Journal of Frame and Matrix Theory (JFMT) 2 (2025) No. 2, 56–74
Online ISSN: 3041-8526
Published by Faculty of Mathematics and Computer Science
Hakim Sabzevari University
http://dx.doi.org/10.22034/jfmt.2025.504073.1024

Restricted gaussian process for predicting latent functions

Mehdi Zaferanieh[a,*], Alireza Shafiee-Fard[b], Morteza Jafarzadeh[a], Hesam Hasanpoor[b]

[a] *Department of Mathematics and Computer Science, Hakim Sabzevari University, Sabzevar, Iran.*

[b] *Department of Information Technology and Computer Engineering, Sabzevar Branch, Islamic Azad University, Sabzevar, Iran.*

(Communicated by Saeed Karimi)

Abstract

In this paper, we evaluate the gaussian process (GP) as a powerful toolkit for nonparametric classification, and regression. Unlike traditional parametric methods, GPs provide a distribution over functional spaces to model the uncertainty in predictions. The relationship between GP and input correlation kernel functions are illustrated, and some different kernels are introduced. Moreover, practical applications of GP for large scale problems using the Nyström approximation have been studied, and several numerical examples have been provided to verify the validity and efficiency of the proposed method. The implementation codes have been executed in *Python* using $Scikit-learn$ library.

Keywords: Gaussian process, Predicting latent functions, Global optimization, Nyström approximation.

*Corresponding author
Email addresses: m.zaferanieh@hsu.ac.ir (Mehdi Zaferanieh), shafiee.fard@gmail.com (Alireza Shafiee-Fard), m.jafarzadeh@hsu.ac.ir (Morteza Jafarzadeh), hesam.hasanpoor@iaus.ac.ir (Hesam Hasanpoor)

Received: 2 February 2025 *Accepted:* 26 August 2025

MSC 2020: Primary: 60G15; Secondary: 90C26.

1 Introduction

The GPs have piqued the interest of many machine learning researchers due to their flexibility and ability to handle uncertain predictions, [30]. In the core concept of GPs, certain distributions on predicted functions are favored for use as non-parametric toolkits over Neural Networks (NN)s, and classical regression methods, as well, [21]. Also, GP is a standard choice model in Bayesian Optimization (BO) in which uncertainty about gathering data is required, [2, 6]. Despite the wide application of GPs, understanding their concepts involving multivariate normal distributions, covariance (kernels), and nonparametric models can be challenging due to their complexities, [16]. A multivariate normal distribution (MND) is a generalization of the univariate Gaussian distribution to multiple dimensions and is fully specified by a mean vector and a covariance matrix. This distribution forms the mathematical foundation of GPs, as any finite collection of function values in a GP follows a joint MND.

The GP methods rely on suitable mean and kernel functions, leads to an efficient algorithm implementation with cubic run-time complexity. To reduce the implementation complexity, four main approaches are commonly used including the "induced point" [27], "stochastic gradient descent" (SGD) [15], Nyström approximation [29], and preconditioning [1, 28], to inverse kernel functions. The induced point method employs a subset of data points instead of the whole data and would be merged by Nyström approximation and singular value decomposition (SVD) methods. The induced points may be selected randomly or through some more sophisticated methods that consider the distribution of data, [8]. The SGD method also works on a subset of given training data, but it updates the necessary parameter of the GP by considering the gradient of the subset selected data, see [7, 19]. The SGD particularly is useful when data exhibit some correlation, [5]. This capability is useful in real application where the data points are often interrelated, and such traditional methods as gradient descent (GD) or lagrange relaxation may struggle with this scenario, [18, 25].

The other limitation is quadratic memory requirement which makes GPs impractical for large-scale problems, [13]. In this cases, sparse gaussian process (SGP) is a favorable alternative where utilizes a set of induced points to approximate the posterior distribution. In this paper, we explore the fundamental concepts of GPs, covering essential topics such as prediction functions and updating weights to align with approximation methods. Additionally, we introduce a hybrid induced point method and Nyström approximation to reduce the run time complexity as well as necessary memory requirement. Also, the SVD method is introduced as alternative matrix approximation method.

The GPs have piqued the interest of many machine learning researchers due to their flexibility and ability to handle uncertain predictions, [30]. In the core concept of GPs, certain distributions on predicted functions are favored for use as non-parametric toolkits over Neural Networks (NN)s, and classical regression methods, as well, [21]. Also, GP is a standard choice model in Bayesian Optimization (BO) in which uncertainty about gathering data is required, [2, 6]. Despite the wide application of GPs, understanding their concepts involving multivariate normal distributions, covariance (kernels), and nonparametric models can be challenging due to their complexities, [16].

The GP methods rely on suitable mean and kernel functions, leads to an efficient algorithm implementation with cubic run-time complexity. To reduce the implementation complexity, four main approaches are commonly used including the "induced point" [27], "stochastic gradient descent" (SGD) [15], Nyström approximation [29], and preconditioning [1, 28], to inverse kernel functions. The induced point method employs a subset of data points instead of the whole data and would be merged by Nyström approximation and singular value decomposition (SVD) methods. The induced points may be selected randomly or through some more sophisticated methods that consider the distribution of data, [8]. The SGD method also works on a subset of given training data, but it updates the necessary parameter of the GP by considering the gradient of the subset selected data, see [7, 19]. The SGD particularly is useful when data exhibit some correlation, [5]. This capability is useful in real application where the data points are often interrelated, and such traditional methods as gradient descent (GD) or lagrange relaxation may struggle with this scenario, [18, 25].

The other limitation is quadratic memory requirement which makes GPs impractical for large-scale problems, [13]. In this cases, sparse gaussian process (SGP) is a favorable alternative where utilizes a set of induced points to approximate the posterior distribution. In this paper, we explore the fundamental concepts of GPs, covering essential topics such as prediction functions and updating weights to align with approximation methods. Additionally, we introduce a hybrid induced point method and Nyström approximation to reduce the run time complexity as well as necessary memory requirement. Also, the SVD method is introduced as alternative matrix approximation method.

