Gaussian Process Regression for Transportation System Estimation and Prediction Problems: the Deformation and a Hat Kernel

*Zhiyuan Liu, Member, IEEE, Cheng Lyu, Jinbiao Huo, Shuaian Wang, Jun Chen**

***Abstract*—Gaussian process regression (GPR) is an emerging machine learning model with potential in a wide range of transportation system estimation and prediction problems, especially those where the uncertainty of estimation needs to be measured, for instance, traffic flow analysis, the transportation infrastructure performance estimation problems and transportation simulation-based optimization problems. The kernel function is the core component of GPR, and the radial basis function (RBF) kernel is the most commonly used one, suitable for tasks without special knowledge about the patterns of data, like trend and periodicity. However, an inappropriate hyperparameter of the kernel function may lead to over-fitting or under-fitting of GPR. During hyperparameter optimization, the usage of the RBF kernel often suffers from the issue of failing to find the optimal hyperparameter. This paper aims to address this problem by promoting the use of the hat kernel, which can reduce the risk of under-fitting. Moreover, we propose the notion of deformation, corresponding to severe over-fitting of a GPR. To further address this issue, we investigate the connection between deformation and the Bayesian generalization error of GPR. Two lower bounds for the hyperparameter of the hat kernel are also proposed to avoid deformation of GPR.**

***Index Terms*—Hat kernel, Hyperparameter optimization, Gaussian process, Kernel machine, Lower bound**

I. Introduction

MACHINE learning models have drawn increasing attention from the transportation community in recent years [1-4]. The foremost reason for the burst of research on this topic can be attributed to the data richness. Apart from common loop detectors and floating car data, a variety of new data sources are currently available, including cell phone location data and social media data, which significantly enlarge the possibility to sense full-time citywide traffic dynamics [5-9]. In addition, another motivation for the use of machine learning tools lies in the growing demand for accuracy in real-time traffic system estimation and traffic management, e.g., the infrastructure deterioration evaluation. travel demand prediction, though weak in interpretability, can give highly accurate results so as to facilitate traffic management optimization, e.g., improved traffic safety, reduced delay for travellers and increased revenue for transportation service providers [10-13]. Finally, huge advances in computing capability provide strong support for practical applications of these new models with large amount of data. Among the machine learning models, Gaussian process regression is an emerging and efficient model potential in transportation system etimation and prediction problems [14, 15]. This paper focuses on Gaussian process regression and discusses issues pertaining to its training process.

Gaussian process regression (GPR) is a Bayesian modelling approach that adopts Gaussian process as a prior over prediction labels, any subset of which follows a multivariate Gaussian distribution [16]. Despite the Gaussian prior over labels, it does not assume the functional form of the mapping from inputs to outputs as models like linear regression do. Therefore, GPR is able to model complex nonlinear relationship between variables, and it has been proven that many types of neural networks with strong predictive ability converge to Gaussian process [17-19]. Also, compared with neural networks, the Gaussian prior allows GPR to make prediction using a limited number of hyperparameters, which is much easier to analyse and optimize. GPR has a wide range of applications, including traffic flow modelling, spatio-temporal prediction, and simulation-based optimization [20, 21]. Adaptations can also be made on GPR to solve classification problems [16].

The core component that grants GPR the ability of prediction is the kernel function. It determines the covariance between different samples in order to shape the similarity between them. The hyperparameters of the kernel function can significantly influence the prediction results. By maximising the marginal likelihood function, GPR can find the optimal hyperparameters that achieve a balance between model complexity and data-fitting quality. However, due to the nonconvexity of the objective function, i.e., marginal likelihood, it is prone to be stuck in local optimum, leading to severe over-fitting or under-fitting issue. The inappropriate selection of hyperparameters results in deformed GPR, limits the use of GPR on solving transportation problems

To overcome the above issue, it is necessary to study what will happen when inappropriate kernel hyperparameters are used and methods to alleviate it. To achieve this goal, we will show the advantage of the hat kernel over commonly used radial basis function (RBF) kernel, and present two lower bounds for its hyperparameter to avoid severe over-fitting of the model.

