine the joint prior $p\left(\underline{g}, \underline{g}_t|\underline{y}_{1:t-1}\right)$, the chain rule for probability distribution is applied, which yields

$$\begin{aligned} p\left(\underline{g}, \underline{g}_t|\underline{y}_{1:t-1}\right) &= p\left(\underline{g}_t|\underline{g}\right) \cdot p\left(\underline{g}|\underline{y}_{1:t-1}\right) \\ &= \mathcal{N}\left(\underline{g}_t; \underline{\mu}_t^p, \mathbf{B}\right) \cdot \mathcal{N}\left(\underline{g}; \underline{\mu}_{t-1}^g, \mathbf{C}_{t-1}^g\right) \end{aligned} \tag{5}$$

with

$$\underline{\mu}_t^p \triangleq m(\mathbf{X}_t) + \mathbf{J}_t \cdot \left(\underline{\mu}_{t-1}^g - m(\mathbf{X})\right), \tag{6}$$

$$\mathbf{B} \triangleq k(\mathbf{X}_t, \mathbf{X}_t) - \mathbf{J}_t \cdot k(\mathbf{X}, \mathbf{X}_t), \tag{7}$$

$$\mathbf{J}_t \triangleq k(\mathbf{X}_t, \mathbf{X}) \cdot k(\mathbf{X}, \mathbf{X})^{-1}. \tag{8}$$

The first equality in (5) follows from assuming that $\underline{g}_t$ is conditionally independent of the past observations $\underline{y}_{1:t-1}$ given $\underline{g}$. As any finite representation of a GP is Gaussian, this also holds for the joint prior. Hence, the conditional distribution $p\left(\underline{g}_t|\underline{g}\right)$ is Gaussian as well and results from the joint prior by conditioning on $\underline{g}$ (see for example Chapter 2.6 in [8]), which results in the second equality.

After some algebraic transformations, where some basic properties of Gaussian distributions and the Woodbury formula is utilized, the product in (5) yields the joint Gaussian $p\left(\underline{g}, \underline{g}_t|\underline{y}_{1:t-1}\right) = \mathcal{N}\left(\underline{q}; \mathbf{Q}\right)$ of $\underline{g}$ and $\underline{g}_t$ with mean and covariance

$$\underline{q} \triangleq \begin{bmatrix} \underline{\mu}_{t-1}^g \\ \underline{\mu}_t^p \end{bmatrix} \quad \text{and} \quad \mathbf{Q} \triangleq \begin{bmatrix} \mathbf{C}_{t-1}^g & \mathbf{C}_{t-1}^g \mathbf{J}_t^{\mathrm{T}} \\ \mathbf{J}_t \mathbf{C}_{t-1}^g & \mathbf{C}_t^p \end{bmatrix}, \tag{9}$$

respectively, and with covariance $\mathbf{C}_t^p \triangleq \mathbf{B} + \mathbf{J}_t \mathbf{C}_{t-1}^g \mathbf{J}_t^{\mathrm{T}}$. This inference step coincides with the augmented Kalman Smoother proposed in [15], but there no update step for basis vectors as introduced next is derived.

3.2. Update

The next step is to perform the update and marginalization in (4). For this purpose, the joint prior $p\left(\underline{g}, \underline{g}_t | \underline{y}_{1:t-1}\right) = p\left(\underline{g}_t | \underline{y}_{1:t-1}\right) \cdot p\left(\underline{g} | \underline{g}_t, \underline{y}_{1:t-1}\right)$ is now factorized by conditioning on $\underline{g}_t$. Furthermore, the fact that $\underline{g}$ is not observed is utilized and thus, $p\left(\underline{y}_t | \underline{g}, \underline{g}_t\right) = p\left(\underline{y}_t | \underline{g}_t\right)$ is independent of $\underline{g}$. Since $p\left(\underline{y}_t | \underline{g}_t\right) = \mathcal{N}\left(\underline{y}_t; \underline{g}_t, \sigma^2 \mathbf{I}\right)$ according to (1) and $p\left(\underline{g}_t | \underline{y}_{1:t-1}\right) = \mathcal{N}\left(\underline{g}_t; \underline{\mu}_t^p, \mathbf{C}_t^p\right)$ according to (9) are both Gaussian, $\underline{g}_t$ can be updated easily via a *Kalman filter* update step. Updating $\underline{g}$ and integrating out $\underline{g}_t$ is then performed simultaneously.

Applying the well-known Kalman filter update yields $p\left(\underline{g}_t | \underline{y}_{1:t}\right) = \mathcal{N}\left(\underline{g}_t; \underline{\mu}_t^e, \mathbf{C}_t^e\right)$ with mean and covariance

$$\underline{\mu}_t^e \triangleq \underline{\mu}_t^p + \mathbf{G}_t \cdot \left(\underline{y}_t - \underline{\mu}_t^p\right),$$
$$\mathbf{C}_t^e \triangleq \mathbf{C}_t^p - \mathbf{G}_t \mathbf{C}_t^p,$$

respectively, where $\mathbf{G}_t \triangleq \mathbf{C}_t^p \cdot \left(\mathbf{C}_t^p + \sigma^2 \mathbf{I}\right)^{-1}$ is the Kalman gain. The multiplication of the two Gaussians $p\left(\underline{g}_t | \underline{y}_{1:t}\right)$ and $p\left(\underline{g} | \underline{g}_t, \underline{y}_{1:t-1}\right)$ again results in a joint Gaussian distribution of $\underline{g}$ and $\underline{g}_t$ with mean and covariance

