RECURSIVE GAUSSIAN PROCESS REGRESSION

Marco F. Huber

AGT International, Darmstadt, Germany
marco.huber@ieee.org

ABSTRACT

For large data sets, performing Gaussian process regression is computationally demanding or even intractable. If data can be processed sequentially, the recursive regression method proposed in this paper allows incorporating new data with constant computation time. For this purpose two operations are performed alternating on a fixed set of so-called basis vectors used for estimating the latent function: First, inference of the latent function at the new inputs. Second, utilization of the new data for updating the estimate. Numerical simulations show that the proposed approach significantly reduces the computation time and at the same time provides more accurate estimates compared to existing on-line and/or sparse Gaussian process regression approaches.

Index Terms— Gaussian processes, recursive processing, on-line regression, smoothing

1. INTRODUCTION

Gaussian processes (GPs) allow non-parametric learning of a regression function from noisy data. They can be considered Gaussian distributions over functions conditioned on the data [1]. In contrast to classical regression, GPs provide not only a regression function but also provide uncertainty estimates (error bars) depending on the noise and variability of the data.

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 (see for example [2, 3, 4, 5, 6, 7, 8]). Typically, these approximations operate on a subset of size s of the training data, which reduces the computation load to $\mathcal{O}(s^2 \cdot n)$ for training. However, most of these approximations assume that the whole data set is available prior to the training and thus, training is performed *off-line* in a batch mode.

Only a few approaches have been proposed that allow sequential training of GPs for data that arrives *on-line*, e.g., from a time series. In [9] for instance, clusters on the incoming data points are identified and created sequentially. The assignment of data points to clusters and the number of clusters, however, depend a threshold value that is heavily application specific and requires careful tuning. For specific kernel functions, the approach proposed in [10] allows transforming GP regression into a Kalman filtering and smoothing problem that merely scales with $\mathcal{O}(n)$. Unfortunately, this approach so far is only applicable for one-dimensional inputs.

Similar to [10], the approach proposed in this paper considers GP training a Bayesian filtering problem. To allow for a large number of inputs of *arbitrary* dimension, the regression function is represented by means of a finite set of *basis vectors*. Training with incoming data, i.e., updating the mean and covariance estimate featured by the basis vectors, is performed *on-line* in a recursive fashion. Thus, after updating, the newly arrived data points can be discarded, while the estimate provided by the basis vectors is sufficient for prediction.

2. PROBLEM STATEMENT

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

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

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, $\mathbf{X}_\mathcal{D} = \left[\underline{x}_1, \ldots, \underline{x}_n\right]$ 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 $g(.)$ from the data $\mathcal{D}$. The GP is completely defined by a mean function $m(\underline{x}) \triangleq \mathrm{E}\{g(\underline{x})\}$, which specifies 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 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

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

It is worth mentioning that the approach proposed in this paper holds for arbitrary mean and covariance functions. In (1), $\mathbf{\Lambda}$ is a diagonal matrix of the characteristic length-scales 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 variance σ^2 are called the *hyperparameters* of the GP. In this paper, it is assumed that the hyperparameters are given and thus, are not learned from data.

As a GP forms a Gaussian distribution over functions, we can write $g(\underline{x}) \sim \mathcal{GP}\left(m(\underline{x}), k(\underline{x}, \underline{x}')\right)$. For any finite set of inputs, the resulting distribution of the outputs is a multivariate Gaussian. 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

$$\mu_g(\underline{x}_*) = \mathrm{E}\{g_*\} = m_* + \underline{k}_*^{\mathrm{T}} \mathbf{K}_x^{-1} (\underline{y} - \underline{m}) \,, \tag{2}$$

$$\sigma_g^2(\underline{x}_*) = \mathrm{var}\{g_*\} = k_{**} - \underline{k}_*^{\mathrm{T}} \mathbf{K}_x^{-1} \underline{k}_* \,, \tag{3}$$

respectively. Here, $\mathrm{var}\{.\}$ 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 (2) and (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 regression approach aims at performing all calculations computationally very efficient on a set of $s \ll n$ so-called *basis vectors*.

