# A Perturbed Gaussian Process Regression with Chunk Sparsification for Tracking Non-stationary Systems

Dong Li, Dongbin Zhao, Zhongpu Xia
The State Key Laboratory of Management and Control for Complex Systems. Institute of Automation
Chinese Academy of Sciences. Beijing 100190, China.
E-mail: dongleecsu@gmail.com, dongbin.zhao@ia.ac.cn, zhongpu.xia@gmail.com

**Abstract:** Gaussian processes provide a Bayesian regression framework, commonly used in stationary regression by online learning, where the data is fed one by one. In this paper, a perturbed Gaussian process with chunk sparsification is proposed to track non-stationary systems. A chunk sparsification mechanism is designed to select the optimal bases from a chunk of data. The non-stationary system is modeled by a perturbed Gaussian process (PGP), which enables it to track the changing of the system through forgetting the history information at one point or a chunk of points. Finally, PGP is tested on two experiments. The results show that PGP requires less bases than the approximate linear dependence (ALD) sparsification, especially when the data is collected from a trajectory.

**Key Words:** Gaussian process, non-stationary regression, chunk spasification

## 1 INTRODUCTION

The Gaussian process (GP) [1], [2] is one of the kernel machines and provides the state-of-art nonparametric Bayesian regression framework commonly used in machine learning. The kernel methods provide a critical setting to address many nonlinear problems in pattern recognition and signal processing [3], [4]. GP, which is a generalization of the Gaussian distribution, is a stochastic process that deals with functions. An essential advantage of GP over other non-Bayesian models is that it explicitly models the system from a probabilistic perspective [5], [6]. In the regression issues, it is capable of representing not only the prediction of the target output, but also the uncertainty of the estimation. Engel [7] et al. propose a recursive implementation based on kernel methods named kernel recursive least-squares (KRLS) algorithm, which outperforms support vector regression both in error rate (MSE criterion) and computational speed in nonlinear regression.

The online kernel methods attempt to predict the target of interest by adapting their parameters when a new data is collected, typically by minimizing a least-squares cost function. In order to make the kernel algorithms feasible, the growth of memory and computational complexities should be greatly reduced, which can be achieved by approximately representing the target utilizing only a subset of bases [8]. As an online kernel method, a standard online GP always collects samples randomly from the input space. While in some examples, e.g., the Internet package transmitting problem and typical reinforcement learning problem [9], [10], a chunk of data may be collected from the trajectory of the system. The sparsification mechanisms proposed in [7], [8], [11] are limited for the input data fed one by one. Therefore, a chunk sparsification algorithm is established in this paper to address the data chunk issue.

Another issue is that the system always gradually varies as time goes by, which means it is a non-stationary system. The ability of tracking non-stationary is critical and essential to model the system in many problems like detecting the changes in sensors networks [12]. However, the standard GP performs poorly on tracking [13], [14]. To address this problem, back-to-the-prior (B2P) forgetting and uncertainty-injection (UI) forgetting are proposed in [13], and exponentially weighted KRLS (EW-KRLS) in [15]. The EW-KRLS performs tracking by utilizing a forgetting factor, which is theoretically capable of capturing the time-varying information. However, it would gradually lose its forgetting ability due to the regularization diminishing. Eventually, it cannot perform tracking any more. Moreover, an ever-growing memory is necessary when tracking, so this method may face both numerical problems and memory problems. The algorithm proposed in [13], namely, kernel recursive least-squares tracker (KRLST) is able to address tracking non-stationary problems using a fixed memory size. But KRLST implements B2P or UI forgetting techniques in the whole input space. In another word, it injects uncertainties to all the inputs and forgets them equally. This method is no longer suitable when one deals with a chunk of data which has three properties. First, the input received is a chunk of data every time when one is at the data receiving procedure. Second, the data closed in time instance are highly correlated within a data chunk. Third, the correlation over different data chunks is low. In this circumstance, it is obviously not a good approach to perform KRLST forgetting techniques because one cannot obtain the whole input space, and KRLST would treat all the historical data equally, which is likely to break prop-

This work is supported by National Natural Science Foundation of China (NSFC) under Grants No. 61273136, No. 61573353 and No. 61533017.

978-1-4673-9714-8/16/$31.00 

erty two and three above. Motivated by this, a perturbed Gaussian process (PGP) method is proposed in this paper to forget different samples with different levels.

PGP is able to pick out the most suitable basis using approximate linear dependence (ALD) criterion [7] from a chunk of data. The non-stationary process is described as a zero mean Gaussian distribution added to the kernel function's posterior. Then, the forgetting technique can be implemented among the different data chunks or within a data chunk.

