Reduced run-time and memory complexity regression with a Gaussian processes prior

Amal Omrani [1] & Anis Fradi [2] & Chafik Samir [3]

[1] *Paris Dauphine University, France, amal.omrani@dauphine.psl.eu*
[2] *University of Clermont Auvergne, France, anis.fradi@uca.fr*
[3] *University of Clermont Auvergne, France, chafik.samir@uca.fr*

Résumé. Les processus gaussiens ont connu un grand succès d'utilisation et de performance dans plusieurs applications d'apprentissage automatique. Entre autres, ils possèdent des propriétés théoriques et pratiques qui en font une solution flexible et adaptable pour des problèmes de régression et de classification. Cependant, ils présentent des inconvénients limitants en raison de la complexité computationnelle et en mémoire. Dans ce papier, nous proposons une nouvelle méthode pour construire des processus gaussiens efficaces sur le plan computationnel dans le cadre d'une régression ou d'une classification. En particulier, nous démontrons que les modèles proposés ont une complexité en temps et en mémoire inférieure à celle des processus gaussiens standards. Nous confirmons cette meilleure performance, en pratique, grâce à des expériences et des comparaisons variées.

Mots-clés. processus gaussiens, régression, classification, regroupement, complexité.

Abstract. Gaussian processes have been successfully used in machine learning applications. They have nice theoretical and practical properties that make them a flexible solution for regression and classification problems. However, they have serious limitations due to the memory and computational complexity. In this paper, we propose a new method to build computationally efficient Gaussian processes for regression and classification problems. The proposed models are shown to have lower time and memory complexity than standard Gaussian process models. Furthermore, we confirm this better performance in practice with varied experiences and comparisons.

Keywords. Gaussian processes, regression, classification, clustering , complexity.

1 Introduction

Gaussian processes (GPs) are powerful probabilistic models used in machine learning and statistics [5]. They define a distribution over functions, where any finite subset of function values follows a multivariate Gaussian distribution. A GP is characterized by a mean function and a covariance function that encapsulates the smoothness and correlations in the function space. This flexibility allows GPs to model a wide range of complex and non-linear relationships, making them particularly valuable in regression, classification, and uncertainty quantification tasks. GPs are flexible statistical models that have gained significant popularity in the fields of econometrics, shape analysis, signal processing, data science and machine

learning [3]. Their ability to provide uncertainty estimates alongside predictions makes GPs essential in decision-making processes where understanding the confidence in the model's output is crucial.

GPs have some limitations, especially when dealing with high number of observations. The two main challenges are complexity and memory usage:

- **Computational Complexity**: The computational complexity of GPs is cubic with respect to the number of data points N, making them impractical for large datasets. Inference and learning involve inverting a covariance matrix, which requires $\mathcal{O}(N^3)$ operations. This makes GPs slow and resource-intensive as the dataset size increases.
- **Memory Consumption**: GPs store and manipulate a covariance matrix that grows rapidly with the number of data points N, requiring a significant amount of memory. The covariance matrix is of size $\mathcal{O}(N^2)$, leading to high memory consumption for large datasets. This can be a significant bottleneck in memory-limited environments.

Efforts to address these limitations include approximate inference methods like variational approaches and sparse approximations [1]. These methods attempt to reduce the computational and memory complexity of GPs while still providing reasonable performance. In addition, [6] addressed the computational challenges by employing an approximation method for the covariance matrix, a crucial component in computations involving kernel methods. Moreover, [4] introduced an innovative approach using the Fast Fourier Transform (FFT) for stationary covariances. This method efficiently computes and manipulates covariance functions in the frequency domain.

In pursuit of the same objective, we introduce two clustering-based approaches for selecting the most informative observations, focusing on managing large datasets. Within each cluster, we choose representative or centroid points to serve as observations for the GP. This strategy effectively alleviates the computational and memory demands associated with handling the entire dataset, rendering the GP more scalable [2]. By carefully selecting representative points from the clusters, we uphold a strong approximation of the underlying data structure and variability. This enables us to create effective models and accurate predictions while significantly enhancing computational efficiency. This technique strikes a critical balance between model performance and computational feasibility, making GPs a more viable option for larger datasets.

Organization. The remainder of this paper is structured as follows. In Section 2, we introduce the standard Gaussian process regression model. Section 3 outlines our proposed method. We provide an illustrative example in Section 4, and conclude the paper in Section 5.

