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.

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/

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

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

In (2) $\mathbf{\Lambda} = \operatorname{diag}(l_1, l_2, \ldots, l_d)$ 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., $\underline{\eta} = [\alpha, l_1, \ldots, l_d, \sigma]^{\mathsf{T}}$ 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

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

respectively. Here, $\operatorname{var}\{\cdot\}$ 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 $p_0\left(\underline{g}\right) = \mathcal{N}\left(\underline{g}; \underline{\mu}_0^g, \mathbf{C}_0^g\right)$ 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

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

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