$$\underline{\mu}_t = \begin{bmatrix} \underline{\mu}_t^g \\ \underline{\mu}_t^e \end{bmatrix} \quad \text{and} \quad \mathbf{C}_t = \begin{bmatrix} \mathbf{C}_t^g & \mathbf{L}_t \mathbf{C}_t^e \\ \mathbf{C}_t^e \mathbf{L}_t^{\mathrm{T}} & \mathbf{C}_t^e \end{bmatrix},$$

respectively, where $\mathbf{L}_t \triangleq \mathbf{C}_{t-1}^g \mathbf{J}_t^{\mathrm{T}} \left(\mathbf{C}_t^p\right)^{-1}$ and

$$\underline{\mu}_t^g = \underline{\mu}_{t-1}^g + \tilde{\mathbf{G}}_t \cdot \left(\underline{y}_t - \underline{\mu}_t^p\right), \tag{10}$$

$$\mathbf{C}_t^g = \mathbf{C}_{t-1}^g - \tilde{\mathbf{G}}_t \mathbf{J}_t \mathbf{C}_{t-1}^g, \tag{11}$$

$$\tilde{\mathbf{G}}_t = \mathbf{L}_t \cdot \mathbf{G}_t = \mathbf{C}_{t-1}^g \mathbf{J}_t^{\mathrm{T}} \cdot \left(\mathbf{C}_t^p + \sigma^2 \mathbf{I}\right)^{-1}. \tag{12}$$

Since we are merely interested in obtaining the distribution $p\left(\underline{g} | \underline{y}_{1:t}\right) = \mathcal{N}\left(\underline{g}; \underline{\mu}_t^g, \mathbf{C}_t^g\right)$, i.e., updating the latent function at the basis vectors $\mathbf{X}$ in order to keep the memory and computational complexity bounded over time, $\underline{g}_t$ is integrated out. This corresponds to neglecting the mean $\underline{\mu}_t^e$ and covariance $\mathbf{C}_t^e$ of $\underline{g}_t$ as well as the cross-covariance $\mathbf{L}_t \mathbf{C}_t^e$.

Putting all together, at steps $t = 1, 2, \ldots$ the proposed approach named *recursive GP* (RGP) recursively processes observations $\underline{y}_t$ at the inputs $\mathbf{X}_t$ as listed in Algorithm 1. This recursion commences from the initial mean $\underline{\mu}_0^g = m(\mathbf{X})$ and covariance $\mathbf{C}_0^g = k(\mathbf{X}, \mathbf{X})$.

Algorithm 1. Recursive Gaussian process (RGP)

```
  ▷ Inference
1: Calculate gain matrix $\mathbf{J}_t$ according to (8)
2: Calculate mean $\underline{\mu}_t^p$ by means of (6) and
   covariance matrix $\mathbf{C}_t^p$ by means of (9)
  ▷ Update
3: Calculate gain matrix $\tilde{\mathbf{G}}_t$ according to (12)
4: Calculate mean $\underline{\mu}_t^g$ by means of (10) and
   covariance matrix $\mathbf{C}_t^g$ by means of (11)
```

4. On-line learning

In this section, the assumption of a priori known hyperparameters is relaxed. Instead, the goal is now to learn the hyperparameters $\underline{\eta}$ simultaneously with estimating the values of the latent function $g(\cdot)$ at the basis vectors. This is achieved by formulating the learning part as a recursive parameter estimation problem, which can be performed together with the function value estimation. Similar to Section 3, this boils down to calculating a joint posterior distribution $p\left(\underline{z}_t | \underline{y}_{1:t}\right) = \mathcal{N}\left(\underline{z}_t; \underline{\mu}_t^z, \mathbf{C}_t^z\right)$, where $\underline{z}_t^{\mathrm{T}} \triangleq \left[\underline{g}^{\mathrm{T}}, \underline{\eta}_t^{\mathrm{T}}\right]$ is the joint hidden state with mean and covariance

$$\underline{\mu}_t^z \triangleq \begin{bmatrix} \underline{\mu}_t^g \\ \underline{\mu}_t^\eta \end{bmatrix}, \quad \mathbf{C}_t^z \triangleq \begin{bmatrix} \mathbf{C}_t^g & \mathbf{C}_t^{g\eta} \\ \mathbf{C}_t^{\eta g} & \mathbf{C}_t^\eta \end{bmatrix}.$$

Starting point for this calculation is a joint prior distribution $p\left(\underline{z}_{t-1} | \underline{y}_{1:t-1}\right)$ at step $t-1$, which is updated with the new observations $\underline{y}_t$. This requires the following two operations: (*inference*) calculating a joint distribution $p\left(\underline{z}_{t-1}, \underline{g}_t | \underline{y}_{1:t-1}\right)$ by exploiting the results of Section 3.1, and (*update*) incorporation of the new observations $\underline{y}_t$ and marginalization to obtain $p\left(\underline{z}_t | \underline{y}_{1:t}\right)$.

