# Gaussian Process and Beyond

**Sailun Xu**
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
sailunx@andrew.cmu.edu

**Kaiji Lu**
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
kaijil@andrew.cmu.edu

## 1 Introduction

In this project, we will be mainly focusing on Gaussian Process for the regression problem. A Gaussian Process is a collection of random variables where any finite number of which have a jointly Gaussain Distribution,

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

where $x \in \mathbb{R}^p$ is an arbitrary dimensional input variable. The mean function $m(x) = \mathbb{E}[f(x)]$ and covariance kernel $k(x, x') = cov(f(x), f(x'))$ characterize the prior mean and wiggliness of function value.

Then for a concrete sample: $\{x_i\}_{i=1}^N$

$$[f(x_1), f(x_2), \cdots, f(x_n)]^T \sim \mathcal{N}(\mu, K)$$

where the $N \times N$ covariance matrix $K : K_{ij} = k(x_i, x_j)$, and mean vector $\mu_i = m(x_i)$

Consider the regression model

$$y_i = f(x_i) + \epsilon_i, \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Suppose that we observe data $X, Y = \{x_i, y_i\}_{i=1}^n$, and we are given a new test point $x_{n+1}$
Denote:

$$K : K_{ij} = k(x_i, x_j), 1 \leq i, j \leq n$$
$$k : k_i = k(x_i, x_{n+1}), 1 \leq i \leq n$$
$$\mu : \mu_i = m(x_i)$$

According to [1], the joint distribution could be written as:

$$y_1, \cdots, y_n, y_{n+1} \sim \mathcal{N}\left(\begin{bmatrix} \mu \\ m(x_{n+1}) \end{bmatrix}, \begin{bmatrix} K + \sigma^2 I & k \\ k^T & k(x_{n+1}, x_{n+1}) + \sigma^2 \end{bmatrix}\right) \tag{1}$$

We recognize the predictive distribution as a conditional Gaussian Distribution:

$$y_{n+1} \sim \mathcal{N}(k^T (K + \sigma^2 I)^{-1} Y, k(x_{n+1}, x_{n+1}) + \sigma^2 - k^T (K + \sigma^2 I)^{-1} k) \tag{2}$$

We could also analytically calculate the marginal likelihood of the data, therefore leading to the MLE estimator $\hat{\theta}_{MLE}$:

$$\hat{\theta}_{MLE} = \arg\max_\theta \log p(y|\theta) = \arg\max_\theta -y^T (K_\theta + \sigma^2 I) y + \log \left| K_\theta + \sigma^2 I \right|$$

Preprint. Work in progress.

## 2 Scalable Structured Gaussian Process

### 2.1 Fast Structure Exploitation

#### 2.1.1 Kronecker Methods

According to [2], suppose that we have $N$ multidimensional inputs on a Cartesian grid, $x \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_P$, $n_p$ per grid dimensino $\mathcal{X}_p$, where $N = \prod_{p=1}^{P} n_p$. For simplicity, throughout this report we assume that $n_p = N^{\frac{1}{P}}$ We have also a product kernel $k(x_i, x_j) = \prod_{p=1}^{P} k(x_i^{(p)}, x_j^{(p)})$, then the covariance matrix $K = K_1 \otimes \cdots \otimes K_p$, a kronecker product of the covariance matrices per grid dimension. Then we could write down the eigen decomposition of $K$ as:

$$K = QVQ^T = Q_1V_1Q_1^T \otimes \cdots \otimes Q_PV_PQ_P^T, \text{ where } K_p = Q_pV_pQ_p^T, 1 \leq p \leq P$$

By the mixed product property, we could also write $K$ as([3]):

$$K = (Q_1 \otimes \cdots \otimes Q_P)(V_1 \otimes \cdots \otimes V_P)(Q_1 \otimes \cdots \otimes Q_P)^T$$

Therefore the space complexity is reduced from $O(N^2)$ to $O(PN^{\frac{2}{P}})$, and the eigen decomposition time complexity is reduced from $O(N^3)$ to $O(PN^{\frac{3}{P}})$

Once we have the eigendecomposition of $K$ at hand, then the inference becomes trivial in that:

$$(K + \sigma^2 I)^{-1}y = (QVQ^T + \sigma^2 I)^{-1}y = Q(V + \sigma^2 I)^{-1}Q^T y$$

Denote the operation of stacking a matrix $X \in \mathbb{R}^{m,n}$ columns together and forming a vector by $vec(X) \in \mathbb{R}^{mn}$, that of reshaping a matrix $X$ column-wisely by $reshape(X, p, q)$, where $pq = mn$