Let $\mathbf{X} \triangleq \left[\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_s\right]$ be the matrix of locations of the basis vectors and $\underline{g} \triangleq g(\mathbf{X})$ the corresponding (unkown) values of the latent function. It is assumed that the basis vectors remain fixed for all processing steps $t = 0, 1, \ldots$ Since $g(\underline{x})$ is assumed to be a GP, the distribution $p_0(\underline{g}) = \mathcal{N}(\underline{g}; \underline{\mu}_0^g, \mathbf{C}_0^g)$ of $\underline{g}$ at the initial step $t = 0$ of the recursive processing is Gaussian with mean $\underline{\mu}_0^g \triangleq m(\mathbf{X})$ and covariance $\mathbf{C}_0^g \triangleq k(\mathbf{X}, \mathbf{X})$. At an arbitrary step $t > 0$, new observations $\underline{y}_t \triangleq [y_{t,1}, y_{t,2}, \ldots, y_{t,n_t}]^{\mathrm{T}}$ at inputs $\mathbf{X}_t \triangleq \left[\underline{x}_{t,1}, \underline{x}_{t,2}, \ldots, \underline{x}_{t,n_t}\right]$ become available. The goal is now to calculate the posterior distribution $p_t\big(\underline{g}|\underline{y}_{1:t}\big)$, with $\underline{y}_{1:t} = (\underline{y}_1, \ldots, \underline{y}_t)$, by updating the prior distribution of $\underline{g}$ at step $t-1$

$$p_{t-1} \triangleq p_{t-1}\big(\underline{g}|\underline{y}_{1:t-1}\big) = \mathcal{N}(\underline{g}; \underline{\mu}_{t-1}^g, \mathbf{C}_{t-1}^g) \tag{4}$$

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

3. RECURSIVE PROCESSING

One might think of exploiting (2) and (3) for incorporating the new observations. This however, is not suitable for recursive processing for mainly two reasons. Firstly, the latest estimate of the latent function in terms of the distribution p_{t-1} or the mean $\underline{\mu}_{t-1}^g$ and covariance $\mathbf{C}_{t-1}^g$ is not utilized. Secondly, no correlation or cross-covariance between $\mathbf{X}$ and $\mathbf{X}_t$ is provided, which however is of paramount importance for updating p_{t-1}. Instead, for deriving a recursive algorithm, the desired posterior distribution is expanded according to

$$p_t = c_t \int \underbrace{p_t(\underline{y}_t|\underline{g}, \underline{g}_t) \cdot p_{t-1}\big(\underline{g}, \underline{g}_t|\underline{y}_{1:t-1}\big)}_{= p_t\left(\underline{g}, \underline{g}_t|\underline{y}_{1:t}\right)} \mathrm{d}\underline{g}_t \tag{5}$$

by applying Bayes' law and by integrating out $\underline{g}_t \triangleq g(\mathbf{X}_t)$ from the joint distribution $p_t\big(\underline{g}, \underline{g}_t|\underline{y}_{1:t}\big)$. Here, c_t is a normalization constant. Based on (5), calculating the posterior distribution can be performed in two steps: I. *Inference*, i.e., calculating the joint prior $p_{t-1}\big(\underline{g}, \underline{g}_t|\underline{y}_{1:t-1}\big)$ given the prior p_{t-1} in (4). II. *Update*, i.e., updating the joint prior with the observations $\underline{y}_t$ and integrating out $\underline{g}_t$.