A. Literature Review

Research on GPR originates from the work by Williams and Rasmussen (1995) [22]. GPR is capable of employing limited samples to predict outputs of unseen inputs in the feasible set. There has been an increasing interest of using GP to solve problems in the transporation discipline. Two types of problems are usually addressed using GPR: traffic prediction problems [14, 23] and traffic management/control problems [24-26]. In traffic prediction problems, GPR is employed to capture the relationship between spatio-temporal features and traffic conditions . The traffic

- *Zhiyuan Liu, Cheng Lyu, Jinbiao Huo, Jun Chen* are with the Jiangsu Key Laboratory of Urban ITS, Jiangsu Province Collaborative Innovation Center of Modern Urban Traffic Technologies, School of Transportation, Southeast University, Nanjing, China. E-mail: zhiyuanl@seu.edu.cn, cheng.lyu@seu.edu.cn, Jinbiaoh@outlook.com, chenjun@seu.edu.cn. (*corresponding author).*
- *Shuaian Wang is with Department of Logistics & Maritime Studies, The Hong Kong Polytechnic University, Kowloon, Hong Kong. E-mail: hans.wang@polyu.edu.hk.*

management/control problems are mainly concerned with determining the optimal decision among various transport management and control strategies[27]. In solving traffic management/control problems, GPR, as the most commonly used surrogate model in Bayesian optimization, is generally used to construct surrogates of the objective function.

As introduced above, the kernel function is one of the essences of GPR, which, however, has been extensively studied early before. Machine learning models involving the kernel function are often termed kernel machines [28], such as kernel ridge regression and support vector machine. The aim of the kernel function is finding a mapping function to map the input data to a new vector space, such that their new representations can reflect the latent nonlinear characteristics of the data. Early research studied the kernel function in a purely mathematic way, mainly focusing on the properties of various kernel functions, the condition for a kernel function to be valid, and the rule of constructing new kernel functions. Genton [29] presented an overview of typical types of kernel functions commonly used by kernel machines. Based on the properties of the kernel, the author categorised kernel functions into anisotropic stationary kernels, isotropic stationary kernels, compactly supported kernels, locally stationary kernels, nonstationary kernels, and separable nonstationary kernels. Of all these types of kernel functions, researchers are mostly concerned about the former four, which are all stationary kernels. Stationary kernels express the similarity between samples as a function of their distance, which is translation invariant.

Another line of research studied the kernel machine in a more pragmatic way. Most recent studies on GPR focus on four topics, namely the fast approximation of GPR, the adaptation to noisy or discrete data, the combination with Bayesian optimization, and its practical applications in various subjects. The demand for the fast approximation of GPR arises from its high time complexity involved in matrix inversion. A general solution is to sample a limited set of points instead of the full dataset and use the sampled points to estimate the GPR [30]. One of its improvements is the combination with variational inference, which approximates the GPR by minimising the discrepancy between variational distribution and original posterior distribution [31]. The second topic arises because a standard GPR addresses data in a continuous space and does not allow for noises in the training data. Modifications were made on the kernel to solve the problem. For example, Tomczak et al. [32] defined the kernel function value to be zero between inputs of different categories. As for the third topic, Bayesian optimization is an emerging method for black-box optimization, often applied in simulation-based optimizations [33] and hyperparameter tuning of deep neural networks [20]. GPR is generally adopted as the base surrogate model to describe the surface of optimization target. Finally, GPR was applied in many practical problems as a predictive tool, such as traffic volume prediction [21] and route planning.

Selecting appropriate parameters to avoid underfitting/overfitting issues is critical for constructing machine learning methods. Specially, the overfitting issue, making the model fail to generalize from observed data to unseen data, has been a major area of interest within the field of machine learning [34, 35]. Although one of the most important applications of GPR is hyperparameter optimization, its own kernel hyperparameter also needs optimizing. The limited number of hyperparameters has long been labelled an advantage of GPR over neural networks due to less effort devoted to hyperparameter optimization. However, it is practically found that optimizing these hyperparameters is a nontrivial task. It has been revealed in transportation practice that inappropriate hyperparameters affect the performance of GPR significantly both in prediction and optimization tasks. It can be further enhanced if there is a guidance on the choice of its hyperparameter.

B. Objectives and Contributions

The objective of this paper is to offer a thorough analysis of the hyperparameter optimization process of GPR. The contributions of this study are three-fold. First, we analyse the possible cause of over-fitting and under-fitting of a GPR model in terms of the choice of kernel hyperparameter. Then, we attempt to amend the under-fitting issue by promoting the use of the hat kernel. To further alleviate the risk of over-fitting, we identify the deformation problem of GPR given small hyperparameter value and propose the lower bound for optimizing the hyperparameter of the hat kernel.