4.1. Inference

To incorporate the new inputs $\mathbf{X}_t$, it is necessary to infer the latent function $g(.)$ at $\mathbf{X}_t$. For this purpose, the intermediate result (9) derived in Section 3.1 is exploited. The part of the mean $\underline{q}$ and the covariance $\mathbf{Q}$ regarding $\underline{g}_t$ can alternatively be calculated by employing a Kalman predictor on the linear state-space model

$$\underline{g}_t = \mathbf{J}_t \cdot \underline{g} + \underline{w}_t, \quad \underline{w}_t \sim \mathcal{N}(\underline{b}, \mathbf{B}), \tag{13}$$

where $\underline{b} \triangleq m(\mathbf{X}_t) - \mathbf{J}_t \cdot m(\mathbf{X})$ and $\mathbf{B}$ is according to (7). In order to also correlate $\underline{g}_t$ with the hyperparameters, the model in (13) is extended to a state-space model given by

$$\begin{bmatrix} \underline{z}_{t-1} \\ \underline{g}_t \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \\ \mathbf{J}_t\left(\underline{\eta}_{t-1}\right) & \mathbf{0} \end{bmatrix}}_{\triangleq \mathbf{A}_t(\underline{\eta}_{t-1})} \cdot \underbrace{\begin{bmatrix} \underline{g} \\ \underline{\eta}_{t-1} \end{bmatrix}}_{\underline{z}_{t-1}} + \underline{w}_t, \tag{14}$$

where the noise $\underline{w}_t \sim \mathcal{N}\left(\underline{\mu}_t^w, \mathbf{C}_t^w\right)$ is Gaussian with mean and covariance

$$\underline{\mu}_t^w \triangleq \begin{bmatrix} \underline{0} \\ \underline{0} \\ \underline{b}\left(\underline{\eta}_{t-1}\right) \end{bmatrix}, \quad \mathbf{C}_t^w \triangleq \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}\left(\underline{\eta}_{t-1}\right) \end{bmatrix}, \tag{15}$$

respectively. Please note, that we now made the dependence on the hyperparameters explicit. The first two rows in (14) and (15) are merely an identity mapping of the given joint state $\underline{z}_{t-1}$, while the last row corresponds to (13).

Based on model (14), performing a prediction would yield the desired joint distribution $p\left(\underline{z}_{t-1}, \underline{g}_t | \underline{y}_{1:t-1}\right)$. Unfortunately, the model is nonlinear with respect to the hyperparameters $\underline{\eta}_{t-1}$ and thus, the prediction cannot be performed exactly in closed form. An approximate prediction has to be employed instead. In order to keep the approximation error bounded, we exploit the fact that the model in (14) is *conditionally linear*, i.e., for a given hyperparameter the model is linear and prediction can be performed exactly via the Kalman predictor.

For conditionally linear models, efficient prediction techniques have been proposed in [1]. Here, for the nonlinear part—the hyperparameters in our case—a collection of so-called *sigma points* $\hat{\underline{\eta}}_i$ is selected and given weights ω_i, $i = 1 \ldots s$. The mostly employed selection scheme for sigma points, which is also applied here, is the *unscented transform* [9]. However, any sigma point algorithm can be employed for the proposed GP learning approach, see for instance [7,6]. Compared to Monte Carlo sampling, the sigma points have the benefit of being deterministically selected and the sample mean as well as the sample covariance coincide with the mean $\underline{\mu}_{t-1}^{\eta}$ and covariance $\mathbf{C}_{t-1}^{\eta}$.

For each sigma point $\hat{\underline{\eta}}_i$ a Kalman predictor is applied on (14). Combining the individual predictions yields the mean and covariance

$$\underline{\mu}_t^p = \sum_{i=1}^{s} \omega_i \cdot \underline{\mu}_i^p, \tag{16}$$

$$\mathbf{C}_t^p = \sum_{i=1}^{s} \omega_i \cdot ((\underline{\mu}_i^p - \underline{\mu}_t^p)(\underline{\mu}_i^p - \underline{\mu}_t^p)^{\mathrm{T}} + \mathbf{C}_i^p) \tag{17}$$

of the joint distribution $p\left(\underline{z}_{t-1}, \underline{g}_t | \underline{y}_{1:t-1}\right) \approx \mathcal{N}\left(\underline{\mu}_t^p, \mathbf{C}_t^p\right)$ with

$$\underline{\mu}_i^p \triangleq \mathbf{A}_t(\hat{\underline{\eta}}_i) \begin{bmatrix} \underline{\mu}_{t-1}^g + \mathbf{S}_t \cdot (\hat{\underline{\eta}}_i - \underline{\mu}_{t-1}^{\eta}) \\ \hat{\underline{\eta}}_i \end{bmatrix} + \underline{\mu}_t^w(\hat{\underline{\eta}}_i), \tag{18}$$

$$\mathbf{C}_i^p \triangleq \mathbf{A}_t(\hat{\underline{\eta}}_i) \begin{bmatrix} \mathbf{C}_{t-1}^g - \mathbf{S}_t \mathbf{C}_{t-1}^{\eta g} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{A}_t(\hat{\underline{\eta}}_i)^{\mathrm{T}} + \mathbf{C}_t^w(\hat{\underline{\eta}}_i) \tag{19}$$

and $\mathbf{S}_t = \mathbf{C}_{t-1}^{g\eta}(\mathbf{C}_{t-1}^{\eta})^{-1}$.