2 Multivariate Normal Distribution

A multivariate Normal Distribution (MND) has been defined by a mean vector, and a covariance matrix. The covariance matrix of some given data is symmetric and semi-positive definite. If $\vec{x}$ is a stochastic variable, then the joint probability density function is given as:

$$P(\vec{x}|\vec{\mu},\Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp(-\frac{1}{2}(\vec{x}-\vec{\mu})^T\Sigma^{-1}(\vec{x}-\vec{\mu})) \tag{2.1}$$

where $\vec{\mu}$ is the mean vector from dimension n, and Σ is a $n \times n$ covariance matrix. Next, we propose the regression function as a linear combination of some basis function. Let $y = f(\vec{x}) + \varepsilon$, where $f(\vec{x}) = \vec{x}^T W$ and $\vec{x}$ is an input of features for real-valued function $f(.)$ and ε is random noise variable, with a mean equal zero and variance σ_n^2. Here, σ_n^2 is the variance of given train data. By this assumption, the conditional probability of $\vec{y}|\vec{x}, \vec{w}$ is given by relation (2.2):

$$p(\vec{y}|\vec{x}, \vec{w}) = \Pi_{i=1}^{n} p(y_i|x_i, w) = \Pi_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp(-\frac{(y_i - \vec{x_i}^T \vec{w})^2}{2\sigma_2^n}) \tag{2.2}$$
$$= \frac{1}{(2\pi\sigma_n^2)^{\frac{n}{2}}} \exp(-\frac{1}{2\sigma_n^2} \|\vec{y} - \vec{x}^T \vec{w}\|^2) = N(\vec{x}^T \vec{w}, \sigma_n^2 I)$$

In relation (2.2), all random variables are considered to be independent. Next, for predicting the objective function at the point $\vec{x^*}$ using class D of training data, we have

$$E(\vec{w}^T \vec{x^*}|D, \vec{x^*}) = \vec{x^*} E(\vec{w}|D) = \vec{x^{*T}} \vec{w_0}, \tag{2.3}$$

where $E(\vec{w}|D)$ is considered to be $\vec{w_0}$. The variance of $\vec{w}^T \vec{x^*}|D, \vec{x^*}$ is given by relation (2.4):

$$\begin{aligned} var(\vec{w}^T \vec{x^*}|D, \vec{x^*}) &= E((\vec{w}^T \vec{x^*} - \vec{w_0}^T \vec{x^*})^2|D, \vec{x^*}) \\ &= E(\vec{x^{*T}}(\vec{w} - \vec{w_0})(\vec{w} - \vec{w_0})^T \vec{x^*}|D, \vec{x^*}) \\ &= \vec{x^{*T}} E((\vec{w} - \vec{w_0})(\vec{w} - \vec{w_0})^T|D)\vec{x^*} = \vec{x^{*T}} cov(\vec{w}|D)\vec{x^*} \end{aligned} \tag{2.4}$$

Therefore, expected value and variance for a new point $\vec{x^*}$ are associated to the mean value of vector weight w and its covariance with respect to the train data, denoted by class D. In relations (2.3) and (2.4), we must examine $\vec{w}|D$ to obtain the expected mean value and variance of the test data.

3 Prior and Posterior Distributions

Using Bayes' theorem, one can verify the probability of a hypothesis using training data, as discussed in [11]. Its mathematical formulation is given by relation (3.1):

$$f(\vec{\theta}|\vec{x}) = \frac{f(\vec{\theta}) f(\vec{x}|\vec{\theta})}{f(\vec{x})} \Rightarrow \text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}. \tag{3.1}$$

Here, $\vec{\theta}$ is the vector of unknown parameters to be estimated, $f(\vec{\theta})$ is the prior distribution over $\vec{\theta}$, $f(\vec{x}|\vec{\theta})$ is the likelihood function of the observed data $\vec{x}$, and $f(\vec{x})$ is the marginal distribution of the training data, given in continuous form as:

$$f(\vec{x}) = \int f(\vec{x}|\vec{\theta}) g(\vec{\theta})\, d\vec{\theta}. \tag{3.2}$$

Using Bayes' rule and relation (2.2), the posterior distribution of the weight vector $\vec{w}$ is:

$$\begin{aligned} p(\vec{w}|\vec{y},\vec{x}) &= \frac{p(\vec{y}|\vec{w},\vec{x})p(\vec{w})}{p(\vec{y}|\vec{x})} && (3.3)\\ &\propto \exp\left(-\frac{1}{2\sigma_n^2}(\vec{y}-\vec{X}^T\vec{w})^T(\vec{y}-\vec{X}^T\vec{w})\right)\exp\left(-\frac{1}{2}\vec{w}^T\vec{\Sigma}_p^{-1}\vec{w}\right)\\ &\propto \exp\left(-\frac{1}{2}(\vec{w}-\overline{\vec{w}})^T\left(\frac{1}{\sigma_n^2}\vec{X}\vec{X}^T+\vec{\Sigma}_p^{-1}\right)(\vec{w}-\overline{\vec{w}})\right). && (3.4)\end{aligned}$$

In relation (3.3), the term $p(\vec{y}|\vec{x})$ is independent of $\vec{w}$ and is omitted. The last term indicates a multivariate normal distribution with mean:

$$\overline{\vec{w}} = \frac{1}{\sigma_n^2}\left(\frac{1}{\sigma_n^2}\vec{X}\vec{X}^T+\vec{\Sigma}_p^{-1}\right)^{-1}\vec{X}\vec{y}, \tag{3.5}$$

and covariance matrix:

$$\vec{A}^{-1} = \left(\frac{1}{\sigma_n^2}\vec{X}\vec{X}^T+\vec{\Sigma}_p^{-1}\right)^{-1}. \tag{3.6}$$

Hence, the conditional probability distribution $\vec{w}|\vec{X},\vec{y}$ is given as:

$$p(\vec{w}|\vec{X},\vec{y}) \sim \mathcal{N}\left(\overline{\vec{w}} = \frac{1}{\sigma_n^2}\vec{A}^{-1}\vec{X}\vec{y}, \vec{A}^{-1}\right). \tag{3.7}$$

Next, we proceed to more advanced cases where the input and output data are related via nonlinear functions.