3.1. Inference

In order to determine the joint prior $p_{t-1}\big(\underline{g}, \underline{g}_t|\underline{y}_{1:t-1}\big)$, it is important to emphasize that the joint distribution $p\big(\underline{g}, \underline{g}_t\big)$ is Gaussian with mean and covariance

$$\underline{\mu} = \begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{X}_t) \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} k(\mathbf{X}, \mathbf{X}) & k(\mathbf{X}, \mathbf{X}_t) \\ k(\mathbf{X}_t, \mathbf{X}) & k(\mathbf{X}_t, \mathbf{X}_t) \end{bmatrix} \,, \tag{6}$$

respectively. This follows from the fact that $g(.)$ is a GP and any finite representation of this GP yields a Gaussian distribution. Thus, the joint prior can be written as

$$\begin{aligned} p_{t-1}\big(\underline{g}, \underline{g}_t|\underline{y}_{1:t-1}\big) &\approx p(\underline{g}_t|\underline{g}) \cdot p_{t-1} \\ &= \mathcal{N}\big(\underline{g}_t; \underline{\mu}_t^p, \mathbf{B}\big) \cdot \mathcal{N}\big(\underline{g}; \underline{\mu}_{t-1}^g, \mathbf{C}_{t-1}^g\big) \,, \end{aligned} \tag{7}$$

with

$$\begin{aligned} \underline{\mu}_t^p &= m(\mathbf{X}_t) + \mathbf{J}_t \cdot \big(\underline{\mu}_{t-1}^g - m(\mathbf{X})\big) \,, \\ \mathbf{B} &= k(\mathbf{X}_t, \mathbf{X}_t) - \mathbf{J}_t \cdot k(\mathbf{X}, \mathbf{X}_t) \,, \\ \mathbf{J}_t &= k(\mathbf{X}_t, \mathbf{X}) \cdot k(\mathbf{X}, \mathbf{X})^{-1} \,. \end{aligned}$$

The first equality in (7) follows from assuming that $\underline{g}_t$ is conditionally independent of the past observations $\underline{y}_{1:t-1}$ given $\underline{g}$. Hence, the conditional distribution $p(\underline{g}_t|\underline{g})$ is Gaussian and results from the joint distribution $p\big(\underline{g}, \underline{g}_t\big)$ in (6) by conditioning on $\underline{g}$ (see for example Chapter 2.6 in [11]).

After some algebraic transformations, where some basic properties of Gaussian distributions and the Woodbury formula is utilized, the product in (7) yields the joint Gaussian $p_{t-1}\big(\underline{g}, \underline{g}_t|\underline{y}_{1:t-1}\big)$ of $\underline{g}$ and $\underline{g}_t$ with mean and covariance

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

with covariance $\mathbf{C}_t^p \triangleq \mathbf{B} + \mathbf{J}_t \mathbf{C}_{t-1}^g \mathbf{J}_t^{\mathrm{T}}$.

3.2. Update

Given the result of the previous section that the joint prior in (7) is a Gaussian $\mathcal{N}(\underline{q}, \mathbf{Q})$, the next step is to perform the update and marginalization in (5). For this purpose, (5) is rearranged to

$$p_t = \int \underbrace{c_t \cdot p_t\big(\underline{y}_t|\underline{g}_t\big) \cdot \overbrace{p_{t-1}\big(\underline{g}_t|\underline{y}_{1:t-1}\big)}^{= p_{t-1}\left(\underline{g}, \underline{g}_t|\underline{y}_{1:t-1}\right)} \cdot p_{t-1}\big(\underline{g}|\underline{g}_t, \underline{y}_{1:t-1}\big)}_{= p_t\left(\underline{g}_t|\underline{y}_{1:t}\right) \text{ (Kalman filter)}} \mathrm{d}\underline{g}_t \tag{8}$$

under consideration that $\underline{g}$ is not observed and thus, $p_t(\underline{y}_t|\underline{g}_t)$ is independent of $\underline{g}$. Since $p_t(\underline{y}_t|\underline{g}_t) = \mathcal{N}(\underline{y}_t; \underline{g}_t, \sigma^2 \mathbf{I})$ and