2 Standard Gaussian process regression

Gaussian process regression (GPR) main goal is to model and predict the relationship between input data points $\mathbf{x}_i \in \mathbb{R}^d$ and their corresponding output values $y_i \in \mathbb{R}$ from the nonlinear

relationship

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \quad \text{with} \quad \epsilon_i \sim \mathcal{N}(0, \sigma_N^2) \tag{1}$$

where $f(\mathbf{x})$ is assumed to be a realization of a GP f. The stochastic process f is characterized by a mean function $m(\mathbf{x})$ and a covariance function (or kernel) $k(\mathbf{x}, \mathbf{x}')$ denoted as follows

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{2}$$

The mean m is usually assumed to be zero for which the GP is assumed to be a centered GP. The kernel k typically depends on a set of unknown hyperparameters that need to be learned from the data when maximizing the log marginal likelihood which is proportional to

$$L = -\frac{1}{2}\log|\mathbf{K} + \sigma_N^2\mathbf{I}| - \frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma_N^2\mathbf{I})^{-1}\mathbf{y} \tag{3}$$

A commonly used kernel is the squared exponential (SE) defined as: $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\ell^2}\right)$, which depends on a length scale hyperparameter ℓ. The optimization process usually involves optimization techniques such as gradient-based optimization or other optimization algorithms to maximize the log marginal likelihood with respect to the hyperparameters, including the noise variance σ_N^2 and the length scale ℓ for the SE kernel, to obtain the best-fitting hyperparameters for the given dataset.

In GPR predicting the mean μ_* and the variance σ_*^2 for a test point $\mathbf{x}_* \in \mathbb{R}^d$ involves the calculation of their respective mathematical expressions

$$\mu_* = \mathbf{k}_*^T(\mathbf{K} + \sigma_N^2\mathbf{I})^{-1}\mathbf{y} \tag{4}$$

$$\sigma_*^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T(\mathbf{K} + \sigma_N^2\mathbf{I})^{-1}\mathbf{k}_* \tag{5}$$

where $\mathbf{K}$ is the covariance matrix of the training inputs $\mathbf{x}_i$, $\mathbf{k}_*$ is the vector of covariances between the test point $\mathbf{x}_*$ and training inputs $\mathbf{x}_i$, $k(\mathbf{x}_*, \mathbf{x}_*)$ is the variance of the test point $\mathbf{x}_*$, $\mathbf{I}$ is the identity matrix and $\mathbf{y}$ is the vector of outputs y_i.

To address the problem of GPR when dealing with large datasets, we introduce two clustering approaches based on selecting the most important points for GPR.

3 Reduced Gaussian process regression

Consider a large ($N \gg 1$) sample: $\mathcal{D} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}_{i=1}^N$. The canonical GPR is expressed as

$$\begin{aligned} y_i &= f(\mathbf{x}_i) + \epsilon_i, \quad \text{with} \quad i = 1, \ldots, N \\ f(\mathbf{x}) &\sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}')) \end{aligned} \tag{6}$$

Our challenge lies in managing the computational complexity of standard GPR while maintaining a good performance. Hence, we propose the use of two clustering approaches.

3.1 An efficient representation of observations

The objective is to identify a reduced subset of training observations that effectively capture the essential information within the data, facilitating both learning (3) and inference (4) tasks. Denote the number of clusters as k with $k \ll N$. Initially, a clustering algorithm will group similar observations, resulting in clusters represented as $\mathcal{C} = \{C_j\}_{j=1}^k$ where each cluster C_j contains a subset of training data points that are similar within the cluster j. Subsequently, each cluster C_j will be represented by its center $\mathbf{c}_j \in \mathbb{R}^d$. These cluster centers will serve as a reduced set of observations. The reduced GPR model will then be learned on a reduced subset of observations, denoted as $\tilde{\mathcal{D}} = \{(\mathbf{c}_j, y_j)\}_{j=1}^k$, satisfying

$$\begin{aligned} y_j &= f(\mathbf{c}_j) + \epsilon_j, \quad \text{with} \quad j = 1, \ldots, k \\ f(\mathbf{c}) &\sim \mathcal{GP}(0, k(\mathbf{c}, \mathbf{c}')) \end{aligned} \tag{7}$$

k-Means clustering. The goal of the k-Means algorithm is to partition the data into k clusters, such that each data point belongs to the cluster with the nearest centroid. Algorithm 1 iteratively assigns data points to the nearest centroid and updates the centroids until convergence, where convergence is typically defined as no change in the centroid assignments. Since the resulting centroids are usually not among the data $\mathbf{x}_1, \ldots, \mathbf{x}_N$, we use the nearest neighbors (NN) classification method applied to the data in each cluster, and the nearest one will be assigned as the centroid $\mathbf{c}_j$.