3.1 Projecting Inputs to the Feature Spaces

Linear regression (LR) has some limitations in recognizing nonlinear interactions between inputs and their corresponding outputs within an acceptable error range. Hence, utilizing basis functions to lift the input data to a higher-dimensional space is beneficial; see [23].

As an example, consider a function $\vec{\phi} : \mathbb{R} \to \mathbb{R}^n$ defined as $\vec{\phi}(x) = (1, x, x^2, \ldots, x^n)$. Then, the prediction function—similar to the linear case—is given by relation (3.8) to handle

nonlinear scenarios:

$$f(x) = \vec{\Phi}(\vec{X})^T \vec{w} + \xi, \tag{3.8}$$

and

$$p(f^*|x^*, \vec{X}, \vec{y}) \simeq \mathcal{N}\left(\frac{1}{\sigma_n^2}\vec{\phi}(x^*)^T \vec{A}^{-1}\vec{\Phi}\vec{y}, \vec{\phi}(x^*)^T \vec{A}^{-1}\vec{\phi}(x^*)\right), \tag{3.9}$$

where $\vec{A} = \frac{1}{\sigma_n^2}\vec{\Phi}\vec{\Phi}^T$ and $\vec{\Phi} = \vec{\Phi}(\vec{X})$. If we define $\vec{K} = \vec{\Phi}^T\vec{\Sigma}_p\vec{\Phi}$ as a kernel trick, where $\vec{\Sigma}_p$ is a positive diagonal matrix, then the following relations hold:

$$\begin{aligned} \frac{1}{\sigma_n^2}\vec{\Phi}(\vec{K} + \sigma_n^2\vec{I}_{n\times n}) = \frac{1}{\sigma_n^2}\vec{\Phi}(\vec{\Phi}^T\vec{\Sigma}_p\vec{\Phi} + \sigma_n^2\vec{I}_{n\times n}) = \vec{A}\vec{\Sigma}_p\vec{\Phi} \\ \Rightarrow \frac{1}{\sigma_n^2}\vec{A}^{-1}\vec{\Phi} = \vec{\Sigma}_p\vec{\Phi}(\vec{K} + \sigma_n^2\vec{I}_{n\times n})^{-1}. \end{aligned} \tag{3.10}$$

By substituting result (3.10) into expression (3.9), we obtain:

$$\begin{aligned} p(f^*|x^*, \vec{X}, \vec{y}) \simeq \mathcal{N}(\frac{1}{\sigma_n^2}\vec{\phi}(x^*)^T\vec{\Sigma}_p\vec{\Phi}(\vec{K} + \sigma_n^2\vec{I}_{n\times n})^{-1}\vec{y}, \\ \vec{\phi}_*\vec{\Sigma}_p\vec{\phi}_*^T - \vec{\phi}_*^T\vec{\Sigma}_p\vec{\Phi}(\vec{K} + \sigma_n^2\vec{I}_{n\times n})^{-1}\vec{\Phi}^T\vec{\Sigma}_p\vec{\phi}_*), \end{aligned} \tag{3.11}$$

where $\vec{\phi}_* = \vec{\phi}(x^*)$. In relation (3.11), if we define $k(x, x') = \vec{\phi}(x)^T\vec{\Sigma}_p\vec{\phi}(x')$, then the predictive distribution at a new point x^* is given by:

$$p(f^*|x^*, \vec{X}, \vec{y}) \simeq \mathcal{N}(\vec{k}_{*n}\vec{K}_{nn}^{-1}\vec{y},\ \vec{k}_{**} - \vec{k}_{*n}\vec{K}_{nn}^{-1}\vec{k}_{n*}). \tag{3.12}$$

Therefore, the parameters of the normal distribution, $\mu(x^*)$ and $\sigma^2(x^*)$, are:

$$\mu(x^*) = \vec{k}_{*n}\vec{K}_{nn}^{-1}\vec{y}, \tag{3.13}$$

$$\sigma^2(x^*) = \vec{k}_{**} - \vec{k}_{*n}\vec{K}_{nn}^{-1}\vec{k}_{n*}, \tag{3.14}$$

Next, we provide a discussion about kernel tricks and their various applications.

3.2 A Brief Discussion in Kernel Classification

The main reason to use kernels in learning systems is to improve the computational power of linear machines and extend linear hypotheses to nonlinear ones. The main property of interest for kernels is that for all $\vec{x}$ and $\vec{z}$ in the input space $X \subseteq \mathbb{R}^d$, we have $\kappa(\vec{x}, \vec{z}) =$

$\vec{\phi}(\vec{x})^T\vec{\phi}(\vec{z})$, where $\vec{\phi}(\cdot)$ is a nonlinear map from the input space X to the feature space Φ.

Typically, there are three classes of kernels: stationary, non-stationary, and locally stationary kernels; see [9].

Stationary kernels imply that the correlation between data points depends only on their relative distances, and not on their absolute positions. The Radial Basis Function (RBF) and Matérn kernels are examples of stationary kernels.

In contrast, **non-stationary kernels** depend on the individual values of $\vec{x}$ and $\vec{z}$, not just on their relative distance. An example of a non-stationary kernel is the polynomial kernel [20].

The **Matérn kernel** is defined as follows:

$$\kappa(\vec{x},\vec{z}) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\|\vec{x}-\vec{z}\|}{\rho}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}\|\vec{x}-\vec{z}\|}{\rho}\right), \tag{3.15}$$

where $\nu, \rho > 0$ are tuning parameters, K_ν is the modified Bessel function of the second kind, and $\Gamma(\nu)$ is the Gamma function [3]. When $\nu = \frac{1}{2}$, the Matérn kernel reduces to the RBF kernel.

Smaller values of ν are suitable for modeling non-smooth or less smooth functions, while larger values—especially $\nu \geq 2$—are suitable for smooth functions [14].

The general form of the **polynomial kernel** is:

$$\kappa(\vec{x},\vec{z}) = (\vec{x}^T\vec{z} + c)^d, \tag{3.16}$$

where c is a constant and d is the degree of the kernel. Both are hyperparameters that control the complexity of the model. Common choices are $d \leq 5$ and $c = 1$ [26].