$p_{t-1}(\underline{g}_t|\underline{y}_{1:t-1}) = \mathcal{N}(\underline{g}_t; \underline{\mu}_t^p, \mathbf{C}_t^p)$ 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 equations yields $p_t(\underline{g}_t|\underline{y}_{1:t}) = \mathcal{N}(\underline{g}_t; \underline{\mu}_t^e, \mathbf{C}_t^e)$ with mean and covariance

$$\underline{\mu}_t^e = \underline{\mu}_t^p + \mathbf{G}_t \cdot \left(\underline{y}_t - \underline{\mu}_t^p\right) , \tag{9}$$

$$\mathbf{C}_t^e = \mathbf{C}_t^p - \mathbf{G}_t \mathbf{C}_t^p , \tag{10}$$

respectively, where $\mathbf{G}_t = \mathbf{C}_t^p \cdot \left(\mathbf{C}_t^p + \sigma^2 \mathbf{I}\right)^{-1}$ is the Kalman gain. The multiplication of the two Gaussians $p_t(\underline{g}_t|\underline{y}_{1:t})$ and $p_{t-1}\big(\underline{g}|\underline{g}_t, \underline{y}_{1:t-1}\big)$ in (8) 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 $\mathbf{L}_t = \mathbf{C}_{t-1}^g \mathbf{J}_t^{\mathrm{T}} \left(\mathbf{C}_t^p\right)^{-1}$ and

$$\underline{\mu}_t^g = \underline{\mu}_{t-1}^g + \mathbf{L}_t \cdot \left(\underline{\mu}_t^e - \underline{\mu}_t^p\right) , \tag{11}$$

$$\mathbf{C}_t^g = \mathbf{C}_{t-1}^g + \mathbf{L}_t \cdot (\mathbf{C}_t^e - \mathbf{C}_t^p) \cdot \mathbf{L}_t^{\mathrm{T}} . \tag{12}$$

However, since we are merely interested in obtaining the distribution $p_t = \mathcal{N}(\underline{g}; \underline{\mu}_t^g, \mathbf{C}_t^g)$, 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 $\mathbf{L}_t \mathbf{C}_t^e$.

3.3. Summary

Putting all together, at steps $t = 1, 2, \ldots$ the proposed approach recursively processes observations $\underline{y}_t$ at the inputs $\mathbf{X}_t$ by means of the following set of equations:

Inference

$$\mathbf{J}_t = k(\mathbf{X}_t, \mathbf{X}) \cdot k(\mathbf{X}, \mathbf{X})^{-1} , \tag{13}$$

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

$$\mathbf{C}_t^p = k(\mathbf{X}_t, \mathbf{X}_t) + \mathbf{J}_t \cdot \left(\mathbf{C}_{t-1}^g - k(\mathbf{X}, \mathbf{X})\right) \cdot \mathbf{J}_t^{\mathrm{T}} ,$$

Update

$$\tilde{\mathbf{G}}_t = \mathbf{C}_{t-1}^g \mathbf{J}_t^{\mathrm{T}} \cdot \left(\mathbf{C}_t^p + \sigma^2 \mathbf{I}\right)^{-1} , \tag{14}$$

$$\underline{\mu}_t^g = \underline{\mu}_{t-1}^g + \tilde{\mathbf{G}}_t \cdot \left(\underline{y}_t - \underline{\mu}_t^p\right) , \tag{15}$$

$$\mathbf{C}_t^g = \mathbf{C}_{t-1}^g - \tilde{\mathbf{G}}_t \mathbf{J}_t \mathbf{C}_{t-1}^g . \tag{16}$$

This recursion commences from the initial mean $\underline{\mu}_0^g \triangleq m(\mathbf{X})$ and covariance $\mathbf{C}_0^g \triangleq k(\mathbf{X}, \mathbf{X})$ of $\underline{g}$. The updated mean (15) and covariance (16) result from substituting $\underline{\mu}_t^e$ in (11) with (9) and $\mathbf{C}_t^e$ in (12) with (10), respectively, where $\tilde{\mathbf{G}}_t = \mathbf{L}_t \cdot \mathbf{G}_t$.