The nonconvexity of the optimization objective of GPR, i.e., marginal likelihood function, is often resolved using a multi-start gradient-based optimizer, which lacks theoretical guidance. To better address the problem, we investigate the objective function and decompose it into two key components, one for model complexity and another for data-fitting quality. Both of the identification of the deformation problem as well as the lower bound analysis are based on the analysis and interpretation of the objective function.

The remainder of the paper is organised as follows. Section II provides a brief introduction to GPR. Section III presents the properties of two kernel functions and discusses the difficulty in optimizing kernel hyperparameters. Then, a thorough analysis of the optimization objective function, the definition of deformation, and its connection with Bayesian generalization error are elucidated in Section IV. Two lower bounds of kernel hyperparameter are also proposed in this section. Finally, conclusions are shown in Section V.

II. Preliminaries of Gaussian Process Regression

This section provides a preliminary introduction to Gaussian process regression (GPR), including the prior and posterior of GPR, as well as the basic concept concerning the kernel function.

A. Prior and posterior

Similar to other machine learning models for the task of supervised learning, GPR aims at identifying a mapping $f:\mathbf{x}\mapsto y$, where $\mathbf{x}\in\mathbb{R}^m$ is an m-dimensional input vector, and $y\in\mathbb{R}$ is the output label. As a Bayesian approach, GPR models each output label y as a random variable Y. A Gaussian process (GP) assumes that any label Y follows a Gaussian distribution $\mathsf{N}(\mu,\sigma^2)$, and any collections of multiple labels $\mathbf{Y}$ together follows a multivariate Gaussian distribution $\mathsf{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$.

Before training a GPR model, we need to place a prior over the outputs. For an arbitrary input $\mathbf{x}_*\in\mathbb{R}^m$, the random variable Y_* corresponding to the output $f(x_*)$ is assumed to follow a prior distribution,

$$Y_* \sim \mathsf{N}\left(\mu(\mathbf{x}_*), \kappa(\mathbf{x}_*, \mathbf{x}_*)\right) \tag{1}$$

where $\mu(\cdot)$ denotes the mean function and $\kappa(\cdot,\cdot)$ denotes the kernel function. For simplicity, the mean function is often assumed to be constantly zero, that is,

$$Y_* \sim \mathsf{N}\left(0, \kappa(\mathbf{x}_*, \mathbf{x}_*)\right) \tag{2}$$

Such simplification does not affect the model as the mean function is merely an offset of the label. If an arbitrary mean function is needed, the subtraction of the mean function value from the label can be modelled instead.

Denote a training dataset containing n samples as $D = (\mathbf{x}_{1:n}, \mathbf{y}_{1:n})$, where $\mathbf{x}_{1:n} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^{\acute{u}}$, $\mathbf{y}_{1:n} = [y_1, y_2, \ldots, y_n]^{\acute{u}}$, and the random vector corresponding to $\mathbf{y}_{1:n}$ as $\mathbf{Y}_{1:n}$. The joint prior distribution of Y_* and $\mathbf{Y}_{1:n}$ can be written as follows,

$$\begin{pmatrix} \mathbf{Y}_{1:n} \\ Y_* \end{pmatrix} \sim \mathsf{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K} & \boldsymbol{\kappa}_* \\ \boldsymbol{\kappa}_*^{\acute{u}} & \kappa_{**} \end{pmatrix}\right) \tag{3}$$

where $\mathbf{K}$ is a $n \times n$ kernel matrix, whose element on the i th row and j th column is $\kappa(\mathbf{x}_i, \mathbf{x}_j)$, $\boldsymbol{\kappa}_* = [\kappa(\mathbf{x}_1, \mathbf{x}_*), \kappa(\mathbf{x}_2, \mathbf{x}_*), \ldots, \kappa(\mathbf{x}_n, \mathbf{x}_*)]^{\acute{u}}$, and $\kappa_{**} = \kappa(\mathbf{x}_*, \mathbf{x}_*)$.

By conditioning the joint distribution on observed sample data, the posterior distribution of Y_* can be obtained,

$$Y_* \mid x_*, \mathbf{x}_{1:n}, \mathbf{y}_{1:n} \sim \mathsf{N}\left(\boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathbf{y}, \kappa_{**} - \boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\boldsymbol{\kappa}_*\right) \tag{4}$$

where the posterior mean $\boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathbf{y}$ is the point estimate of Y_* and the posterior variance $\kappa_{**} - \boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\boldsymbol{\kappa}_*$ is a measure of the estimation uncertainty.