4.2. Update

In order to incorporate the new observations $\underline{y}_t$ in a very computationally efficient manner, the update is performed in two steps. For this purpose, we decompose the joint distribution into an observed and an unobserved part according to

$$p(\underline{z}_t | \underline{y}_{1:t}) = \int \underbrace{p(\underline{g}, \underline{\eta}_t^- | \sigma, \underline{g}_t)}_{\text{unobserved}} \cdot \underbrace{p(\sigma, \underline{g}_t | \underline{y}_{1:t})}_{\text{observed}} \mathrm{d}\underline{g}_t,$$

where $\underline{\eta}_t^-$ indicates the vector of all hyperparameters excluding σ. The observable state $\underline{o}_t^{\mathrm{T}} \triangleq \left[\sigma, \underline{g}_t^{\mathrm{T}}\right]$ directly affects the observations $\underline{y}_t$ according to (1) and thus, can be directly updated. In the second step, correlation in the joint distribution $p\left(\underline{z}_{t-1}, \underline{g}_t | \underline{y}_{1:t-1}\right)$ is exploited for updating the unobservable part $\underline{u}_{t-1}^{\mathrm{T}} \triangleq \left[\underline{g}^{\mathrm{T}}, (\underline{\eta}_t^-)^{\mathrm{T}}\right]$, where the decomposition of mean $\underline{\mu}_t^p$ and covariance $\mathbf{C}_t^p$ regarding $\underline{u}_{t-1}$ and $\underline{o}_t$ is according to

$$\underline{\mu}_t^p = \begin{bmatrix} \underline{\mu}_{t-1}^u \\ \underline{\mu}_t^o \end{bmatrix}, \quad \mathbf{C}_t^p = \begin{bmatrix} \mathbf{C}_{t-1}^u & \mathbf{C}_t^{uo} \\ \mathbf{C}_t^{ou} & \mathbf{C}_t^o \end{bmatrix}.$$

This two-step procedure reduces the computational demand significantly as smaller matrices are multiplied and inverted.

4.2.1. Update observable state

The sigma point technique utilized for the inference step is one way for updating the observable state. However, to not introduce approximation errors, the process model (1) is reformulated to

$$y_i = g(\underline{x}_i) + \sigma \cdot v, \quad v \sim \mathcal{N}(0, 1) \tag{20}$$

with v being uncorrelated with σ. The model (20) is equivalent to (1) since $\epsilon = \sigma \cdot v$ with identical mean and variance. Here, the standard deviation σ of the observation noise is made explicitly accessible. This simplifies the update, as the mean and covariance of the observations as well as the cross-covariance between observable state and observations can be calculated exactly in closed form according to

$$\underline{\mu}_t^y = \mathrm{E}\left\{\underline{y}_t\right\} = \mathrm{E}\left\{\underline{g}_t + \sigma \cdot \underline{v}\right\} = \mathrm{E}\left\{\underline{g}_t\right\}, \tag{21}$$

$$\mathbf{C}_t^y = \mathrm{cov}\left\{\underline{y}_t\right\} = \mathrm{cov}\left\{\underline{g}_t\right\} + \mathrm{var}\{\sigma\} + \mathrm{E}\{\sigma\}^2, \tag{22}$$

$$\mathbf{C}_t^{oy} = \mathrm{cov}\left\{\underline{o}_t, \underline{y}_t\right\} = \mathrm{cov}\left\{\underline{o}_t, \underline{g}_t\right\}, \tag{23}$$

where $\mathrm{E}\{\sigma\}$, $\mathrm{E}\left\{\underline{g}_t\right\}$ are the elements of $\underline{\mu}_t^o$ and $\mathrm{var}\{\sigma\}$, $\mathrm{cov}\left\{\underline{g}_t\right\}$, $\mathrm{cov}\left\{\underline{o}_t, \underline{g}_t\right\}$ are the elements of $\mathbf{C}_t^o$.

Assuming that the observed state $\underline{o}_t$ and the observations $\underline{y}_t$ are jointly Gaussian distributed—a typical assumption in Gaussian filters like the unscented Kalman filter—updating the observed state can be performed by conditioning on the observations, which yields the desired conditional distribution $p\left(\sigma, \underline{g}_t | \underline{y}_{1:t}\right) \approx \mathcal{N}\left(\underline{\mu}_t^e, \mathbf{C}_t^e\right)$ with mean and covariance according to

$$\underline{\mu}_t^e = \underline{\mu}_t^o + \mathbf{G}_t \cdot \left(\underline{y}_t - \underline{\mu}_t^y\right), \tag{24}$$

$$\mathbf{C}_t^e = \mathbf{C}_t^o - \mathbf{G}_t \cdot \mathbf{C}_t^y \cdot \mathbf{G}_t^{\mathrm{T}} \tag{25}$$

with gain matrix $\mathbf{G}_t = \mathbf{C}_t^{o,y}(\mathbf{C}_t^y)^{-1}$ and $\underline{\mu}_t^y$, $\mathbf{C}_t^y$, $\mathbf{C}_t^{oy}$ of (21)–(23).