Algorithm 1 k-Means clustering.

1: **Input:** Data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and a number of clusters k
2: **Output:** centroids: $\mathbf{c}_1, \ldots, \mathbf{c}_k$
3: Initialize k cluster centroids randomly: $\mathbf{c}_1^{(0)}, \ldots, \mathbf{c}_k^{(0)}$
4: **repeat**
5: For each $\mathbf{x}_i$ find $\text{cluster}(\mathbf{x}_i) = \arg\min_{1 \le l \le k} \|\mathbf{x}_i - \mathbf{c}_l^{(t)}\|^2$
6: Update the centroids by computing the mean of each cluster:

$$\mathbf{c}_l^{(t+1)} = \frac{1}{|\{\mathbf{x}_i \,|\, \text{cluster}(\mathbf{x}_i) = l\}|} \sum_{\{\mathbf{x}_i \,|\, \text{cluster}(\mathbf{x}_i) = l\}} \mathbf{x}_i$$

7: **until** convergence

k-Medoids clustering. The goal of the k-Medoids algorithm is to partition the data points into k clusters, where each cluster is represented by a medoid. The medoid is defined as the data point within the cluster that minimizes the sum of distances to all other points in the same cluster. Algorithm 2 aims to minimize the total sum of distances between data points and their respective medoids.

3.2 The computational challenge

The canonical GPR in (6) exhibits a time complexity of $\mathcal{O}(N^3)$ for both learning and inference due to the need to invert the $N \times N$ covariance matrix $\mathbf{K}$, limiting its scalability for large

Algorithm 2 k-Medoids clustering.

1: **Input:** Data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and a number of clusters k
2: **Output:** Medoids: $\mathbf{c}_1, \ldots, \mathbf{c}_k$
3: Initialize k cluster medoids randomly: $\mathbf{c}_1^{(0)}, \ldots, \mathbf{c}_k^{(0)}$
4: **repeat**
5: For each $\mathbf{x}_i$ find cluster$(\mathbf{x}_i) = \arg\min_{1 \leq l \leq k} ||\mathbf{x}_i - \mathbf{c}_l^{(t)}||^2$
6: Update the medoids among $\mathbf{x}_i$ by selecting the point that minimizes the sum of distances within the cluster the medoid:

$$\mathbf{c}_l^{(t+1)} = \arg\min_{\mathbf{x}_i \in \mathbf{X}} \sum_{\mathbf{x} \in \text{cluster}(\mathbf{x}_i)} ||\mathbf{x} - \mathbf{x}_i||^2$$

7: **until** convergence

datasets when $N \gg 1$. Moreover, another limitation of GPR is the memory scaling $\mathcal{O}(N^2)$ in a direct implementation. In contrast, k-Means and k-Medoids clustering offer more efficient solutions. For k-Means clustering, the complexity is $\mathcal{O}(TkN)$, where T represents the number of iterations. On the other hand, k-Medoids clustering yields a time complexity of $\mathcal{O}(k(N-k)^2)$. By employing k-Means or k-Medoids to select a reduced subset of representatives, the time complexity of the reduced GPR (RGPR) model in (7) can be substantially reduced to $\mathcal{O}(k^3)$ and the memory consumption to $\mathcal{O}(k^2)$ where $k \ll N$.

4 Experimental results

Experimental protocol. We employ a real dataset for binary classification comprising 1500 observations and a total of 200 features, illustrating the body temperature of dogs represented by their probability densities. Within this dataset, temporal measures of infected (501) and uninfected (999) dogs are recorded over a 24-hour period. Figure 1 shows the mean curve of each class with their confidence intervals of a level of 90%. We use 75% (1125 instances) for training and reserve 25% (375 instances) for test. For evaluation we consider three measures: Accuracy, F1-score, and AUC value.