B. Kernel function and hyperparameter optimization

It can be observed from the prior and posterior distribution of Y_* that the kernel function plays a crucial role in GPR estimation. We notice that the kernel function essentially determines the covariance matrix of the multivariate Gaussian distribution in a GP, indicating the similarity between each two samples. For generic tasks without prior knowledge about special patterns in data (e.g., periodicity and a growing trend), we tend to assume that similar input vectors lead to similar outputs, and the similarity only correlates with the relative distance between two samples. Here, we would like to present two definitions.

Definition 1 (Stationary kernel). *A kernel function $\kappa(\mathbf{x}, \mathbf{x}')$ is stationary if it is a function only of the distance between two input samples $d = |\mathbf{x} - \mathbf{x}'|$.*

Definition 2 (Monotonic kernel). *A kernel function $\kappa(\mathbf{x}, \mathbf{x}')$ is monotonic if it is stationary and is monotonically nonincreasing with respect to the distance between two input samples $d = |\mathbf{x} - \mathbf{x}'|$.*

A monotonic kernel can well suit many practical applications, such as the hyperparameter optimization of deep neural networks, hence the focus of this study.

Once the functional form of the kernel function is determined, training a GPR model becomes a search for the optimal hyperparameter of the kernel function. Denote the hyperparameter of the kernel function as θ. The optimal hyperparameter θ^* can be obtained by maximising the probability of observing $\mathbf{y}_{1:n}$ given input training data $\mathbf{x}_{1:n}$ and the kernel function, i.e., the marginal likelihood (ML) of GPR. As $\mathbf{Y}_{1:n}$ also follows a multivariate Gaussian distribution $\mathsf{N}(\mathbf{0}, \mathbf{K})$, the ML can be formulated as follows,

$$p(\mathbf{y}_{1:n} \mid \mathbf{x}_{1:n}; \theta) = \frac{1}{(2\pi)^{n/2} \mid K \mid^{1/2}} \exp\left(-\frac{1}{2}\mathbf{y}^{\acute{u}}\mathbf{K}^{-1}\mathbf{y}\right) \tag{5}$$

To facilitate calculation, its logarithm form, i.e., the log marginal likelihood (LML), is often adopted instead,

$$\begin{aligned} &\log p(\mathbf{y}_{1:n} \mid \mathbf{x}_{1:n}; \theta) \\ &= \log\left[\frac{1}{(2\pi)^{n/2} \mid K \mid^{1/2}} \exp\left(-\frac{1}{2}\mathbf{y}^{\acute{u}}\mathbf{K}^{-1}\mathbf{y}\right)\right] \\ &= \log\left[(2\pi)^{-n/2} \mid K \mid^{-1/2}\right] - \frac{1}{2}\mathbf{y}^{\acute{u}}\mathbf{K}^{-1}\mathbf{y} \\ &= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log \mid \mathbf{K} \mid - \frac{1}{2}\mathbf{y}^{\acute{u}}\mathbf{K}^{-1}\mathbf{y} \end{aligned} \tag{6}$$

The optimization problem $\arg\max_\theta \log p(\mathbf{y}_{1:n} \mid \mathbf{x}_{1:n}; \theta)$ does not have a closed-form solution. As the gradient of LML can be evaluated, gradient based optimizers, such as the Newton method can be adopted to search for the optimal hyperparameter. However, the nonconvexity of the optimization objective brings difficulty to the problem. A common compromised solution is to try multiple initial points for the Newton method and may still experience failure if the initial points are not properly sampled. More discussions on this issue are presented in the following section.

III. Hat Kernel for Gaussian Process Regression

In this section, we discuss the issues on the hyperparameter optimization of the GPR, and comparison between the radial basis function (RBF) kernel and the hat kernel is presented.

A. RBF kernel

One of the most common kernel functions used in GPR is the RBF kernel, also referred to as the Gaussian kernel.

$$\kappa_{\mathrm{RBF}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left[-\frac{1}{2}\left(\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2}{l}\right)^2\right] \tag{7}$$

where $\|\cdot\|_2$ denotes the ℓ_2 norm, and the length scale l is the hyperparameter of the RBF kernel.

The RBF kernel is popular in a variety of tasks and is used as the default setting in various machine learning toolboxes (e.g., Scikit-learn [36]) since it is infinitely differentiable and can yield a smooth posterior estimate. However, optimizing the hyperparameter for the RBF kernel can be uneasy. For the sake of demonstration, a simple sinusoidal function is presented as an example (see Figure 1, the x-axis displays the variable x, the y-axis displays the function $y = \sin x + 5$). We also randomly sampled 50 points on the function curve between -20 to 20.