4.2.2. Update joint state

By means of (24) and (25) it is now possible to update the unobserved part. Therefore, the results in [1] are exploited, which yields the distribution $p\left(\underline{g}, \underline{\eta}_t^- | \sigma, \underline{g}_t\right) = \mathcal{N}\left(\underline{\mu}_t^u, \mathbf{C}_t^u\right)$ with mean and covariance

$$\underline{\mu}_t^u = \underline{\mu}_{t-1}^u + \mathbf{L}_t \cdot (\underline{\mu}_t^e - \underline{\mu}_t^o), \tag{26}$$

$$\mathbf{C}_t^u = \mathbf{C}_{t-1}^u + \mathbf{L}_t \cdot (\mathbf{C}_t^e - \mathbf{C}_t^o) \cdot \mathbf{L}_t^{\mathrm{T}}, \tag{27}$$

where $\mathbf{L}_t = \mathbf{C}_t^{uo}(\mathbf{C}_t^o)^{-1}$. It is worth mentioning that this update has the same structure as the backward pass of the Rauch–Tung–Striebel smoother [14]. To finalize the update step and thus to obtain the joint posterior distribution $p\left(\underline{z}_t | \underline{y}_{1:t}\right) = \mathcal{N}\left(\underline{z}_t; \underline{\mu}_t^z, \mathbf{C}_t^z\right)$ with updated basis vectors and hyperparameters, the results in (24)–(27) are combined according to

$$\underline{\mu}_t^z = \begin{bmatrix} \underline{h}^{\mathrm{T}} \cdot \underline{\mu}_t^e \\ \underline{\mu}_t^u \end{bmatrix}, \quad \mathbf{C}_t^z = \begin{bmatrix} \underline{h}^{\mathrm{T}} \mathbf{C}_t^e \underline{h} & \underline{h}^{\mathrm{T}} \mathbf{C}_t^e \mathbf{L}_t^{\mathrm{T}} \\ \mathbf{L}_t \mathbf{C}_t^e \underline{h} & \mathbf{C}_t^u \end{bmatrix} \tag{28}$$

with $\underline{h}^{\mathrm{T}} \triangleq [1, 0, 0, \ldots, 0]$. The first row in (28) corresponds to marginalizing out $\underline{g}_t$.

The simultaneous regression and hyperparameter learning approach proposed above is named recursive GP with learning (RGP*) and is summarized in Algorithm 2.

Algorithm 2. Recursive Gaussian process with hyperparameter learning (RGP*)

```
▹ Inference
1: Draw sigma points $(\omega_i, \hat{\underline{\eta}}_i)$, $i = 1, \ldots, s$ from $\mathcal{N}(\underline{\mu}_{t-1}^{\eta}, \mathbf{C}_{t-1}^{\eta})$
2: For each sigma point determine $\underline{\mu}_i^p, \mathbf{C}_i^p$
     by means of (18) and (19)
3: Calculate $\underline{\mu}_t^p$ and $\mathbf{C}_t^p$ according to (16) and (17)
▹ Update
4: Calculate $\underline{\mu}_t^y, \mathbf{C}_t^y$, and $\mathbf{C}_t^{oy}$ according to (21)–(23)
5: Update mean $\underline{\mu}_t^e$ and covariance $\mathbf{C}_t^e$ of observed state by
     means of (24) and (25), respectively
6: Update mean $\underline{\mu}_t^u$ and covariance $\mathbf{C}_t^u$ of unobserved state by
     means of (26) and (27), respectively
7: Calculate mean $\underline{\mu}_t^z$ and covariance $\mathbf{C}_t^z$ of joint state $\underline{z}_t$ by
     means of (28)
```

5. Discussion

To predict the latent function $g(\cdot)$ for non basis vectors, e.g., for plotting the estimate, the steps 1–4 have to be performed. Similarly, additional basis vectors can be introduced, e.g., in order to refine the resolution or improve the accuracy. For this purpose, the inference step has to be applied, where $\mathbf{X}_t$ now contains the location of the additional basis vectors.

Directly modeling some of the hyperparameters by means of a Gaussian distribution may not be appropriate in some cases. For instance, the length-scale hyperparameters of the SE kernel in (2) have to be positive. To account for such constraints, a standard trick in GP regression is to transform the hyperparameters first and then to train the transformed parameters. After training, the inverse transformation is applied in order to obtain the original hyperparameters. In case of positive hyperparameters, the logarithm for transforming and the exponential function as inverse transformation are common. RGP* can directly be used to also train/estimate transformed hyperparameters.

Assuming Gaussian noise ϵ in (1) is not reasonable for every application. Capturing a non-Gaussian distribution by the proposed methods can for instance be achieved via warping as proposed in [18]. Alternatively, $p\left(\underline{g}|\underline{y}_{1:t-1}\right)$ could be represented by means of a mixture of Gaussians.

RGP is exact in all computations, i.e., no approximations are employed despite the fact that only a fixed set of basis vectors is used. RGP* instead requires two approximations: the sigma point approximation of $\underline{\eta}_t$ in the inference step and the jointly Gaussian assumption of observations and observed state in the update step. Furthermore, if σ is not correlated with any other element of the joint state $\underline{z}_t$, it will not be correlated with $\underline{g}_t$ in the inference step. This correlation however is important to learn σ as well. This drawback can be compensated by ensuring that the initial covariance $\mathbf{C}_0^\eta$ has non-zero off-diagonal elements. In the simulation (see Section 6), $\mathbf{C}_0^\eta$ is almost a diagonal matrix, except of the values that describe the correlation between σ and all other hyperparameters. Here, full (positive) correlation is assumed, i.e., the values of the corresponding line and column in $\mathbf{C}_0^\eta$ are chosen in such a way that the sum of the off-diagonal values is equal to the diagonal value–this satisfies the row sum and the column sum criterion. The initial cross-covariance $\mathbf{C}_0^{g,\eta}$ is assumed to be a zero matrix.