Locally stationary kernels are not discussed in detail here, but they depend on the average locations of the input data [20].

4 Gaussian process with global approximation

As it is mentioned in Section (1), The GP in classical implementation applied for n given data has complexity $O(n^3)$ with memory storage requirement of $O(n^2)$ [12]. To enhance the performance of GP, several inferring matrix methods have been proposed to be combined with GP, and enhance its performance. Next we describe, hybrid methods of Nyström approximation and SVD with GPs.

4.1 Nyström Approximation

One common strategy to reduce the number of evaluated data points when constructing the kernel matrix is to remove uncorrelated entries, leading to an approximate kernel matrix $\overline{\vec{K}}_{nn}$ with many zero entries [10, 24].

Another efficient strategy is to choose a subset of the data, called **induced points**, to approximate the kernel matrix in dimension $m \ll n$, where m is the number of induced points and n is the number of training points [4]. The Nyström approximation is expressed as:

$$\vec{K}_{nn} \approx \vec{K}_{nm}\vec{K}_{mm}^{-1}\vec{K}_{mn}, \tag{4.1}$$

where $\vec{K}_{nm}$ and $\vec{K}_{mn}$ are kernel matrices between the induced and training data points, and $\vec{K}_{mm}$ is the kernel matrix among the induced points.

Here, the m pairs of induced data are denoted by $(\vec{X}_m, \vec{f}_m)$, where the latent variables $\vec{f}_m$ depend on the full latent function vector $\vec{f}$. According to the GP prior, we have:

$$p(\vec{f}_m) = \mathcal{N}(\vec{0}, \vec{K}_{mm}),$$

and the joint distribution of $\vec{f}$ and $\vec{f}_*$ is:

$$p(\vec{f}, \vec{f}_*) = \int p(\vec{f}, \vec{f}_*|\vec{f}_m)\, p(\vec{f}_m)\, d\vec{f}_m \tag{4.2}$$

$$= \int p(\vec{f}|\vec{f}_m)\, p(\vec{f}_*|\vec{f}_m)\, p(\vec{f}_m)\, d\vec{f}_m. \tag{4.3}$$

Relations (4.2) and (4.3) are equivalent when $\vec{f}$ and $\vec{f}_*$ are conditionally independent given $\vec{f}_m$. The conditional distributions are:

$$p(\vec{f}|\vec{f}_m) = \mathcal{N}(\vec{f} \mid \vec{K}_{nm}\vec{K}_{mm}^{-1}\vec{f}_m, \vec{K}_{nn} - \vec{K}_{nm}\vec{K}_{mm}^{-1}\vec{K}_{mn}), \tag{4.4}$$

$$p(\vec{f}_*|\vec{f}_m) = \mathcal{N}(\vec{f}_* \mid \vec{k}_{*m}\vec{K}_{mm}^{-1}\vec{f}_m, k_{**} - \vec{k}_{*m}\vec{K}_{mm}^{-1}\vec{k}_{m*}). \tag{4.5}$$

In (4.4), the induced points are used to approximate the full latent vector $\vec{f}$. Notably, only $m \ll n$ points are needed to infer the predictive mean $\mu(\vec{x}^*)$ and variance $\sigma^2(\vec{x}^*)$.

Comparing the following predictive equations:

$$\mu(\vec{x}^*) = \vec{k}_{*m}\vec{K}_{mm}^{-1}\vec{f}_m, \tag{4.6}$$

$$\sigma^2(\vec{x}^*) = k_{**} - \vec{k}_{*m}\vec{K}_{mm}^{-1}\vec{k}_{m*}, \tag{4.7}$$

to their full-GP counterparts (3.13) and (3.14), highlights the approximation role played by the induced points.

4.2 Principal Component Analysis

Another approach is to consider the eigen-decomposition of the positive semi-definite kernel matrix $\vec{K}_{nn} = \vec{U}_{nn}\vec{\Sigma}_{nn}\vec{U}_{nn}^{\top}$, where $\vec{U}_{nn}$ is the matrix of eigenvectors and $\vec{\Sigma}_{nn}$ is the diagonal matrix of eigenvalues. The approximate inference of $\vec{K}_{nn}$ can then be given by:

$$\vec{K}_{nn} \approx \vec{U}_{nm}\vec{\Sigma}_{mm}\vec{U}_{mn}^{\top}, \tag{4.8}$$

where m corresponds to the number of the largest m eigenvalues of the kernel matrix $\vec{K}_{nn}$ retained for the approximation.

If the approximate kernel matrix in (4.8) does not retain the positive semi-definiteness property, an augmented kernel matrix $\vec{K}_{nn}^{\epsilon} = \vec{K}_{nn} + \sigma_{\epsilon}^{2}\vec{I}_{nn}$ can be used. By applying the Sherman-Morrison-Woodbury formula [17], the inverse of $\vec{K}_{nn}^{\epsilon}$ can be approximated as:

$$(\vec{K}_{nn}^{\epsilon})^{-1} \approx \sigma_{\epsilon}^{-2}\vec{I}_{nn} + \sigma_{\epsilon}^{-2}\vec{U}_{nm}\left(\sigma_{\epsilon}^{2}\vec{\Sigma}_{mm}^{-1} + \vec{U}_{mn}^{\top}\vec{U}_{nm}\right)^{-1}\vec{U}_{mn}^{\top}. \tag{4.9}$$

In relation (4.9), the key idea is that the inverse of the full kernel matrix $\vec{K}_{nn}^{\epsilon}$ can be efficiently computed using only $\vec{K}_{mm}$—the kernel corresponding to m selected components—and the low-rank decomposition components $\vec{U}_{nm}$ and $\vec{U}_{mn}^{\top}$ obtained from the singular value decomposition (SVD).

5 Numerical Examples

In this section, we have provided some numerical examples to assess the validity of the proposed GP solution methodology. Specially, we employed SVD and *Nyström* approximation technique to reduce both the running time and required memory of the implemented GP. The SVD and Nyström approximation focus on finding a suitable approximation of kernel inverse by using the induced points. We choose the induced points randomly from the allowable domain of dataset. All codes have been implemented in *Python* using $Scikit-learn$ library, and is available at the *Github* repository [22].