4. DISCUSSION

A close inspection of the inference step shows that it has the same structure as the backward pass of the *Rauch-Tung-Striebel* (RTS) smoother [12]. In contrast to classical RTS smoothing, the inference step operates in the input domain and not in the time domain. It predicts the function value $g(\underline{x}_*)$ at any (test) input $\underline{x}_*$ given all information acquired so far and thus, is dual to the prediction (2), (3) of a classical GP.

So far, it was assumed that the set of basis vectors is fixed. The inference step, however, can also be utilized for introducing new basis vectors. This might be of interest in locations where the current estimate of the latent function is inaccurate. By replacing $\mathbf{X}_t$ with $\left[\mathbf{X}, \mathbf{X}'\right]$, the inference step provides the initial mean and covariance as well as the cross-covariance between the new basis vectors and the old ones.

The computations of the inference step scale with $\mathcal{O}(s^2 \cdot n_t)$ due to calculating $\mathbf{J}_t$ in (13), where n_t is the number of observations at step t. Here, the inversion of the kernel matrix $k(\mathbf{X}, \mathbf{X})$ is computationally unproblematic, as it has to be calculated only once at step $t = 0$. Once the gain matrix $\mathbf{J}_t$ is calculated, predictions for a single test input are in $\mathcal{O}(s)$ (mean) and $\mathcal{O}(s^2)$ (covariance). Assuming that all observations are processed at once, predictions of the recursive GP are as complex as predictions of sparse GP approaches relying on a representation comparable to the basis vectors, e.g., pseudo-inputs in [4] or subset of regressors in [6]. In this case, the complexity is in $\mathcal{O}(s^2 \cdot n)$ for initialization as well as $\mathcal{O}(s)$ (mean) and $\mathcal{O}(s^2)$ (covariance) for predictions, respectively. In contrast to most sparse GP approaches, the proposed method can process new observations on-line.

The update step scales with $\mathcal{O}(n_t \cdot s^2)$, where the complexity results from matrix multiplications for which more efficient algorithms exist, e.g., Strassen's algorithm [13]. The inversion in (14) again is not critical as the affected matrix is of size $n_t \times n_t$, where typically $n_t \ll s$.

5. SIMULATION EXAMPLES

The proposed approach is compared to existing on-line and/or sparse GP approaches: a full GP (named FGP in the following), the on-line approaches local GP ([9], LGP) and sparse on-line GP ([14], SOGP), as well as the sparse (but off-line) approaches Bayesian committee machine ([5], BCM), subset of regressors ([6], SRM), and sparse GP using pseudo-inputs ([4], SGP). Further, our approach is applied in three different modes: updating/training the basis vectors with every observation (RGP_1), with a batch of 10 successive observations (RGP_{10}), or with a batch of all observations (RGP_{all}). In the latter case, RGP becomes an off-line algorithm. For training the on-line algorithms (LGP, SOGP, RGP_1, and RGP_{10}), the training data is presented sequentially, while for the remaining off-line algorithms, the training data is processed in a batch. All GP methods are implemented in MATLAB, where the GPML toolbox [15] is utilized for FGP and SGP.