The objective of this experiment is to evaluate the effectiveness of clustering-based methods, specifically k-Means and k-Medoids, in reducing the computational complexity of the standard GP classifier (GPC) while maintaining the predictive performance. For comparison, we deal with the PCA by extracting two principal directions for each observation (PCA GPC), which represent 98% of the initial information. In this investigation, we rely on the `sklearn` library (scikit-learn), which provides implementations of GPC, PCA, k-Means, and k-Medoids clustering algorithms.

Results and discussion. Table 1 shows the details of the training set for each case. In particular, the number of clusters for k-Means and k-Medoids is selected by employing the Elbow method for a fixed value range of [4,80]. The visualization in Figures 3, 2, 5, and 4 illustrates the reduced samples using two features selected in two directions through PCA.

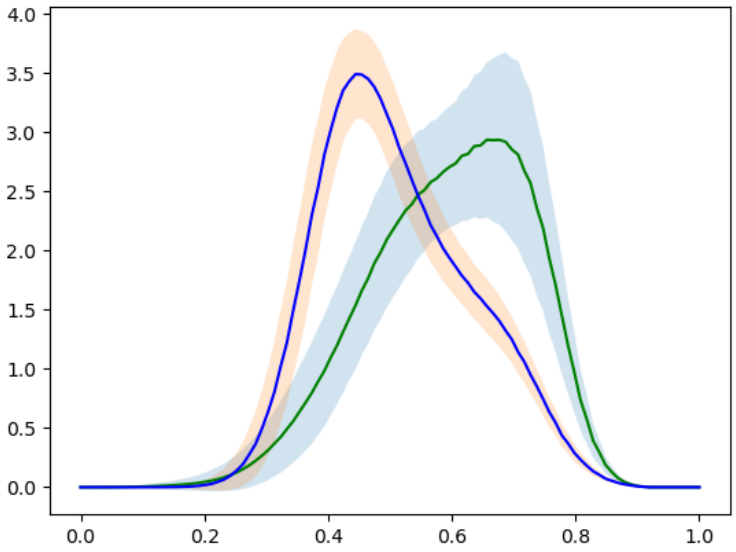


Figure 1: The mean curve of the first class (green) and the second class (blue) with their respective confidence intervals.

Method	**Class 1**	**Class 2**	(N, d)
GPC	365	760	(1125, 200)
PCA GPC	365	760	(1125, 2)
k-Means GPC	59	66	(125, 200)
k-Medoids GPC	65	70	(135, 200)

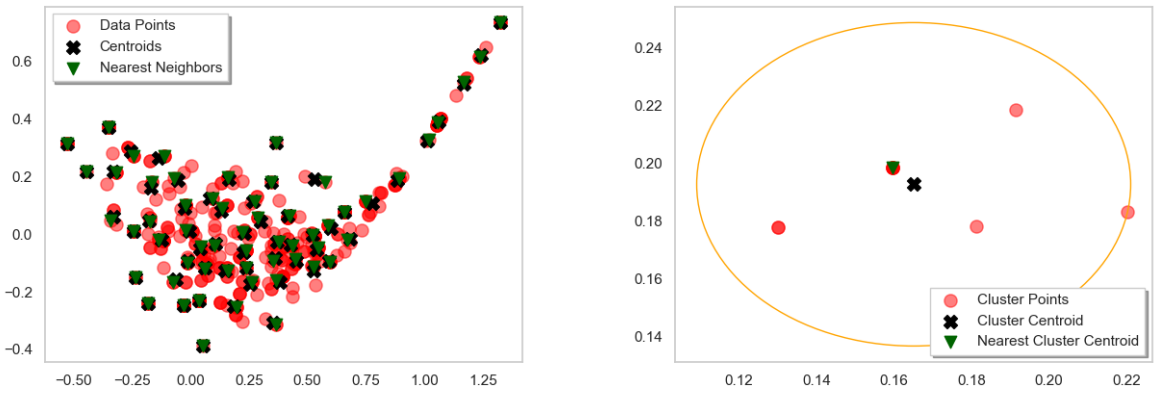
Table 1: Repartition and dimension of training set.

Specifically, it showcases centroids and their nearest points used as the reduced set for training in the case of k-Means, and medoids that will be utilized as the subset for model training in the case of k-Medoids. It is important to note that during the clustering process, we segregate the positive and negative classes, applying clustering to each class independently. Subsequently, we concatenate the reduced subsets of each class to obtain the final reduced training set.