Example 5.1. To illustrate the application of the equation (28), we implemented a python example, by utilizing a two dimension input random generated data. The code generates dataset and report the approximation of the inverse of kernel matrix K_{nn}^{ϵ}. Dataset consists of 1000 random samples taken from the interval $[-200, 200]\times[-200, 200]$. The approximation have been performed for six different values of $m = 10, 20, 30, 40, 50, 60$ and, five values of $\sigma_{\epsilon} = 0.2, 0.5, 1, 1.5, 2$. The results are represented in Table (1), show the accuracy of the approximation of the inverse of kernel matrix K_{nn}^{ϵ} for different values of m and σ_{ϵ}. The obtained results in Table (1) reveal, when the number of m increases, the loss function value

m	σ_ε	$\|K_{nn}^{-1} - K_{nn}^{\epsilon\,-1}\|_2$	m	σ_ε	$\|K_{nn}^{-1} - K_{nn}^{\epsilon\,-1}\|_2$	m	σ_ε	$\|K_{nn}^{-1} - K_{nn}^{\epsilon\,-1}\|_2$
10	0.2	50.003	20	0.2	50.004	30	0.2	50.008
10	0.5	8.003	20	0.5	8.004	30	0.5	8.008
10	1.0	2.033	20	1.0	2.049	30	1.0	2.074
10	1.5	1.509	20	1.5	1.512	30	1.5	1.516
10	2.0	1.430	20	2.0	1.431	30	2.0	1.432
40	0.2	50.010	50	0.2	50.012	60	0.2	50.141
40	0.5	8.010	50	0.5	8.012	60	0.5	8.145
40	1.0	2.086	50	1.0	2.107	60	1.0	2.166
40	1.5	1.518	50	1.5	1.522	60	1.5	1.524
40	2.0	1.432	50	2.0	1.434	60	2.0	1.434

Table 1: The obtained results by implementing the SVD on a 1000×1000 matrix

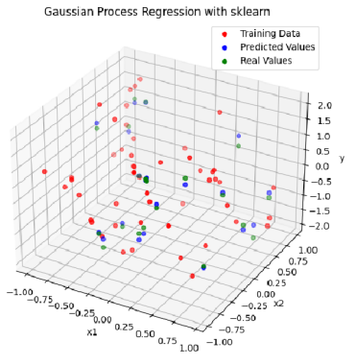
increases. In addition to, when m is selected, by increasing the value of σ_ϵ, loss function error decreases. That is, σ_ϵ decreases the loss function value, since K_{nn}^{ϵ} tends to a dominate diagonal matrices, which is semi-positive or positive definite matrix.

Example 5.2. A GP illustration example, for exploring and predicting the necessity requested region

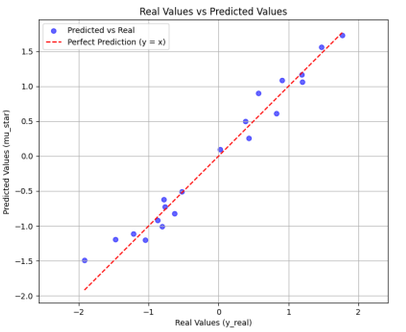
To illustrate the application of the GP method we implemented a Python example utilizing the $scikit-learn$ library. The code generates training and test data sets, fits a GP model, and the validity of GP is verified by comparing the predicted results to the true values. The full implementation can be found on the $GitHub$ repository [22]. The training data set consists of 50 random samples taken from the interval $[-1, 1]\times[-1, 1]$, and the corresponding outputs are generated, using the function $y(x_1, x_2) = \sin(2\pi x_1) + \cos(2\pi x_2)$ with added Gaussian noise. For testing, 20 new samples were used to assess the model's predictions. The kernel employed is a combination of the Radial Basis Function (RBF) and a white noise kernel. The main steps of the implementation include:

1. **Data Generation**: First, a two dimensional training and test data sets are generated, by using the nonlinear function $y(x_1, x_2) = \sin(2\pi x_1) + \cos(2\pi x_2)$.
2. **Model Fitting**: The GP model is taught by utilizing training points, and their corresponding objective values.
3. **Prediction and Evaluation**: The mean and variance of the trained GP model is returned as the output to evaluate the test data where the values of loss function are reported.
4. **Visualization**: The obtained results are depicted in both two and three dimensional spaces to illustrate the validity of the proposed GP model and its implementation.

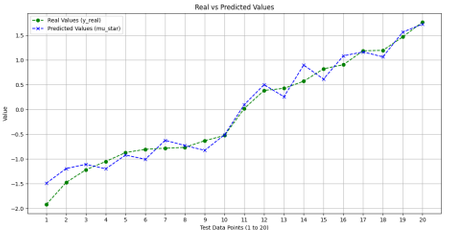
The average absolute and squared errors were computed, and the model's predictions have been illustrated in Figure (1). These plots highlight the capability of GP regression to model nonlinear relationships effectively. Figure (1a) provides a view of the model's performance in three dimension space where training data, real test values, and their predicted values are compared. Next, in Figure (1b) the real and predicted values are depicted in sorted and scatter formats. Additionally, in Figure (1c), the chart of real values has been compared with the obtained predicted values. Finally, to evaluate the uncertainty of predicted values, the bar charts of variances for test points are illustrated in Figure (1d). All results show the predicted values are in high accuracy, although the training set points consists of 50 and test data points consists of 20 which are not so large value. However, the absolute average error is 0.150, and squared average error is 0.033.