Notice that by the Kronecker Matrix Vector product property:

$$(\boldsymbol{A} \otimes \boldsymbol{B})vec(\boldsymbol{X}) = vec(\boldsymbol{CXB^T})$$

and repetitively applying this, for a given $Q \in \mathbb{R}^{N \times N} = Q_1 \otimes \cdots \otimes Q_P, y \in \mathbb{R}^N$ would save us from actually computing the Kronecker Matrix:

$$\left(\bigotimes_{p=1}^{P} Q_p\right) y = \left[Q_1, \cdots \left[Q_{P-1}, [Q_p, y]\right]\right] \tag{3}$$

$$= \left[Q_1 \cdots \left[Q_{P-1},\ reshape\left(\left(Q_P\ reshape(y, N^{\frac{1}{P}}, N^{\frac{P-1}{P}})\right)^T, N^{\frac{1}{P}}, N^{\frac{P-1}{P}}\right)\right]\right] \tag{4}$$

since each $[\cdot]$ operator would take us $O(N^{\frac{1}{P}} \cdot N^{\frac{1}{P}} \cdot N^{\frac{P-1}{P}}) = O(N^{\frac{P+1}{P}})$, therefore the total time complexity would be $O(PN^{\frac{P+1}{P}})$

and learning is also trivial in that:

$$\log\left|K + \sigma^2 I\right| = \log\left(|Q|\left|V + \sigma^2 I\right|\left|Q^T\right|\right) = \log\left|V + \sigma^2 I\right| = \sum_i \log\left(V_{ii} + \sigma^2\right)$$

#### 2.1.2 Toeplitz Methods

A Toeplitz Covariance matrix arises when the kernel is a stationary one, i.e.: $k(x, x') = k(|x - x'|)$ and the inputs lie on a equally spaced one dimensional grid, then all entries on the same diagonal will be the same:

$$K = \begin{bmatrix} k_0 & k_1 & k_2 & \cdots & k_{n-1} \\ k_1 & k_0 & k_1 & \cdots & k_{n-2} \\ \cdots & & & & \\ k_{n-1} & k_{n-2} & k_{n-3} & \cdots & k_0 \end{bmatrix}$$

$$k_i = k(x_j, x_{j+i}) \forall i, 1 \leq j \leq n-i; k_i = k(x_j, x_{j-i}) \forall i, i \leq j \leq n$$

it turns out that we can embed this Toeplitz Covariance into a larger circular matrix:

$$\tilde{K} = \left[\begin{array}{ccccc|cccc|c} k_0 & k_1 & k_2 & \cdots & k_{n-1} & k_{n-2} & k_{n-3} & k_{n-4} & \cdots & k_1 \\ k_1 & k_0 & k_1 & \cdots & k_{n-2} & k_{n-1} & k_{n-2} & k_{n-3} & \cdots & k_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ k_{n-1} & k_{n-2} & k_{n-3} & \cdots & k_0 & k_1 & k_2 & k_3 & \cdots & k_{n-2} \\ \hline k_{n-2} & k_{n-1} & k_{n-2} & \cdots & k_1 & k_0 & k_1 & k_2 & \cdots & k_{n-3} \\ k_{n-3} & k_{n-2} & k_{n-1} & \cdots & k_2 & k_1 & k_2 & k_3 & \cdots & k_{n-2} \\ \cdots & & & & & & & & & \\ k_1 & k_2 & k_3 & \cdots & k_{n-2} & k_{n-3} & k_{n-2} & k_{n-1} & \cdots & k_0 \end{array}\right]$$

Since any circulant matrix could be diagonalize as:

$$\tilde{K} = F \Lambda F^{-1}, \Lambda = diag(F\tilde{K}_1)$$

where $F$ is the Discrete Fourier Transform Matrix and $\Lambda$ is a diagonal matrix containing the eigenvalues obtained by multiplying $F$ with the first row of $\tilde{K}$ [4]

By FFT ([5]), we could perform multiplication of $F$ with $y$ in $O(N \log N)$ time. Also, for computing the $F^{-1}y$, we could apply the linear conjugate gradient method ([6]), which only involves multiplication between $F$ and $y$, also giving us a $O(N \log N)$ time complexity. Therefore, the total inference time for Toeplitz Structure is also $O(N \log N)$