The main contributions of this paper are summarized as follows:

- A chunk sparsification mechanism is designed to pick out the desirable bases from data chunk which does not require the ever-growing memory compared with traditional KRLS selection mechanism;
- A perturbation method for the standard Gaussian process is established to perform forgetting procedure on the data chunk so as to track the non-stationary system.

The rest of the paper is mainly organized as follows. Section II introduces the fundamentals of Bayesian GP. Section III presents a sparsification method to select bases from a data chunk. PGP algorithm is proposed in section IV. Section V validates the performance of PGP by several experiments, and then the conclusion is drawn in section VI.

## 2 BAYESIAN GAUSSIAN PROCESS

In the standard GP regression, it is always stated that GP is a collection of random variables with respect to inputs $x$, any finite subsets of which have joint Gaussian distribution with latent function $f(x)$ and a positive semi-definite covariance function $k(x, x')$, which is also called kernel function [16]. We can write the GP as $f(x) \sim \mathcal{GP}(m(x), k(x, x'))$, where the mean function $m(x)$ is taken to be zero in this paper.

Following the standard setup, the observation $y$ is the sum of the latent function $f(\cdot)$ on collected input data $\mathbf{x}_t$, which is also named bias vector, and independent zero-mean Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$, namely,

$$y_i = f(x_i) + \epsilon_i. \tag{1}$$

The kernel matrix $\mathbf{K}$ with element $\mathbf{K}_{i,j} = k(x_i, x'_j)$ is the covariance matrix of the latent function. When a new sample $x$ is collected, it would calculate its kernel vector $k(\mathbf{x}_t, x)$ related to $\mathbf{x}_t$, and thus the corresponding observation distribution can be calculated as a conditional distribution given the historical data set with the mean $\mu(x)$ and variance $\sigma(x)$:

$$\mu(x) = k(\mathbf{x}_t, x)^{\mathsf{T}}(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} \tag{2a}$$

$$\sigma^2(x) = k(x, x) - k(\mathbf{x}_t, x)^{\mathsf{T}}(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}k(\mathbf{x}_t, x) + \sigma_n^2, \tag{2b}$$

where $\mathbf{y}$ is the past observations. Van et al. [13] provides a Bayesian derivation of the standard GP. It solves the latent function in the online case with a recursive implementation. It assumes the input data set $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^t$ is collected at time $t$ with latent function $\mathbf{f}_t = [f(x_1), f(x_2), ..., f(x_t)]^{\mathsf{T}} \sim \mathcal{N}(0, \mathbf{K}_t)$. Given $\mathcal{D}_t$, the distribution of latent function $\mathbf{f}_t$ can be described as

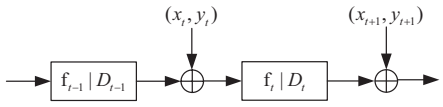


Figure 1: The update diagram of Bayesian GP.

$$p(\mathbf{f}_t|\mathcal{D}_t) = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t). \tag{3}$$

When a new sample $(x_{t+1}, y_{t+1})$ is collected at time $t+1$, it will be added to the data set $\mathcal{D}_{t+1} = \mathcal{D}_t \bigcup (x_{t+1}, y_{t+1})$. Whether it would be added into the bias vector $\mathbf{x}_t$ depends on the approximate linear dependency (ALD) criterion. If it is satisfied, $\mathbf{x}_{t+1} = \mathbf{x}_t \bigcup x_{t+1}$; otherwise, the bias vector will remain the same and $\mathbf{x}_{t+1} = \mathbf{x}_t$. The conditional distribution of latent function can be updated according to Bayesian inference in a sequential form

$$\begin{aligned} p(\mathbf{f}_{t+1}|\mathcal{D}_{t+1}) &= p(f_{t+1}, \mathbf{f}_t|\mathcal{D}_t, y_{t+1}) \\ &= \frac{p(y_{t+1}|f_{t+1}, \mathbf{f}_t, \mathcal{D}_t)p(f_{t+1}|\mathbf{f}_t, \mathcal{D}_t)p(\mathbf{f}_t|\mathcal{D}_t)}{p(y_{t+1}|\mathcal{D}_t)} \\ &= \frac{p(y_{t+1}|f_{t+1})p(f_{t+1}|\mathbf{f}_t)p(\mathbf{f}_t|\mathcal{D}_t)}{p(y_{t+1}|\mathcal{D}_t)}. \end{aligned} \tag{4}$$