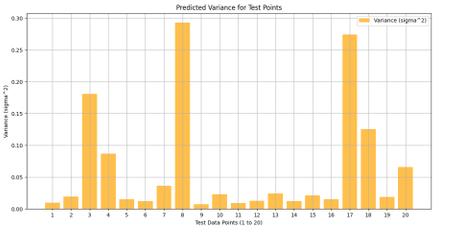
(a) 3D visualization of training data, predicted values, and real test data



(b) Scatter plot of real vs predicted values



(c) Real vs predicted values (sorted by real values)



(d) Variance plot illustrating predictive uncertainty

Figure 1: Visualization of the GP regression performance, including 2 and 3 dimension plots

Example 5.3. Additional GP illustration examples, exploring different nonlinear functions

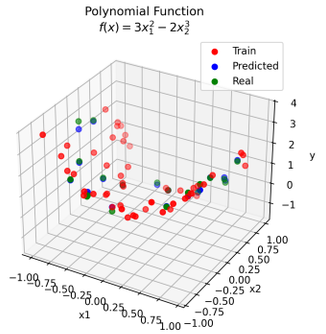
To further evaluate the versatility of the GP model, we tested it on two additional nonlinear functions:

$$\begin{aligned} &\text{Polynomial Function:} && f_1(x_1, x_2) = 3x_1^2 - 2x_2^3, \\ &\text{Exponential–Logarithmic Function:} && f_2(x_1, x_2) = e^{x_1} + \log(|x_2| + 10^{-2}). \end{aligned}$$

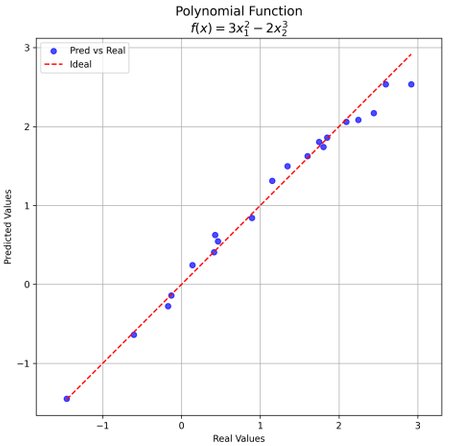
The same training and testing procedure as in the previous example was used:

1. **Data Generation**: For each function, 50 training points were sampled uniformly from $[-1, 1] \times [-1, 1]$ and perturbed with Gaussian noise ($\sigma = 0.1$). 20 test points were sampled independently.
2. **Model Fitting**: A Gaussian Process model was trained using an RBF kernel with an added white noise term.
3. **Prediction and Evaluation**: Mean predictions and predictive variances were obtained for the test points, and the absolute and squared errors were computed.
4. **Visualization**: The training data, predictions, and real test values were visualized in 3D, along with real–predicted comparisons in sorted and scatter formats, and predictive variance plots.

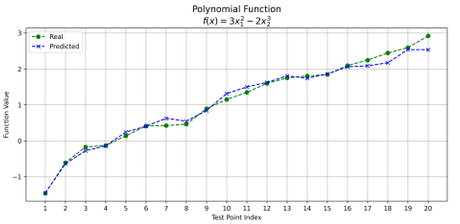
The results are summarized in Figures 2 and 3. For the polynomial function, the GP achieved an absolute average error of 0.134 and squared average error of 0.029. For the exponential–logarithmic function, the errors were 0.148 and 0.035, respectively, showing the GP's robustness across both polynomial and non-polynomial nonlinearities.



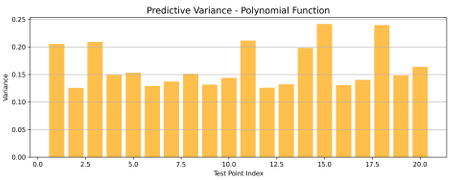
(a) 3D view: Polynomial function



(b) Real vs predicted (polynomial)

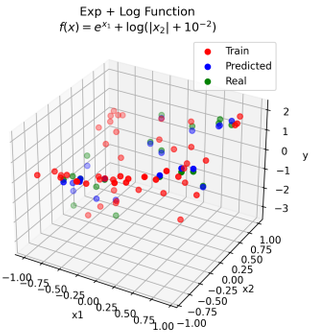


(c) Sorted values comparison (polynomial)

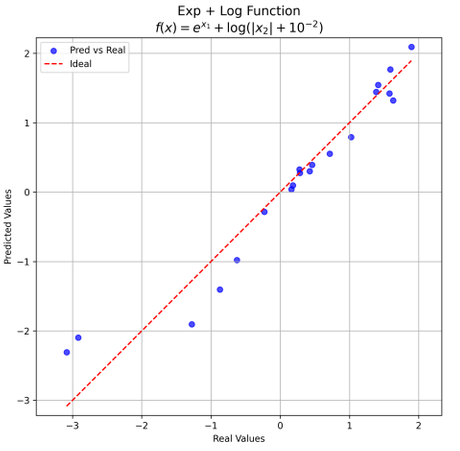


(d) Variance plot (polynomial)

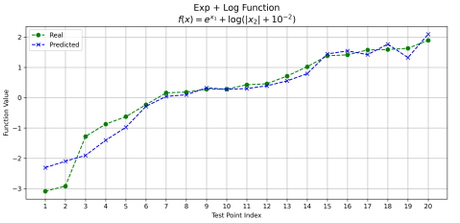
Figure 2: GP performance on the polynomial test function



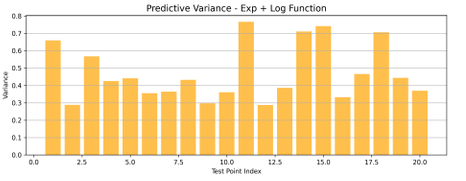
(a) 3D view: Exponential–logarithmic function



(b) Real vs predicted (exp+log)



(c) Sorted values comparison (exp+log)



(d) Variance plot (exp+log)

Figure 3: GP performance on the exponential–logarithmic test function

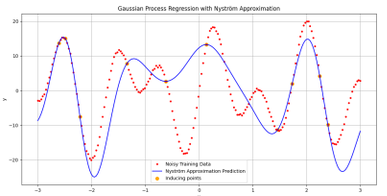
Example 5.4. Nyström Approximation for reducing time complexity

To further illustrate the application of the Nyström approximation method described in this paper, we implemented a *Python* code utilizing synthetic data. The approximation was performed for four different induced points, with sizes $m = 10, 15, 20, 25$, where the number of training data is $n = 200$. The noisy data was generated from the function $y = 10\sin(4x) + 10\sin(7x) + \epsilon$, where ϵ is Gaussian noise with mean zero and standard deviation 0.2. The primary steps of the implementation are as follows:

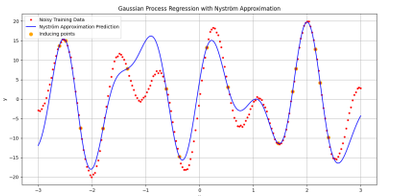
1. **Data Generation**: Here, we generate training data points with function $y(x) = 10\sin(4x) + 10\sin(7x) + \epsilon$ where $\{x_i\}_{i=1}^{n}$ are taken randomly from the interval [-3,3] and $n = 200$.
2. **Nyström Approximation**: We apply the approximation with four different subsets of induced points with cardinalities $m = 10, 15, 20, 25$.