The computation costs of RGP* for a single time step t scale with $\mathcal{O}\left(s \cdot n_t \cdot (m+r)^2 + n_t^3\right)$ and the memory costs scale with $\mathcal{O}\left((m+r)^2\right)$, where s is the number of sigma points, n_t is the number of observations at step t, m is the number of basis vectors, and r is the dimension of $\underline{\eta}$. In case of the employed unscented transform to generate the sigma points, $s = 2r+1$ and thus scales linearly with the number of hyperparameters. The costs of RGP are even less and scale with $\mathcal{O}(n_t \cdot m^2)$ for computations and $\mathcal{O}(m^2)$ for memory. If at each step t the same number of observations is processed, then the computational and memory costs are constant for each step for both RGP and RGP*. Furthermore and in contrast to a full GP the computational and memory costs do not increase over time, i.e., when more and more observations become available.

6. Results

The proposed approaches are compared to existing GP algorithms: a standard GP, SOGP, and SPGP. All methods are implemented in MATLAB by means of the GPML toolbox (http://www.gaussianprocess.org/gpml/code).

For all experiments, two different covariance functions are considered: the SE kernel (2) as well as the sum of SE kernel and the so-called neural network (NN) kernel

$$k(\underline{x}, \underline{x}') = \beta^2 \sin^{-1}\left(\frac{\underline{x}^{\mathrm{T}}\Lambda^{-2}\underline{x}'}{\sqrt{f(\underline{x})f(\underline{x}')}}\right) \tag{29}$$

with $f(\underline{x}) = 1 + \underline{x}^{\mathrm{T}}\Lambda^{-2}\underline{x}$, signal variance β^2, and the diagonal matrix Λ of characteristic length-scales. While the SE kernel is stationary, i.e., it is a function of $\Delta\underline{x} \triangleq \underline{x} - \underline{x}'$, the NN kernel is non-stationary. Thus, also the sum of SE and NN kernel—denoted as SE+NN in the following—allows modeling non-stationary processes.

To train the GP methods, at every step t a set of observations $\underline{y}_t$ at inputs $\mathbf{X}_t$ is randomly selected from a data set. At the end of training, the regression performance of the different approaches is evaluated and compared using a randomly selected test data set. As performance criteria we employ the root mean square error (rmse), the negative log-likelihood (nll), and the total computation time. For each experiment, 50 Monte Carlo (MC) runs are performed.

Full GP and RGP require off-line learning of the hyperparameters. Therefore, 100 input-observation pairs are selected randomly and evidence maximization is performed [13]. Then, the actual training takes place. SOGP and RGP* instead, learn the hyperparameters on-line by exploiting the data provided at step t, while SPGP learns the hyperparameters off-line *after* collecting all data. The number of active set elements is identical for all approaches.

6.1. Synthetic data

At first data generated by means of two different synthetic functions are considered. The first function

$$y = \frac{x}{2} + \frac{25 \cdot x}{1 + x^2} \cdot \cos(x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 0.1) \tag{30}$$

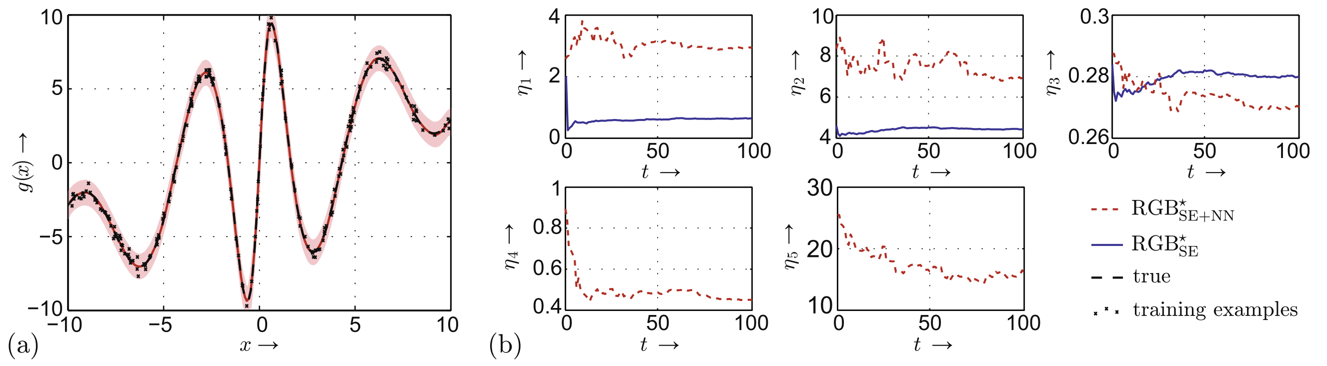

Fig. 1. Exemplary regression result of proposed approach for function (30). (a) True function (black, dashed line), regression result by RGP* with SE+NN covariance function together with 99% confidence area, and the training examples of step $t = 100$ (black crosses). (b) Evolution of the hyperparameters of RGP* with SE (blue, solid) and SE+NN covariance function (red, dashed), where $\eta_1 = l_1$ (length-scale), $\eta_2 = \alpha_1$ (signal standard deviation), $\eta_3 = \sigma$ (noise standard deviation) are the parameters of the SE kernel and $\eta_4 = l_2$ (length-scale), $\eta_5 = \beta$ (signal standard deviation) are the parameters of the NN kernel. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