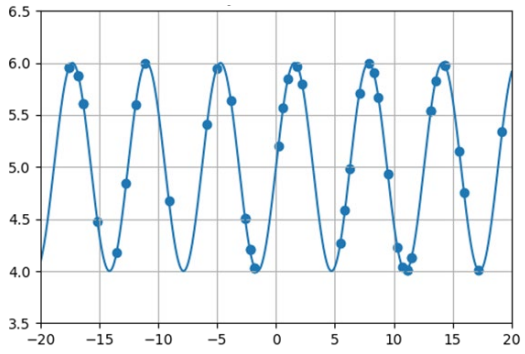


Figure 1: An example sinusoidal function $y = \sin x + 5$ and sampled points.

Then, the LML curve with respect to the length scale of the RBF kernel is plotted in Figure 2. We transform the y-axis to $100 - LML$ for the readability of the figure. The candidate length scale values range from 10^{-5} to 10^{5}. Obviously, the LML curve stays nearly constant for a wide range of length scale values. If a gradient-based optimizer is used for hyperparameter optimization, it is prone to be stuck in the local optimum. The LML curve suggests that the global optimum can be hopefully found only when the initial value of the optimizer falls between 10^{-1} and 10.

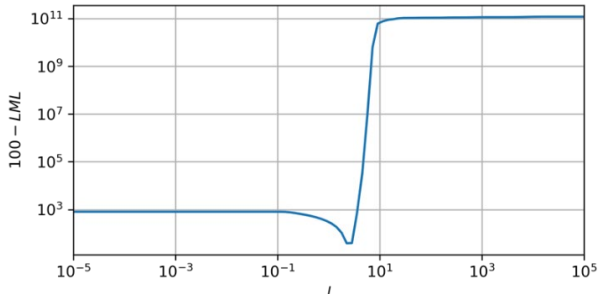


Figure 2: Log marginal likelihood curve with respect to the length scale of the RBF kernel.

B. Hat Kernel

As an attempt to address this problem, we present the use of the hat kernel, as defined below.

$$\kappa_{\text{hat}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \left(1 - \frac{|\,\mathbf{x}_i - \mathbf{x}_j\,|}{l}\right)_+ \tag{8}$$

where $(\cdot)_+$ represents $\max(0, \cdot)$. Both the hat kernel and the RBF kernel are monotonic kernels since they are both nonincreasing functions of the distance between two sample points, $|\,\mathbf{x}_i - \mathbf{x}_j\,|$. The property of monotonicity allows them to be widely applicable as no strong prior assumptions are embedded except the simple one that similar inputs generate closer outputs.

There are two major differences between the hat kernel and the RBF kernel in terms of differentiability and influencing threshold. First, it is obvious that the hat kernel function is nondifferentiable when $|\,\mathbf{x}_i - \mathbf{x}_j\,|$ equals l, implying that the posterior mean function is highly likely to be also nondifferentiable. Second, the hat kernel is a kernel function with a compact support, indicating the function value will always be zero when the distance between two input points is larger than a specific threshold.

The hat kernel is a special case of piecewise polynomial functions . Some examples of valid piecewise polynomial functions $\kappa_{\text{ppR},q}(\mathbf{x}_i, \mathbf{x}_j)$ are presented below.

$$\begin{aligned}
\kappa_{\text{ppR},0}(\mathbf{x}_i, \mathbf{x}_j) &= (1-d)_+^s \\
\kappa_{\text{ppR},1}(\mathbf{x}_i, \mathbf{x}_j) &= (1-d)_+^{s+1}\left[(s+1)d+1\right] \\
\kappa_{\text{ppR},2}(\mathbf{x}_i, \mathbf{x}_j) &= (1-d)_+^{s+2}\left[\left(s^2+4s+3\right)d^2+(2s+6)d+3\right]/3 \\
\kappa_{\text{ppR},3}(\mathbf{x}_i, \mathbf{x}_j) &= (1-d)_+^{s+3}\left[\left(s^3+9s^2+23s+15\right)d^3\right. \\
&\quad \left.+\left(6s^2+36s+45\right)d^2+(15j+45)d+15\right]/15
\end{aligned} \tag{9}$$

where $d = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ and $s = \lfloor R/2 \rfloor + q + 1$. The piecewise polynomial function becomes the hat kernel when $q = 0$, $s = 1$, and $R = 1$. It is worth noting that it is nontrivial to ensure piecewise polynomial functions to be positive definite. They can be safely used only when the number of feature dimensions does not exceed R (Wendland, 2004). Therefore, to ensure the validity of the hat kernel, it should not be applied to problems with more than one input dimension. The discussion in this paper below focuses on the one-dimensional case, and input vectors like $\mathbf{x}_i$ will be written as scalars like x_i.