Comparison. In our comparative study, we assess the performance of three models: The standard GPC, PCA GPC, and reduced GPC by both k-Means and k-Medoids. Figure 6 on the left displays the results over 100 random repetition, while Figure 6 on the right presents the mean values of the performance results. The mean execution times are summarized in Table 2.

5 Conclusion

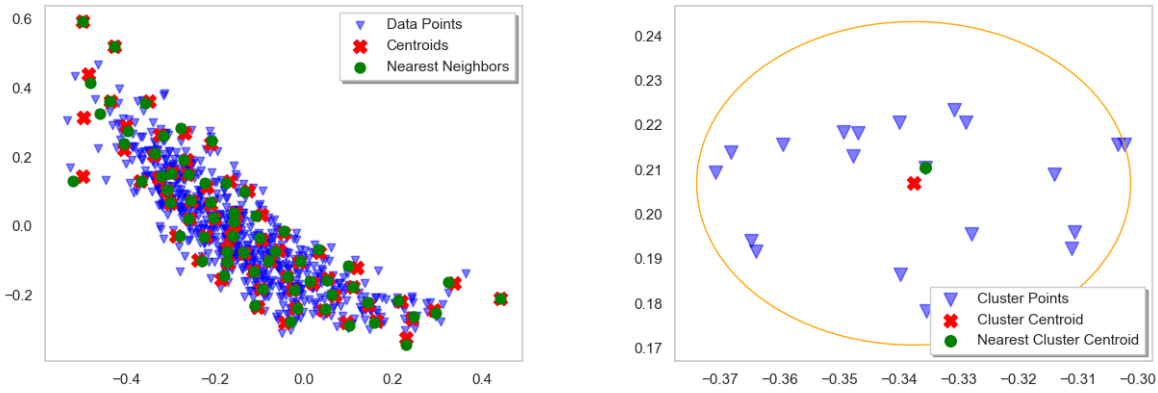
In this paper, we have proposed a new method for reducing the complexity and computation time of standard Gaussian processes for regression and classification through clustering



(a) Centroids of class 1

(b) One cluster of class 1

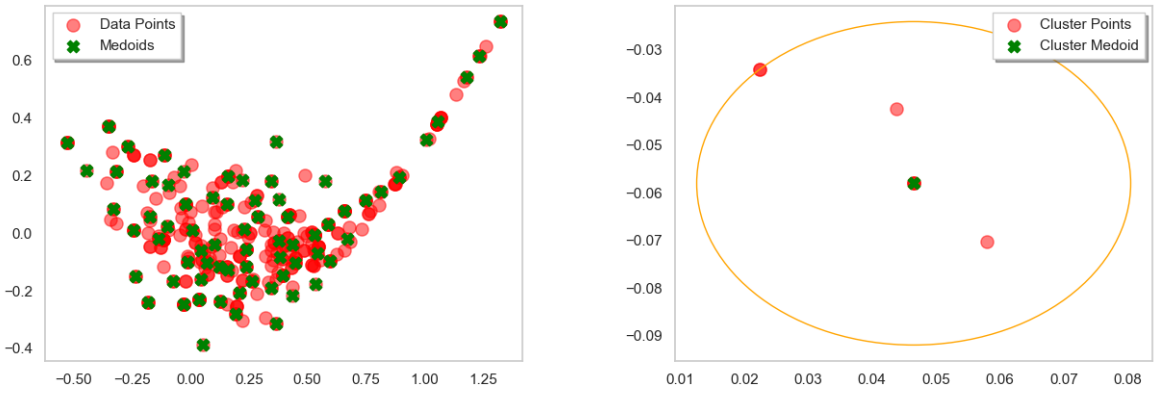
Figure 2: Centroids of class 1 using k-Means.