Where $y_{t+1}$, which can be computed by (1), is the observation of latent function $f_{t+1} = f(x_{t+1})$ disturbed by a noise $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. The last step of (4) follows that $y_{t+1}$ is independent on $\mathbf{f}_t, \mathcal{D}_t$, and $f_{t+1}$ is independent on $\mathcal{D}_t$. Since the linear combination and conditional distribution of Gaussian distribution is still a Gaussian, by feeding (3) into (4), we can obtain the posterior distribution of latent function, which is still a Gaussian distribution

$$p(\mathbf{f}_{t+1}|\mathcal{D}_{t+1}) = \mathcal{N}(\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}). \tag{5}$$

Therefore, with the posterior distribution of latent function, the predictive distribution $p(f_{t+1}|\mathcal{D}_t)$ can be calculate as

$$p(f_{t+1}|\mathcal{D}_t) = \int p(f_{t+1}|\mathbf{f}_t)p(\mathbf{f}_t|\mathcal{D}_t)d\mathbf{f}_t. \tag{6}$$

The posterior (5) can be used as the prior at the next time step when a new sample is collected. So the latent function can be updated recursively in an online case. This above update diagram of Bayesian GP is shown in Figure 1.

## 3 CHUNK SPARSIFICATION

As described above, the dimension of the bias vector grows at each time step when a new sample is incorporated. Hence, the growth would result in the unbounded increase in the computation and memory requirement. Engel et al. [7] sparsified GP by transforming the data into a high-dimensional reproducing kernel Hilbert space where data can be represented in a linear form. The inputs which are

approximately linear dependent (ALD) on the bias vector will be removed, so only the representative bases would be added to bias vector. The criterion for ALD is expressed as

$$\delta^2 = k(x', x') - k(\mathbf{x}_t, x')^\mathsf{T} \mathbf{K}_t^{-1} k(\mathbf{x}_t, x'). \tag{7}$$

A threshold $\nu$ is used for the criterion. If $\delta^2 > \nu$, it means that the bases in the bias vector cannot represent $x'$ under the ALD criterion. Therefore, the new sample $x'$ will be added into the bias vector. Otherwise, it can be removed with less information loss.

The standard online GP focuses on the problem in which the data are made available one at a time. Here we are interested in the problem where multiple input-output pairs are collected and fed into the GP. Suppose $C$ pairs $\{(x_i, y_i)\}_{i=t}^{t+C-1}$ are grabbed from time $t$, similar to the one-pair-collected case, the posterior distribution of latent function can be calculated by Bayesian inference with respect to these $C$ pairs. Set the memory budget to $M$, then the representative bases can be picked out one-by-one sequentially according to the ALD criterion until the number of bases is up to $M$.

It is desired to select the most representative one from a chunk of data. Here we apply the ALD criterion on the data chunk to obtain the optimal basis

$$x = \underset{x' \in \{x_i\}_{i=t}^{t+C-1}}{\arg\max} (k(x', x') - k(\mathbf{x}_t, x')^\mathsf{T} \mathbf{K}_t^{-1} k(\mathbf{x}_t, x')). \tag{8}$$

After a optimal basis $x_{t+1}$ in chunk observed, the joint distribution of the new latent function $f_{t+1}$ and previous latent functions $\mathbf{f}_t$ is still a Gaussian distribution

$$\begin{bmatrix} \mathbf{f}_t \\ f_{t+1} \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{K}_t & k(\mathbf{x}_t, x_{t+1}) \\ k(\mathbf{x}_t, x_{t+1})^\mathsf{T} & k(x_{t+1}, x_{t+1}) \end{bmatrix}). \tag{9}$$

According to the relationship of Gaussian distribution, the latent function $f_{t+1}$ at the new basis $x_{t+1}$ given the previous latent functions $\mathbf{f}_t$ can be described as

$$p(f_{t+1}|\mathbf{f}_t) = \mathcal{N}(\hat{f}_{t+1}, \hat{\gamma}^2_{f_{t+1}}), \tag{10}$$

where $\hat{f}_{t+1} = \mathbf{a}_{t+1}^\mathsf{T} \mathbf{f}_t, \mathbf{a}_{t+1} = \mathbf{K}_t^{-1} k(\mathbf{x}_t, x_{t+1})$, and $\hat{\gamma}^2_{f_{t+1}} = k(x_{t+1}, x_{t+1}) - k(\mathbf{x}_t, x_{t+1})^\mathsf{T} \mathbf{K}_t^{-1} k(\mathbf{x}_t, x_{t+1})$. Assume that the posterior at time $t$ is known $p(\mathbf{f}_t|\mathcal{D}_t) = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$. The predictive distribution of the new observation $y_{t+1}$ given past data can be computed as

$$\begin{aligned} p(y_{t+1}|\mathcal{D}_t) &= \int p(y_{t+1}|f_{t+1}) p(f_{t+1}|\mathbf{f}_t) p(\mathbf{f}_t|\mathcal{D}_t) d\mathbf{f}_t df \\ &= \mathcal{N}(\hat{y}_{t+1}, \hat{\sigma}^2_{y_{t+1}}), \end{aligned} \tag{11}$$