At first, the one-dimensional nonlinear function

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

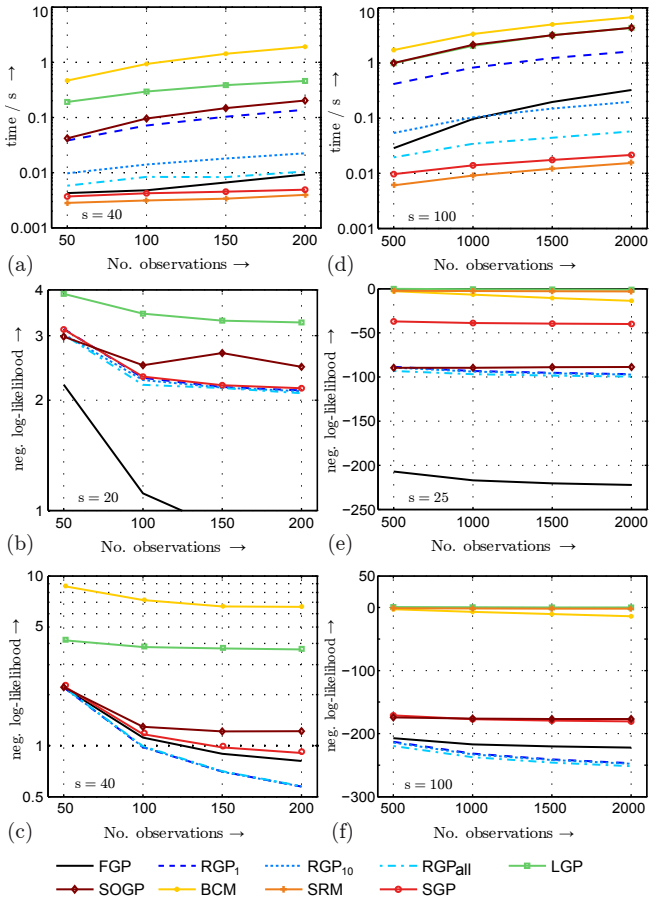

Fig. 1. Results: first column = first example and second column = second example. (a) Average runtime for different number of observations n and sparse elements $s = 40$. (b), (c) Average nll for $s = 20$ and $s = 40$, respectively. The average nll values are increased by one in order to have positive values and allow for a log-scale plot. In (b), SRM and BCM have average values higher than four and thus, are not shown. In (c), SRM is not depicted as its average values are significantly larger than 10. (d) Average runtime for $s = 100$. (e), (f) Average nll for $s = 25$ and $s = 100$, respectively.

is considered as an example. It is similar to the growth model proposed in [16]. To train the GPs, the inputs x_i, $i = 1, \ldots, n$ with $n \in \{50, 100, 150, 200\}$, are sampled uniform at random from the interval $[-10, 10]$. For testing, 200 input-observation pairs are considered. All GPs except of FGP use some sparse representation consisting of s elements (basis vectors, pseudo-inputs, clusters, etc.), where $s = 20$ and $s = 40$ are considered. In case of the RGPs, the basis vectors are placed equidistant on the interval $[-10, 10]$.

In the second example, the satellite observations of the Global Monitoring for Environment and Security program (GMES)[1] are considered. The inputs are the two-dimensional measurement locations and the observations are the particulate matter (PM_{10}) measurements at these locations. The data set comprises 10,000 elements and was recorded at 10th of October 2011. For training, $n \in \{500, 1{,}000, 1{,}500, 2{,}000\}$ elements and for testing 1,000 elements are selected randomly from the data set. For the sparse representations both $s = 25$ and $s = 100$ elements are considered. The basis vectors of the recursive GPs are placed on a regular grid.

For both examples, a zero mean function and the SE kernel (1) are used. All GP methods use the same hyperparameters. For comparison, two performance criteria are considered: the total runtime in seconds comprising training and testing. Further, the negative log-likelihood (nll) of the predicted observations at the test inputs. For both a lower value indicates a better performance, where the nll penalizes uncertainty and inconsistency (prediction error). The averaged results of 100 simulation runs are depicted in Figure 1.

Among all on-line GP methods RGP_1 and RGP_{10} are the fastest and the most accurate. However, RGP_1 has a significantly higher runtime as RGP_{10} due to some overhead when performing inference and update for every single observation.

While BCM cannot compete with RGP regarding runtime and nll, SRM is faster but has a by far higher prediction error and provides too low variance values, i.e., SRM is inaccurate and at the time too confident about its predictions. SGP is slightly less accurate than RGP, but its runtime is much lower. The proposed RGP however, is an on-line method that processes all observations only once. Operating SGP in an on-line fashion would require to revisit all observations acquired so far for each new input in order to provide updated pseudo-inputs. This of course would lead to a much higher computation time compared to RGP. Here, RGP clearly benefits from its recursive structure, which is not present for SGP.