### 2.2 Scalable Kernel Interpolation(SKI)

The aforementioned Kronecker Methods and Toeplitz Methods numerical tricks save the time complexity and are complementary to each other (could be applied at the same time, for example, when input data are **both** equally spaced **and** lie on a high-dimensional grid); however, they only apply when the input data strictly satisfy these properties, which is generally not the case. Thus, we manually introduce a equally spaced grid structure of points $U$, and approximate $K$ using local cubic interpolation ([2]):

$$K_{X,X} \approx W K_{U,U} W^T \triangleq K_{SKI}$$

where $W$ is a sparse matrix in that each row of $W$ contains only the four coefficient of the local cubic interpolation:

$$K_{X,U} \approx W K_{U,U}$$

Suppose there are $M$ inducing points $U$, then a matrix-vector multiplication with $K_{SKI}$ takes only $O(N + M^2)$, as $K_{U,U} \in \mathbb{R}^{M \times M}$ and $W \in \mathbb{R}^{N \times M}$ is sparse. If meanwhile we exploit the Kronecker Structure of $K_{U,U}$, time complexity becomes $O(PM^{\frac{P+1}{P}})$ and if we exploit the Toeplitz Structure of $K_{U,U}$, time complexity becomes $O(N + M \log M)$. We can bypass calculating $K_{SKI}^{-1}$ during the inference by linear conjugate gradient, which only requires $K_{SKI}$'s matrix vector product, and $\#\{\text{iteration}\} \ll N$

## 3 Preliminary Experiment Results

### 3.1 Kernel Reconstruction Error

We compare the approximate $K_{SKI}$ with the exact covariance matrix $K$, for 1000 (sorted) points sampled from $\mathcal{N}(0, 25)$ (Notice that they are not likely to form a grid structure)

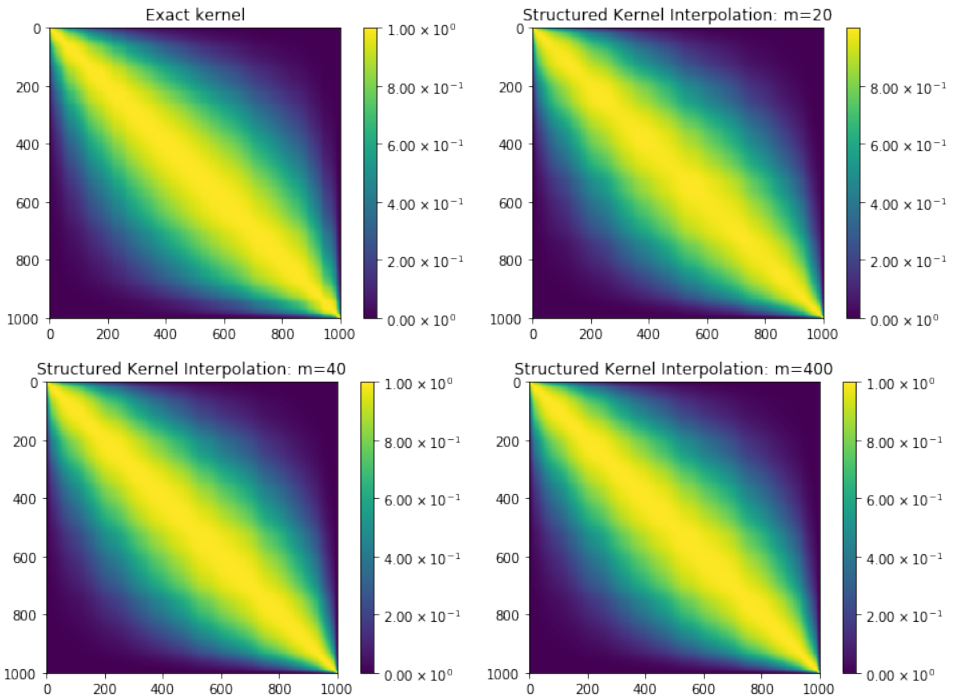


Figure 1: Structured Interpolated Kernel with increasing number of interpolating points compared with the exact kernel

We also plot the absolute difference between $K_{SKI}$ and the exact covariance matrix $K$, i.e.: $|K_{SKI} - K|$:

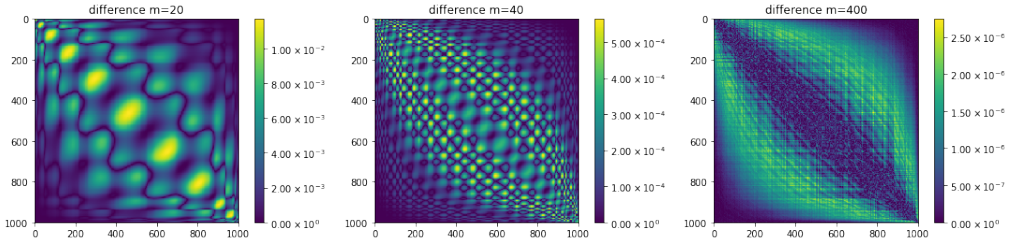


Figure 2: Absolute difference between Structured Interpolated Kernel with increasing number of interpolating points and the exact kernel

We have also plotted the $\|K_{SKI} - K\|_2$ (which is the maximum singular value) and $\|K_{SKI} - K\|_{max}$ (which is the maximum entry) vs the number of interpolating points $M$:

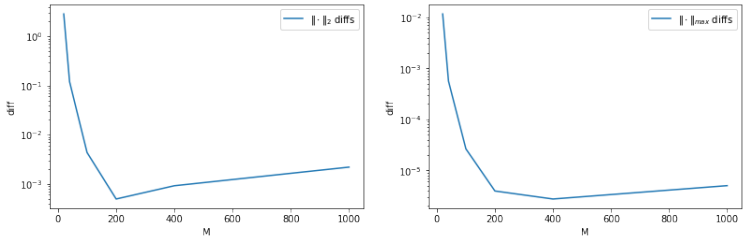


Figure 3: $\|K_{SKI} - K\|_2$ and $\|K_{SKI} - K\|_{max}$ vs number of interpolating points $M$

### 3.2 Image Reconstruction

With the help of 2.2, we are able to perform GP inference on a image which consists of tens of thousands of data points that would otherwise be a prohibiting time complexity for exact kernel calculation. Suppose that we have an small patch of occlusion that we want to predict for an image. We first separate the image into Red, Green and Blue channel, then we choose the posterior mean of the intensity for each channel and combine them up to form our final prediction.

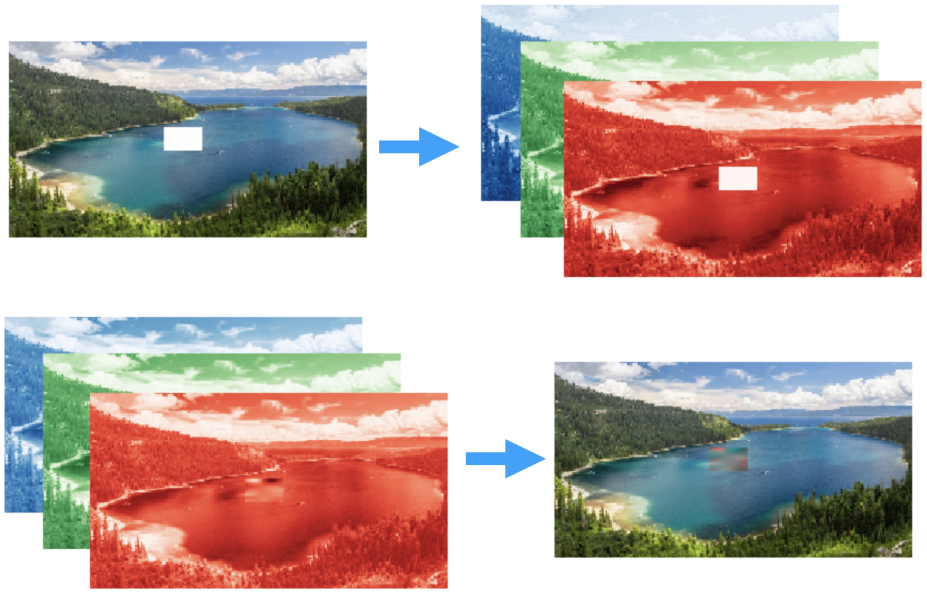

Figure 4: Top row: split an image into RGB channels. Bottom row: combine the prediction of each channel into final prediction

However, we observe that the result is not perfect, especially for the red channel, where there is a weird small patch of high intensity in the top left corner.

## 4 Plan

- For the Structured Kernel Interpolation, although GP doesn't seem to fit well for the patch prediction task, it might be due to the fact that there is a sudden change in the red channel intensity, near the island part. We could try Matérn kernel, which potentially has a higher "wiggliness". We are also proposing to use this method for super resolution task.
- As pointed out by [7], albeit the choice of kernel could dramatically affect the performance of the model, there is little attention on how to choose one. Typically people would select a fixed kernel a priori, or manually construct a kernel, both of which are rather restricted. Even when learning hyper-parameter of kernels, we are limited to a restrictive family of kernel functions. In fact, finite sample estimates will be affected by the kernel choice notwithstanding the use of an universal approximating kernel. We will be looking at Bayesian Nontparametric Kernel Learning[7]

We will re-examine the above problems and carry out experiments on more data sets.

## 5 Work Devision

Kaiji is mainly responsible for the theoretical part of the report and Sailun is mainly responsible for carrying out experiments.

## References


[1] Larry Wasserman. Nonparametric bayesian methods.

[2] Andrew Gordon Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1775–1784. JMLR.org, 2015.

[3] Andrew Gordon Wilson. *Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[4] Steven G. Johnson. Circulant-matrices, September 2017.

[5] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[6] Magnus Rudolph Hestenes and Eduard Stiefel. *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC, 1952.

[7] Arthur Gretton and Christian C. Robert, editors. *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, volume 51 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2016.