where $\hat{y}_{t+1} = \mathbf{a}_{t+1}^\mathsf{T} \boldsymbol{\mu}_t$ , $\hat{\sigma}^2_{y_{t+1}} = \hat{\sigma}^2_{f_{t+1}} + \sigma_n^2$, and $\hat{\sigma}^2_{f_{t+1}} = \hat{\gamma}^2_{f_{t+1}} + \mathbf{a}_{t+1}^\mathsf{T} \boldsymbol{\Sigma}_t \mathbf{a}_{t+1}$. According to the definition (1), we have the observation $y_{t+1}$ given new latent function $f_{t+1}$

$$p(y_{t+1}|f_{t+1}) = \mathcal{N}(f_{t+1}, \sigma_n^2). \tag{12}$$

Therefore, all involved distributions (10)-(12) appearing in (4) are Gaussian distributions. Based on these relationships, the posterior can be computed after a new basis is selected, and be expressed as

$$p(\mathbf{f}_{t+1}|\mathcal{D}_{t+1}) = \mathcal{N}(\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}) \tag{13a}$$

$$\boldsymbol{\mu}_{t+1} = \begin{bmatrix} \boldsymbol{\mu}_t \\ \hat{f}_{t+1} \end{bmatrix} + \frac{y_{t+1} - \hat{y}_{t+1}}{\hat{\sigma}^2_{y_{t+1}}} \begin{bmatrix} \mathbf{h}_{t+1} \\ \hat{\sigma}^2_{f_{t+1}} \end{bmatrix} \tag{13b}$$

$$\boldsymbol{\Sigma}_{t+1} = \begin{bmatrix} \boldsymbol{\Sigma}_t & \mathbf{h}_{t+1} \\ \mathbf{h}_{t+1}^\mathsf{T} & \hat{\sigma}^2_{f_{t+1}} \end{bmatrix} - \frac{1}{\hat{\sigma}^2_{y_{t+1}}} \begin{bmatrix} \mathbf{h}_{t+1} \\ \hat{\sigma}^2_{f_{t+1}} \end{bmatrix} \begin{bmatrix} \mathbf{h}_{t+1} \\ \hat{\sigma}^2_{f_{t+1}} \end{bmatrix}^\mathsf{T}, \tag{13c}$$

where $\mathbf{h}_{t+1} = \boldsymbol{\Sigma}_t \mathbf{a}_{t+1}$.

The inverse of the kernel matrix including a new input can be updated using

$$\mathbf{K}_{t+1}^{-1} = \begin{bmatrix} \mathbf{K}_t^{-1} & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & 0 \end{bmatrix} + \frac{1}{\delta^2} \begin{bmatrix} \mathbf{a}_{t+1} \\ -1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_{t+1} \\ -1 \end{bmatrix}^\mathsf{T}. \tag{14}$$

Then we can use (8) again to select the optimal basis based on the new bias vector. This procedure loops until all the samples in the chunk can be represented by the bias vector approximately, namely,

$$\max_{x' \in \{x_i\}_{i=t}^{t+C-1}} (k(x', x') - k(\mathbf{x}_t, x')^\mathsf{T} \mathbf{K}_t^{-1} k(\mathbf{x}_t, x')) \leq \nu. \tag{15}$$

Note that in the chunk sparsification procedure, the sample $x_{t+1}$ added to bias vector $\mathbf{x}_t$ may not be the input at time $t+1$, but only represents the input satisfied (8). So assuming a $C$ pairs data chunk fed into the sparsification procedure, the number of new bases and latent functions may be less than $C$ when this procedure ends.

The GP is suitable to solve the stationary regression problems. As the uncertainty of latent function decreases after each update, the magnitude of update for the latent function will decrease as more and more samples collected. This mechanism is disadvantageous for tracking a non-stationary system [13], [14]. To address this problem, back-to-the-prior (B2P) forgetting and uncertainty-injection (UI) forgetting is proposed in [13], or exponentially weighted forgetting in [15]. All these approaches will perform forgetting in the whole input space. Here we are interested in forgetting with respect to a chunk of points or a single point. For this purpose, the perturbed Gaussian process is proposed in the next section.