For few observations, RGP is slower than FGP due to the overhead of managing and updating the basis vectors. However, RGP benefits from a large data set as considered in the second example. Here, RGP_{all} has a lower runtime compared to FGP since the prediction is less costly. Furthermore, RGP_{10} also becomes faster from 1,500 observations on. The same behavior is expected for RGP_1 for training data sets with more than 8,000 observations. If the number of basis vectors is sufficiently high, RGP can even have lower nll values compared to FGP. While the rmse between prediction and true values is similar for RGP and FGP, RGP provides much lower prediction variances without being over-confident.

6. CONCLUSIONS

The novel on-line Gaussian process regression approach proposed in this paper relies on a set of basis vectors that is updated recursively with new observations. The number of basis vectors and thus, the computation time for updating or prediction remains constant. Compared to existing on-line GP approaches, the proposed one is computationally more efficient and provides a higher prediction accuracy.

[1] http://www.gmes.info/ - Data accessible via ftp://data-portal.ecmwf.int/

7. REFERENCES

[1] Carl E. Rasmussen and Christopher K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2006.

[2] Neil Lawrence, Matthias Seeger, and Ralf Herbich, "Fast Sparse Gaussian Process Methods: The Informative Vector Machine," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds., pp. 609–616. The MIT Press, 2003.

[3] Alex J. Smola and Peter Bartlett, "Sparse Greedy Gaussian Process Regression," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietrich, and V. Tresp, Eds., pp. 619–625. The MIT Press, 2001.

[4] Ed Snelson and Zoubin Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. Platt, Eds., pp. 1259–1266. The MIT Press, 2006.

[5] Volker Tresp, "A Bayesian Committee Machine," *Neural Computation*, vol. 12, no. 11, pp. 2719–2741, 2000.

[6] Grace Wahba, *Spline Models for Observational Data*, chapter 7 - Finite-Dimensional Approximating Subspaces, CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1990.

[7] Christopher K. I. Williams and Matthias Seeger, "Using the Nystrm Method to Speed Up Kernel Machines," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Diettrich, and V. Tresp, Eds., pp. 682–688. The MIT Press, 2001.

[8] Ananth Ranganathan, Ming-Hsuan Yang, and Jeffrey Ho, "Online Sparse Gaussian Process Regression and Its Applications," *IEEE Transactions on Image Processing*, vol. 20, no. 2, pp. 391–404, Feb. 2011.

[9] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters, "Real-Time Local GP Model Learning," in *From Motor Learning to Interaction Learning in Robots*, Olivier Sigaud and Jan Peters, Eds., vol. 264 of *Studies in Computational Intelligence*, pp. 193–207. Springer-Verlag, 2010.

[10] Jouni Hartikainen and Simo Särkkä, "Kalman Filtering and Smoothing Solutions to Temporal Gaussian Process Regression Models," in *Proceedings of IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, Aug. 2010, pp. 379–384.

[11] Andrew H. Jazwinski, *Stochastic Processes and Filtering Theory*, Dover Publications, Inc., 2007.

[12] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum Likelihood Estimates of Linear Dynamic Systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, Aug. 1965.

[13] Volker Strassen, "Gaussian Elimination is not Optimal," *Numerische Mathematik*, vol. 13, no. 4, pp. 354–356, Aug. 1969.

[14] Lehel Csató and Manfred Opper, "Sparse on-line Gaussian processes," *Neural Computation*, vol. 14, no. 3, pp. 641–668, Mar. 2002.

[15] Carl Edward Rasmussen and Hannes Nickisch, "Gaussian Process for Machine Learning Toolbox," .

[16] Genshiro Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, pp. 1–25, 1996.