is smooth but non-stationary. It is similar to the growth model proposed in [10]. At each step t, 40 input-observation pairs are selected randomly from the interval $[-10, 10]$. In total 100 steps are performed. The sparse representation comprises 50 elements, which are placed equidistant on the interval $[-10, 10]$. As second function we consider

$$y = \mathcal{N}(0.6, 0.04) + \mathcal{N}(0.15, 0.0015) + 4 \cdot H(0.3) + \epsilon, \qquad (31)$$

where $H(\cdot)$ is the Heaviside step function with $H(a) = 0$ if $x \leqslant a$ and $H(a) = 1$ if $x > a$. This function has a discontinuity at $x = 0.3$ and was considered as a benchmark in [20]. The noise ϵ has variance $\sigma^2 = 0.16$. A total of 70 steps are performed, with 50 data points per step drawn from $[-2, 2]$. On this interval, 30 active set elements are placed equidistant.

In Fig. 1(a), an exemplary regression result of RGP* with SE+NN kernel is depicted. The true function is accurately reconstructed. As shown in Fig. 1(b), the hyperparameters are adjusted over time and converge. This leads to improved regression results compared to the other hyperparameter learning approaches SOGP and SPGP as can be seen in Table 1. This holds for both covariance functions, whereas SE+NN yields better results as it is possible to capture the non-stationarity thanks to the non-stationary NN kernel (29). Compared to a full GP, RGP* is slightly inferior. The off-line hyperparameter optimization provides optimal results and RGP* cannot improve further. With the optimal hyperparameters however, RGP performs close to a full GP but with significantly lower runtime.

The results in Table 2 indicate that off-line hyperparameter optimization is not always optimal. Here, the hyperparameters learned by RGP* result in better estimates compared to all other algorithms with at the same time lower computational load. It is worth mentioning that RGP* is the only sparse approach that really exploits the properties of the SE+NN kernel resulting in an improved regression compared to the SE kernel.

6.2. Particulate matter data

Here, satellite observations of the Global Monitoring for Environment and Security program (`http://www.gmes.info/`, data via `ftp://data-portal.ecmwf.int/`) are considered. The inputs are two-dimensional measurement locations and the observations are particulate matter (PM_{10}) measurements at these locations. The data set recorded at October 10, 2011 comprises 10,000 elements. For each MC run, 100 active set elements are distributed randomly over the input domain.

Merely the SE covariance function is employed as the SE+NN covariance function does not lead to any improvement. The average performance is listed in Table 3. While all GP approaches except of SOGP have a similar rmse, there is a strong difference in terms of nll. Here, RGP* performs best as it provides lower prediction covariances and thus, is more certain about its estimates. The deviation of SOGP can be explained by Fig. 2. Occasionally, SOGP fails in accurately modeling the high PM_{10} concentrations in the upper part and it overfits around region (10,40). SPGP behaves similarly, but with much lower deviation from the ground truth. RGP* instead models the concentrations very accurately and even outperforms the full GP, as better hyperparameters are learned from the data.

Table 1
Average rmse, nll, and runtime for function (30).

	rmse	nll	Time in s
SE			
Full GP	0.31 ± 0.02	0.25 ± 0.05	0.82
RGP	0.31 ± 0.02	0.26 ± 0.06	0.16
RGP*	0.37 ± 0.02	0.41 ± 0.14	0.65
SOGP	1.18 ± 0.03	7.49 ± 0.4	0.93
SPGP	0.54 ± 0.02	1.15 ± 0.11	13.05
SE+NN			
Full GP	0.30 ± 0.02	0.24 ± 0.06	1.46
RGP	0.31 ± 0.03	0.24 ± 0.06	0.11
RGP*	0.35 ± 0.05	0.34 ± 0.12	1.81
SOGP	0.44 ± 0.03	0.80 ± 0.13	2.58
SPGP	0.39 ± 0.02	0.51 ± 0.09	24.40

Table 2
Average rmse, nll, and runtime for function (31).

	rmse	nll	Time in s
SE			
Full GP	1.38 ± 0.41	1.70 ± 0.33	0.92
RGP	1.40 ± 0.38	1.98 ± 0.40	0.45
RGP*	0.98 ± 0.11	1.48 ± 0.23	0.38
SOGP	1.68 ± 0.07	2.89 ± 0.17	0.8
SPGP	1.63 ± 0.10	1.91 ± 0.06	5.6
SE+NN			
Full GP	1.04 ± 0.43	1.41 ± 0.35	2.57
RGP	1.12 ± 0.35	1.86 ± 0.19	0.48
RGP*	0.88 ± 0.10	1.39 ± 0.15	1.14
SOGP	1.68 ± 0.07	2.88 ± 0.17	2.21
SPGP	1.65 ± 0.07	1.93 ± 0.05	10.85

Table 3
Average rmse, nll, and runtime for PM_{10} data.