## 4 PERTURBED GAUSSIAN PROCESS

Take a deep analysis of the problem, then we are able to find that the standard GP cannot track the non-stationary system. The posterior distribution $p(\mathbf{f}_t|\mathcal{D}_t)$ at time step $t$ would be used to be the prior at the next time step, which would make the Gaussian regression as a stationary system so that it cannot model the non-stationary system accurately. Since the character of non-stationary system cannot be measured and is hard to model, here we would model this character as a perturbation to the standard GP, i.e. it is assumed that the GP is perturbed by a noise during the time that the role of $p(\mathbf{f}_t|\mathcal{D}_t)$ changes from posterior to prior. Hence, we can formulate this procedure as

$$\mathbf{f}'_t|\mathcal{D}_t = \mathbf{f}_t|\mathcal{D}_t + W, \tag{16}$$

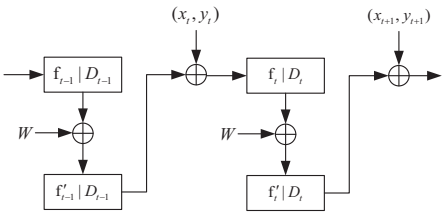


Figure 2: The update diagram of perturbed Gaussian process.

where $W$ is the perturbation to the GP. Here it is assumed to be a Gaussian white noise $\mathcal{N}(0, \mathbf{R})$ with covariance matrix $\mathbf{R}$. As a result, $p(\mathbf{f}'_t|\mathcal{D}_t)$ would play the role of prior distribution instead of $p(\mathbf{f}_t|\mathcal{D}_t)$ at the next step when a new sample is collected. The update diagram of this procedure is shown in Figure 2.

The forgetting method can be implemented with respect to a chunk of points or a single point, and is presented as following.

### 4.1 One-point Perturbation

It is assumed that the perturbation is injected into the vicinity of point $x_{t+1}$ when a new sample $(x_{t+1}, y_{t+1})$ is collected. As the GP relates the output by the kernel function of input, the perturbation injected at the point $x_{t+1}$ would affect the vicinal points. Assume the perturbation injected at the point $x_{t+1}$ is $\sigma_p^2 k(x_{t+1}, x_{t+1})$, where $\sigma_p^2$ is the perturbed factor.

We can see from (6) that the perturbation into the predictive distribution of $f_{t+1}$ can be realized by feeding the noise into $\mathbf{f}_t|\mathcal{D}_t$. According to the relationship of GP, the latent function on the bias vector can be predicted on $f(x_{t+1})$ as $f(\mathbf{x}_t) = k(\mathbf{x}_t, x_{t+1})k(x_{t+1}, x_{t+1})^{-1}f(x_{t+1})$. As a result, the perturbation at $x_{t+1}$ will cause a corresponding perturbation to the bias vector.

The covariance matrix of the perturbation can be formalized as

$$\mathbf{R} = \sigma_p^2 \mathbf{k}_t k(x_{t+1}, x_{t+1})^{-1} \mathbf{k}_t^{\mathsf{T}}. \tag{17}$$

where $\mathbf{k}_t = k(\mathbf{x}_t, x_{t+1})$. Hence, the perturbed posterior distribution is

$$p(\mathbf{f}'_t|\mathcal{D}_t) = \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t + \mathbf{R}). \tag{18}$$

Substituting $p(\mathbf{f}'_t|\mathcal{D}_t)$ in (6) and solving the integral, we obtain the implied posterior GP

$$\begin{aligned} f(x)|\mathcal{D}_t \sim \mathcal{GP}(&\mathbf{a}_{t+1}^{\mathsf{T}}\boldsymbol{\mu}_t, \\ &k_{t,t} + \mathbf{a}_{t+1}^{\mathsf{T}}(\boldsymbol{\Sigma}_t + \sigma_p^2 \mathbf{k}_t \mathbf{k}_t^{\mathsf{T}} - \mathbf{K}_t)\mathbf{a}_{t+1}). \end{aligned} \tag{19}$$

However, substituting the non-perturbed posterior distribution $p(\mathbf{f}_t|\mathcal{D}_t) = \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)$ in (6) and solving the integral, we obtain the posterior GP

$$\begin{aligned} \breve{f}(x)|\mathcal{D}_t \sim \mathcal{GP}(&\mathbf{a}_{t+1}^{\mathsf{T}}\boldsymbol{\mu}_t, \\ &k_{t,t} + \mathbf{a}_{t+1}^{\mathsf{T}}(\boldsymbol{\Sigma}_t - \mathbf{K}_t)\mathbf{a}_{t+1}). \end{aligned} \tag{20}$$

Comparing to (20), the term $\sigma_p^2 \mathbf{k}_t \mathbf{k}_t^{\mathsf{T}}$ in (19) is able to increase the uncertainty of the GP $f(x)|\mathcal{D}_t$ and perform forgetting of the old samples.