(a) Centroids of class 2

(b) One cluster of class 2

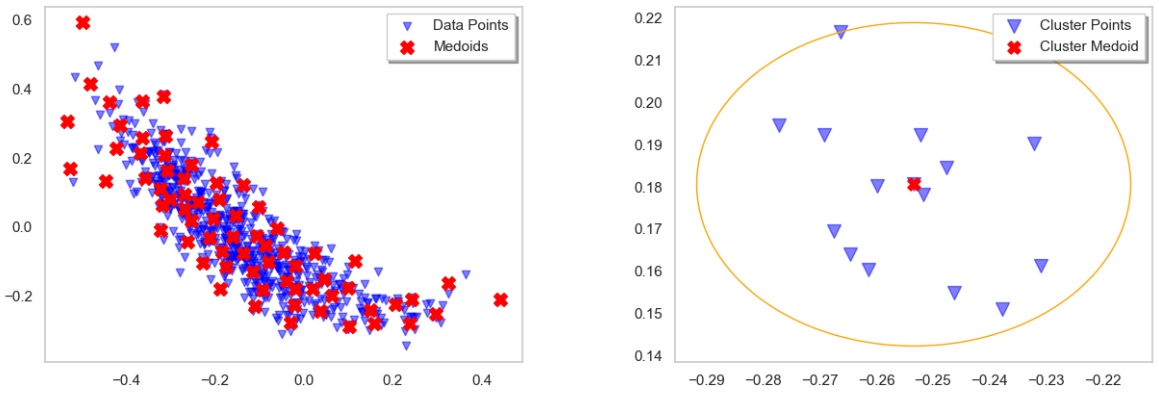
Figure 3: Centroids of class 2 using k-Means.



(a) Medoids of class 1

(b) One cluster of class 1

Figure 4: Medoids of class 1 using k-Medoids.



(a) Medoids of class 2

(b) One cluster of class 2

Figure 5: Medoids of class 2 using k-Medoids.

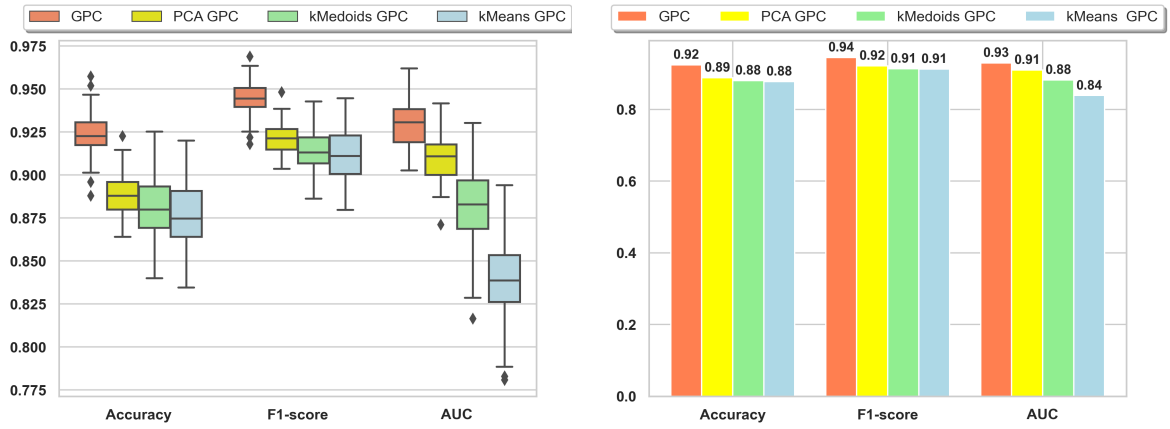


Figure 6: The left figure represents the box plots over 100 repetitions, while the right one illustrates the mean values for each metric.

Method	**Mean Execution Time (seconds)**
GPC	21.55
PCA GPC	12.48
k-Means GPC	1.17
k-Medoids GPC	0.27

Table 2: Mean of execution times for each method across 100 repetitions.

techniques. The experimental study confirmed that the proposed reduced Gaussian process has very promising advantages over the standard model in terms of complexity.

This link provides a code preview: Reduced-Run-Time-and-Memory-Complexity-of-GP.

References

[1] A. Damianou, M. Titsias, and N. Lawrence. Variational Gaussian process dynamical systems. In *Advances in Neural Information Processing Systems*, pages 2510–2518. Curran Associates, Inc., 2011.

[2] A. Fradi, C. Samir, and F. Bachoc. A scalable approximate Bayesian inference for high-dimensional Gaussian processes. *Communications in Statistics - Theory and Methods*, pages 1–68, 2020.

[3] A. Fradi, C. Samir, J. Braga, S. H. Joshi, and J-M. Loubes. Nonparametric Bayesian regression and classification on manifolds, with applications to 3D cochlear shapes. *IEEE Transactions on Image Processing*, 31:2598–2607, 2022.

[4] J. Fritz, W. Nowak, and I. Neuweiler. Application of FFT-based algorithms for large-scale universal kriging problems. *Mathematical Geosciences*, 51:199–221, 2009.

[5] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[6] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, volume 13, pages 585–591. MIT Press, 2000.