3. **Visualization**: We plot the results to compare predictions and real data, by using the inducing points to constitute the kernel matrices.

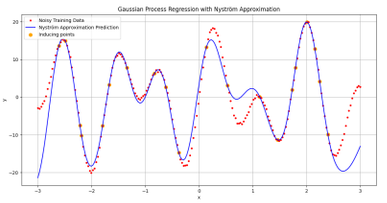
The results are displayed in Figure (4), and illustrate the Nyström approximation for different values of induced points, including $m = 10, 15, 20, 25$. Each subplot includes the noisy training data, predicted values, and inducing points. As the value of m (number of induced points) increases, also the accuracy of predicted points increases, see figures (4a), (4b), (4c) and (4d).



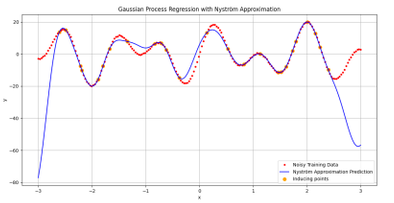
(a) Nyström approximation with $m = 10$



(b) Nyström approximation with $m = 15$



(c) Nyström approximation with $m = 20$



(d) Nyström approximation with $m = 25$

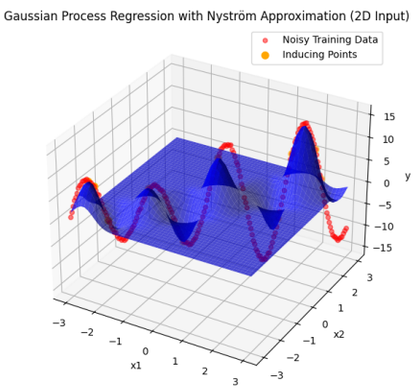
Figure 4: Visualization of Nyström approximation predictions with varying subset sizes m. As m increases, the model's predictions improve in accuracy.

These results highlight the impact of the subset size m on the approximation's performance, and demonstrate the trade-off between computational efficiency and accuracy.

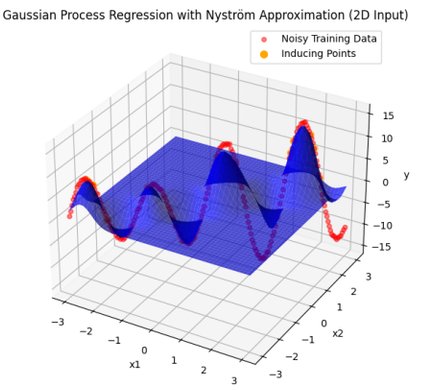
Example 5.5. Nyström approximation for three dimension

To further illustrate the application of Nyström approximation, we implemented a *Python* example, by utilizing a nonlinear two dimension input random generated data. The approximation was performed for four different induced points of sizes $m = 10, 15, 20, 25$, where number of training data is $n = 200$. The noisy data was generated from the function $y = 10\sin(4x_1) + 5\cos(3x_2) + \epsilon$, where ϵ is Gaussian noise with mean 0 and standard deviation 0.2. The primary steps of the implementation are as Example (5.4). The results are

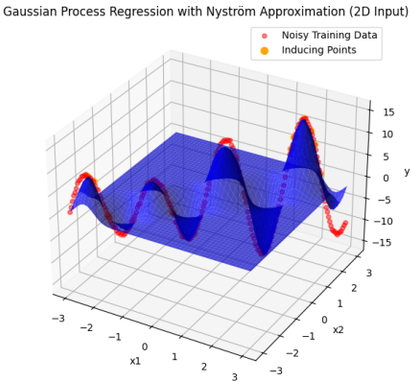
displayed in Figure (5), show the accuracy of Nyström approximation for different number of induced points $m = 10, 15, 20, 25$.



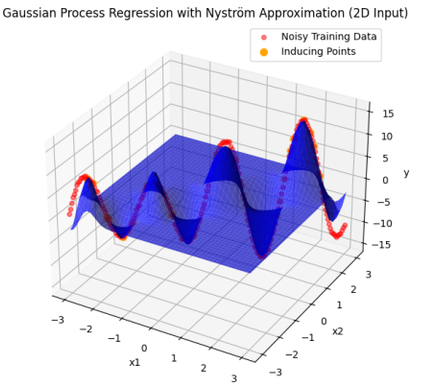
(a) Nyström approximation with $m = 10$



(b) Nyström approximation with $m = 15$



(c) Nyström approximation with $m = 20$



(d) Nyström approximation with $m = 25$

Figure 5: Visualization of Nyström approximation predictions in 3D for varying subset sizes m. As m increases, the model's predictions improve in accuracy.

6 Summary and conclusion

In this paper, the GP has been studied in details. By using the conditional MND, general GP mathematical formulation has been given, incorporating both a surrogate for recognizing latent function and its uncertainty. Then, nonlinear GPs are addressed using the concept of kernel functions. Various kinds of kernel functions are addressed and their

effectiveness on smooth and non-smooth problems are compared. The main shortage and disadvantage of the GP is its limitation on large-scale problems, due to its cubic run time and quadratic memory requirement. To tackle these shortages, we focused on some numerical matrix approximations, among the others Nyström and SVD has been selected, and a hybrid Nyström approximation with GP is given. Also, some numerical examples have been provided to verify the efficiency of GP and hybrid Nyström GP solution approaches. Some corresponding *Python* codes have been written, and embedded in *GitHub* repository [22].