### 4.2 Chunk Perturbation

As mentioned previously, data sometimes may be collected in a form of chunk, like the Internet package. Similar to the one-point perturbation, we will transform the perturbation to the bias vector into a perturbation to a chunk of data. Taking the relationship of the inputs, the perturbation injected into the data chunk can be assumed as $\sigma_p^2 \mathbf{K}(\mathbf{x}', \mathbf{x}')$ where the $\mathbf{x}'$ is the data chunk. So we are able to obtain the corresponding GP on the bias vector as $f(\mathbf{x}_t) = \mathbf{K}(\mathbf{x}_t, \mathbf{x}')\mathbf{K}^{-1}(\mathbf{x}', \mathbf{x}')f(\mathbf{x}')$. This perturbation would cause a corresponding perturbation to the bias vector with covariance matrix

$$\mathbf{R} = \sigma_p^2 \mathbf{K}(\mathbf{x}_t, \mathbf{x}')\mathbf{K}^{-1}(\mathbf{x}', \mathbf{x}')\mathbf{K}(\mathbf{x}_t, \mathbf{x}')^{\mathsf{T}}. \tag{21}$$

Hence, the perturbed posterior distribution is

$$p(\mathbf{f}'_t|\mathcal{D}_t) = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t + \sigma_p^2 \mathbf{K}(\mathbf{x}_t, \mathbf{x}')\mathbf{K}^{-1}(\mathbf{x}', \mathbf{x}')\mathbf{K}(\mathbf{x}_t, \mathbf{x}')^{\mathsf{T}}). \tag{22}$$

In some cases, if we want to perform the forgetting methods on the whole input space, then we can simply replace the $\mathbf{x}'$ in (22) as all the collected data. However, the number of points in the whole input space is numerous so that handling the inverse of $\mathbf{K}_t$ is impossible. Since the bias vector is the representative point, we have $k(x', x') - k(\mathbf{x}_t, x')^{\mathsf{T}}\mathbf{K}^{-1}(\mathbf{x}', \mathbf{x}')k(\mathbf{x}_t, x') < \nu$, for any $x' \in \mathbf{x}'$. Here $\mathbf{x}'$ represents all the collected data, not the data chunk. Because $\nu$ is a very small number, so it can be derived as

$$\mathbf{K}(\mathbf{x}_t, \mathbf{x}')\mathbf{K}^{-1}(\mathbf{x}', \mathbf{x}')\mathbf{K}(\mathbf{x}_t, \mathbf{x}')^{\mathsf{T}} \approx \mathbf{K}_t. \tag{23}$$

$\mathbf{K}_t$ is much more easy to tract because its dimension is rather low, and it can be calculated in a recursive way.

Therefore, forgetting all the history information can be achieved by feeding a perturbation to the bias vector with covariance matrix

$$\mathbf{R} = \sigma_p^2 \mathbf{K}_t. \tag{24}$$

The corresponding perturbed posterior distribution is

$$p(\mathbf{f}'_t|\mathcal{D}_t) = \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t + \sigma_p^2 \mathbf{K}_t). \tag{25}$$

## 5 SIMULATION

In order to test and validate the effectiveness of the proposed approach, we will test it on two different systems: a stationary system regression and a non-stationary system regression.The compared algorithms include approximate linear dependency KRLS (KRLS-ALD), KRLS-tracker (KRLS-T) and PGP.

### 5.1 Stationary System Regression

In the first experiment, we train three algorithms to perform regression on Sinc-Linear function

$$y = \frac{\sin x_1}{x_1} + \frac{x_2}{10}, \qquad x_1, x_2 \in [-10, 10]. \tag{26}$$

The RBF kernel function is selected as the covariance function for the GPs

$$k(x, x') = \alpha \exp(-\frac{\|x - x'\|^2}{2l^2}). \tag{27}$$

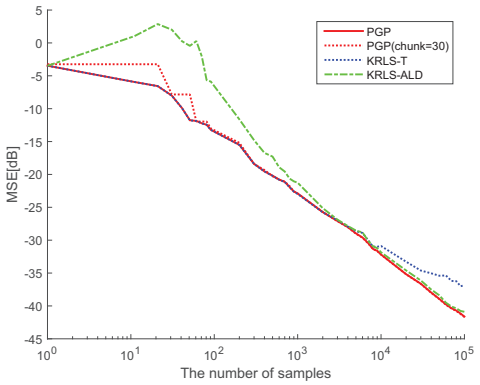


Figure 3: The performance of different algorithms with respect to sample size on the Sinc-Linear function. The memory size of PGP, PGP(chunk) and KRLS-T is set to 80, and KRLS-ALD's memory size is ever-growing.