	SE		
	rmse	nll	Time in s
Full GP	0.59 ± 0.01	1337.37 ± 217.92	0.18
RGP	0.59 ± 0.02	82.20 ± 24.48	0.06
RGP*	0.56 ± 0.01	45.47 ± 9.41	0.89
SOGP	0.86 ± 0.01	598.34 ± 20.22	1.58
SPGP	0.57 ± 0.01	114.83 ± 22.44	13.87

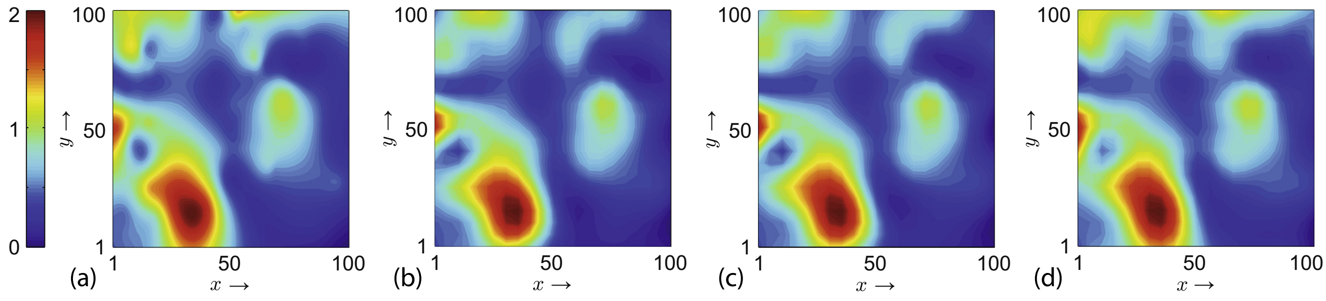

Fig. 2. Particulate matter estimates of different approaches. (a) All training data, (b) SOGP, (c) SPGP, and (d) RGP*.

7. Conclusion and future work

Two novel approaches for on-line GP regression have been proposed. Both approaches are especially well suited for streaming data. Given the hyperparameters, the first approach provides similar performance than a full GP, but with significantly lower computation and memory costs. If the hyperparameters are unknown, our second approach is able to learn the hyperparameters on-line from data. Here, the regression performance is comparable to or even better than state-of-the-art approaches, but with lower computational demand. Future work is devoted to adjust the location and number of the basis vectors over time.

References

[1] F. Beutler, M.F. Huber, U.D. Hanebeck, Gaussian filtering using state decomposition methods, in: Proceedings of the 12th International Conference on Information Fusion (Fusion), Seattle, Washington, 2009, pp. 579–586.
[2] L. Csató, M. Opper, Sparse on-line Gaussian processes, Neural Comput. 14 (2002) 641–668.
[3] J. Hartikainen, S. Särkkä, Kalman filtering and smoothing solutions to temporal Gaussian process regression models, in: Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing, 2010, pp. 379–384.
[4] J. Hensman, N. Fusi, N.D. Lawrence, Gaussian processes for big data, in: Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence, Bellevue, Washington, 2013.
[5] M.F. Huber, Recursive Gaussian process regression, in: Proceedings of the 38th International Conference on Acoustics, Sound, and Signal Processing (ICASSP), Vancouver BC, Canada, 2013, pp. 3362–3366.
[6] M.F. Huber, U.D. Hanebeck, Gaussian filter based on deterministic sampling for high quality nonlinear estimation, in: Proceedings of the 17th IFAC World Congress, Seoul, Republic of Korea, 2008.
[7] K. Ito, K. Xiong, Gaussian filters for nonlinear filtering problems, IEEE Trans. Autom. Control 45 (2000) 910–927.
[8] A.H. Jazwinski, Stochastic Processes and Filtering Theory, Dover Publications Inc., 2007.
[9] S.J. Julier, J.K. Uhlmann, Unscented filtering and nonlinear estimation, Proc. IEEE 92 (2004) 401–422.
[10] G. Kitagawa, Monte Carlo filter and smoother for non-Gaussian nonlinear state space models, J. Comput. Graph. Stat. 5 (1996).
[11] D. Nguyen-Tuong, M. Seeger, J. Peters, Real-time local GP model learning, in: O. Sigaud, J. Peters (Eds.), From Motor Learning to Interaction Learning in Robots, Studies in Computational Intelligence, 264, Springer-Verlag, 2010. pp. 193–207.
[12] J. Quiñonero-Candela, C.E. Rasmussen, A unifying view of sparse approximate Gaussian process regression, J. Mach. Learn. Res. 6 (2005) 1939–1959.
[13] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, The MIT Press, 2006.
[14] H.E. Rauch, F. Tung, C.T. Striebel, Maximum likelihood estimates of linear dynamic systems, AIAA J. 3 (1965) 1445–1450.
[15] S. Reece, S. Roberts, An introduction to Gaussian processes for the Kalman filter expert, in: Proceedings of the 13th International Conference on Information Fusion, Edinburgh, UK, 2010.
[16] A.J. Smola, P. Bartlett, Sparse greedy Gaussian process regression, in: T.K. Leen, T.G. Diettrich, V. Tresp (Eds.), Advances in Neural Information Processing Systems, 13, The MIT Press, 2001. pp. 619–625.
[17] E. Snelson, Z. Ghahramani, Sparse Gaussian processes using pseudo-inputs, in: Y. Weiss, B. Schölkopf, J. Platt (Eds.), Advances in Neural Information Processing Systems, 18, The MIT Press, 2006. pp. 1259–1266.
[18] E. Snelson, C.E. Rasmussen, Z. Ghahramani, Warped Gaussian processes, in: Advances in Neural Information Processing Systems, 16, The MIT Press, 2004.
[19] V. Tresp, A Bayesian committee machine, Neural Comput. 12 (2000) 2719–2741.
[20] S.A. Wood, W. Jian, M. Tanner, Bayesian mixture of splines for spatially adaptive nonparametric regression, Biometrika 89 (2002) 513–528.