References

[1] H. Azin, A.l. Kashkooly, *Multi-step scheme for the approximation of the fractional Riccati differential equation*, J. frame Matrix Theor., **1** (2024) 60–72. doi

[2] O. Baghani, H. Azin, *A modification to $L_{p,\alpha}$ and its applicability in error estimation of triangular functions*, J. frame Matrix Theor., **1** (2023) 9–14. doi

[3] F. Bowman, *Introduction to Bessel functions*, Dover Publications, Inc., New York, 1958. zbl MR

[4] K. Chalupka, C.K.l. Williams, and I. Murray, *A framework for evaluating approximation methods for gaussian process regression*, J. Mach. Learn. Res., **14** (2013) 333–350. zbl MR arXiv

[5] H. Chen, L. Zheng, R. Al Kontar, and G. Raskutti, *Stochastic gradient descent in correlated settings: A study on gaussian processes*, Adv. Neural Inf. Process Syst., Curran Associates, Inc., **33** (2020) 2722–2733. link pdf

[6] P.I. Frazier, *A tutorial on bayesian optimization*, arXiv preprint, arXiv:1807.02811, 2018. arXiv

[7] M.C. Fu, *Stochastic gradient estimation*, Handbook of Simulation Optimization, Springer, New York, 2015. doi

[8] T. Galy-Fajou, M. Opper, *Adaptive inducing points selection for gaussian processes*, arXiv preprint, arXiv:2107.10066, 2021. arXiv

[9] M.G. Genton, *Classes of kernels for machine learning: A statistics perspective*, J. Mach. Learn. Res., **2** (2002) 299–312. zbl MR doi

[10] S. Gilpin, T. Matsuo, and S.E. Cohn, *A generalized, compactly supported correlation function for data assimilation applications*, Q. J. R. Meteorol Soc., **149** (2023) 1953–1989. pdf

[11] P.E. Hart, D.G. Stork, and R.O. Duda, *Pattern classification*, Wiley Interscience, 2nd

ed., New York, 2001. zbl MR

[12] J. Hensman, N. Fusi, and N.D. Lawrence, *Gaussian processes for big data*, arXiv preprint, arXiv:1309.6835, 2013. arXiv

[13] B. Jones, R.T. Johnson, *Design and analysis for the gaussian process model*, Qual. Reliab. Engng. Int., **25** (2009) 515–524. link

[14] M. Kanagawa, P. Hennig, D. Sejdinovic, and B.K. Sriperumbudur, *Gaussian processes and kernel methods: A review on connections and equivalences*, arXiv preprint, arXiv:1807.02582, 2018. arXiv

[15] J.A. Lin, J. Antorán, S. Padhy, D. Janz, J.M. Hernández-Lobato, and A. Terenin, *Sampling from gaussian process posteriors using stochastic gradient descent*, Adv. Neural Inf. Process Syst., Curran Associates, Inc., **36** (2023) 36886–36912. pdf

[16] H. Liu, Y.S. Ong, X. Shen, and J. Cai, *When gaussian process meets big data: A review of scalable gps*, IEEE Trans. Neural Netw. Learn. Syst., **31** (2020) 4405–4423. MR doi

[17] P. Maponi, *The solution of linear systems by using the sherman–morrison formula*, Linear Algebra Appl., **420** (2007) 276–294. zbl MR doi

[18] A. Mustapha, L. Mohamed, and K. Ali, *An overview of gradient descent algorithm optimization in machine learning: Application in the ophthalmology field*, Springer, Cham., p.p.: 349–359, 2020. doi

[19] D. Newton, F. Yousefian, and R. Pasupathy, *Stochastic gradient descent: Recent trends*, TutORials in Operations Research, (2018) 193–220. doi

[20] M.M. Noack, J.A. Sethian, *Advanced stationary and nonstationary kernel designs for domain-aware gaussian processes*, Commun. Appl. Math. Comput. Sci., **17** (2022) 131–156. zbl MR doi

[21] E. Schulz, M. Speekenbrink, A. Krause, *A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions*, J. Math. Psychol., **85** (2018) 1–16. zbl MR doi

[22] A. Shafiefard, *Gaussian process repository*, (2024).

[23] P. Siminelakis, K. Rong, P. Bailis, M. Charikar, and P. Levis,*Rehashing kernel evaluation in high dimensions*, In Proceedings of the 36th International Conference on Machine Learning, vol. 97 of Proceedings of Machine Learning Research (PMLR), pp. 5789-5798, 2019. pdf

[24] A. Smola, P. Bartlett, *Sparse greedy gaussian process regression*, Adv. Neural Inf. Pro-

cess Syst., MIT Press, **13**, 2000. pdf

[25] J. Snoek, H. Larochelle, and R.P. Adams, *Practical bayesian optimization of machine learning algorithms*, Adv. Neural Inf. Process Syst., Curran Associates, Inc., **25** 2012. pdf

[26] C.J. Walder, B.C. Lovell, *Kernel based algebraic curve fitting*, In Proceedings of the International Conference on Advances in Pattern Recognition, Allied Publishers, Kolkata, (2003) 387–390. link

[27] F. Wang, J.E. Chen, *Efficient modeling of random fields by using gaussian process inducing-point approximations*, Comput. Geotech., **157** (2023) 105304. doi

[28] J. Wenger, G. Pleiss, P. Hennig, J. Cunningham, and J. Gardner, *Preconditioning for scalable gaussian process hyperparameter optimization*, In Proceedings of the 39th International Conference on Machine Learning, vol. 162 of Proceedings of Machine Learning Research (PMLR), pp. 23751–23780, 2022.
PMLR, (2022) 23751–23780. pdf

[29] V. Wild, M. Kanagawa, and D. Sejdinovic, *Connections and equivalences between the nyström method and sparse variational gaussian processes*, arXiv preprint, arXiv:2106.01121, 2023. arXiv

[30] C. Williams, C. Rasmussen, *Gaussian processes for regression*, Adv. Neural Inf. Process Syst., MIT Press, **8** 1995. pdf