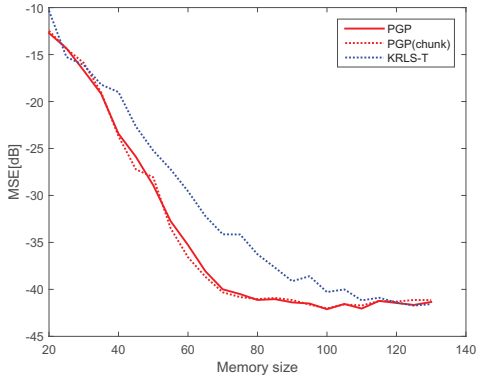


Figure 4: The performance comparison with respect to memory size on the Sinc-Linear function.

In this experiment, the PGP algorithm consists of two cases. In the first case, the input data is a single point $(x_t, y_t)$ at time $t$ . And in the second case, the data chunk with size $C = 30$ is received every $C$ steps. For all algorithms, we determine the hyperparameters by the full GP offline [2] with $\alpha = 1, l = 4.4$. The same hyperparameters are shared among the three algorithms. In addition, the perturbed factor for these algorithms are set to zero since it is a stationary system. The memory budget for all the algorithms are set to $M = 80$. The results of mean-squared-error (MSE) is shown in Figure 3. The MSE is averaged over 50-independent randomly generated training sets. The PGP line corresponds to the first perturbed case described above while PGP(chunk) is the second case. It shows that the PGP and KRLS-T perform better than KRLS-ALD and PGP(chunk) at the beginning. The reason is that KRLS-ALD discards the samples which do not satisfy the ALD criterion, while PGP chooses the bias vector in advance and KRLS-T almost adds all the collected samples into the bias vector at the beginning. The PGP(chunk) starts to work only when the first $C$ samples are collected together so the MSE is high at the beginning.But it performs good after the two data chunks received. After some of samples (3000)

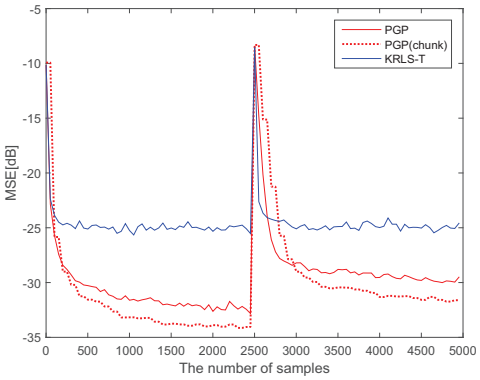


Figure 5: MSE performance comparison of different tracking algorithms on a communications channel that shows an abrupt change at $t = 2500$.

are collected, the KRLS-T degrades because of the switch of the bias vector, which means the loss of information. However, PGP, PGP(chunk) and KRLS-ALD keep the bias vector stationary, and achieve the better performance as more and more samples collected. KRLS-ALD can achieve the same performance because of its ever-growing memory. The performance comparison of three algorithms with respect to memory size is presented in Figure 4. We do not compare the KRLS-ALD algorithm since it is a memory ever-growing algorithm. The performance is averaged over 50-independent randomly generated training sets. We are able to tell from Figure 4 that PGP and PGP(chunk) can performs better than KRLS-T with the same memory size, especially when memory size is not very big. This means that we can use the information more effectively when the memory resources are limited. Moreover, the MSE of PGP and PGP(chunk) is around -41 dB and KRLS-T's is around -36 dB by setting memory size to 80. This result is consistent with the result in Figure 3.

### 5.2 Non-stationary Regression

In this experiment, we would like to study the capability of several tracking algorithms to re-converge after a model switch. All of the algorithms will be tested on the problem of nonlinear system identification of a communication channel with parameters abruptly changed during the process. The communication channel consists of a linear finite impulse response channel followed by a tanh function. It can be depicted as

$$y_t = \tanh(\mathbf{h}^{\mathsf{T}}\mathbf{x}_t), \tag{28}$$

where $\mathbf{x}_t = [x_t, x_{t-1}, ..., x_{t-4}]^{\mathsf{T}}$ and $y_t$ are the input and output of the system at time $t$ respectively. The impulse response $\mathbf{h}_1 = [1.000, -0.3817, -0.1411, 0.5789, 0.1910]^{\mathsf{T}}$ during the first 2500 iterations, and it abruptly changed to $\mathbf{h}_2 = [1.000, -0.0870, 0.9852, -0.2826, -0.1711]^{\mathsf{T}}$ at the 2501 iteration. A signal $x_t \in \mathcal{N}(0, 0.5)$ is fed into the channel and the corresponding measured output is disturbed by a zero-mean Gaussian noise $\mathcal{N}(0, 0.15)$.
Since we are comparing tracking capability, only the KRLS-T, PGP and PGP(chunk) are used to identify the dy-

namics here. The forgetting technique of KRLS-T we applied here is back-to-prior (B2P) method. The RBF kernel function (27) with $l = 2$ is employed here as the kernel function for all the algorithms. The measured noise $\sigma_n$ and the forgetting factor $\sigma_p$ is set to 0.05 and 0.025 respectively. The budget $M = 100$, and the chunk size $C = 100$.

We can tell from Figure 5 that all three algorithms are capable to track the abrupt change of the communication channel. Moreover, PGP(chunk) can obtain the best performance among three algorithms. PGP(chunk) can update its bias vector only when a whole chunk of data is collected, which leads to a stair-like line and looks badly at the beginning. But as more data chunks collected, this drawback can be naturally addressed. In additional experiments, we find that PGP and PGP(chunk) do not have the numerical precision problem. However, only the B2P algorithm in KRLS-T family is feasible, while the uncertainty-injection (UI) method would face the numerical problem if we do not add a jitter term. Hence, PGP is more stable.

## 6 CONCLUSION

In this paper, a PGP algorithm is proposed to study the tracking problems of the non-stationary system in which the input data are often received as data chunks. A chunk sparsification mechanism is established to pick out the desirable bases from data chunk to enrich bias vector. Compared with the traditional ALD mechanism, this procedure can overcome the memory ever-growing problem and achieve good regression performance. Simulations are implemented in two experiments and have corroborated that PGP can make an efficient use of data and obtain better performance when data are fed chunk by chunk. Moreover, compared with KRLS-T, lower regression error can be achieved by PGP when memory resources are limited. Future work involves the combination of PGP and reinforcement learning in a sample efficient form.

## REFERENCES


[1] J. M. Bernardo and A. F. Smith, Bayesian Theory. John Wiley & Sons, 1994.

[2] C. E. Rasmussen, C. K. I. Williams, Gaussian process for machine learning, Cambridge: MIT Press, 2006.

[3] B. Schölkopf and A. J. Smola, Learning with kernels: support vector machines, regularization, optimization, and beyond, Cambridge: MIT Press, 2001.

[4] J. Shawe Taylor and N. Cristianini, Kernel methods for pattern analysis. London, U.K.: Cambridge Univ. Press, 2004.

[5] Zhongpu Xia and Dongbin Zhao, Online reinforcement learning by Bayesian inference, in International Joint Conference on Neural Networks, Killarney, Ireland, 2015.

[6] Zhongpu Xia, Dongbin Zhao, Online bayesian reinforcement learning by gaussian processes, IET Control Theory and Applications, doi: 10.1049/iet-cta.20150669.

[7] Y. Engel, S. Mannor, and R. Meir, The kernel recursive least-squares algorithm, IEEE Trans. on Signal Processing, Vol.23, No.8, 2275-2285, 2004.

[8] W. Liu, I. Park, and J. C. Príncipe, An information theoretic approach of designing sparse kernel adaptive filters, IEEE Trans. Neural Networks, Vol.20, No.12, 1950-1961, 2009.

[9] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction, Cambridge: MIT Press, 1998.

[10] Dongbin Zhao, Zhongpu Xia, Ding Wang, Model-free optimal control for affine nonlinear systems based on action dependent heuristic dynamic programming with convergency analysis, IEEE Trans. on Automation and Science Engineering. vol. 12, no. 4, pp. 1461-1468, 2015.

[11] L. Csató and M. Opper, Sparse representation for Gaussian process models, in Advances in Neural Information Processing Systems, Cambridge, MIT Press, 444-450, 2001.

[12] Cesare Alippi, Derong Liu, Dongbin Zhao, Li Bu, Detecting and reacting to changes in sensing units: the active classifier case, IEEE Trans. on System, Man and Cybernetics Part A-Systems, vol. 44, no. 3, pp. 353-362, 2014.

[13] S. Van Vaerenbergh, M. Lázaro-Gredilla, and I. Santamaría, Kernel recursive least-squares tracker for time-varying regression, IEEE Trans. on Neural Networks and Learning Systems, Vol.23, No.8, 1313-1326, 2012.

[14] R. Grande, T. Walsh, and J. How, Sample efficient reinforcement learning with gaussian processes, in Proceedings of the 31st International Conference on Machine Learning, 1332-1340, 2014.

[15] W. Liu, I. Park, Y. Wang, and J. C. Príncipe, Extended kernel recursive least squares algorithm, IEEE Trans. on Signal Processing. Vol.57, No.10, 3801-3814, 2009.

[16] V. N. Vapnik, The nature of statistical learning theory, New York, NY: Springer-Verlag, 1995.