We also apply the hat kernel to the same example problem as experimented using the RBF kernel. The LML curve with respect to the length scale is plotted in Figure 3. Numerical instability can be observed around the length scale value of 10, which does not affect the general trend of the curve. In comparison to the LML curve of the RBF kernel, the length scale range that is easy to optimize, i.e., between 10^{-1} to 10^{5}, is much larger when using the hat kernel.

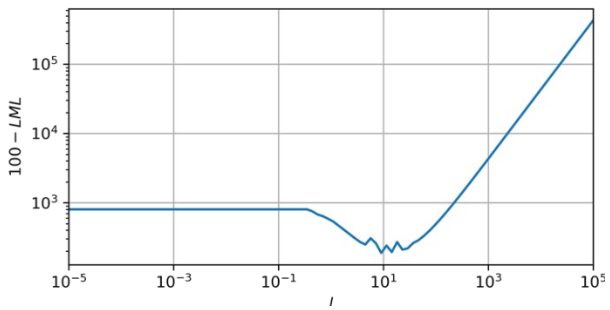


Figure 3: Log marginal likelihood curve with respect to the length scale of the hat kernel.

According to Figure 3, the hat kernel does not solve all the problems involved in hyperparameter optimization. When the length scale value is small, say, less than 10^{-1}, the LML curve of the hat kernel still stays nearly constant.

IV. Deformation of Gaussian Process Regression

The deformation phenomenon of GPR is observed based on our investigations on some transportation problems. In this section, we are going to examine the reason why the LML curve is nearly constant when the length scale of the kernel is small, and the related phenomenon of deformation in GPR.

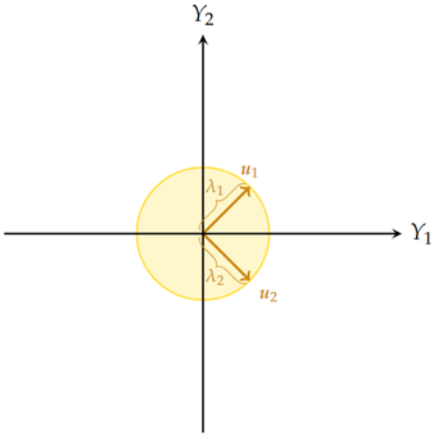
A. Interpretation of LML

Recall that the LML of GPR is composed of three terms, namely the model complexity term $-\frac{1}{2}\log|\mathbf{K}|$, the data-fitting term $-\frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{y}$ and the constant term $-\frac{n}{2}\log(2\pi)$. Optimizing the hyperparameters of a kernel is essentially a trade-off between model complexity and data-fitting quality. An extremely complex model can perfectly fit all the training samples with zero error. However, severe over-fitting issue will arise since the model cannot generalize to any other data, and reducing the model complexity can alleviate this issue. If we turn to the other extreme of a simple model that always yields constant outputs, the data-fitting quality will be unsatisfactory, and the model is of little value in applications. Therefore, it is expected to find an appropriate set of hyperparameters that fit the data well without adding more complexity to the model.

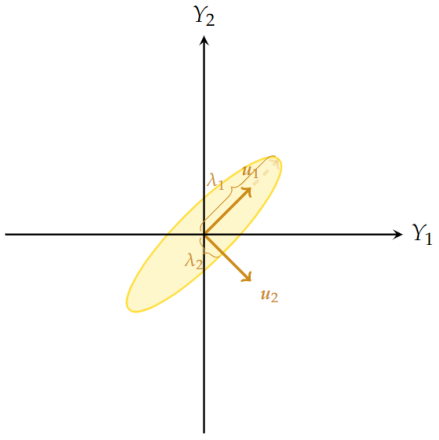
Let us consider the model complexity term first. The core component of this term is the determinant of the kernel matrix $|\mathbf{K}|$. Think of a case in which there are two sample points, i.e., $n=2$ and $\mathbf{K}\in\mathrm{R}^{2\times 2}$. The kernel matrix can be visualised through eigen-decomposition, as demonstrated in Figure 4.

$$\mathbf{K}=\sum_{i=1}^{n}\lambda_i\mathbf{u}_i\mathbf{u}_i^{\mathrm{T}} \tag{10}$$

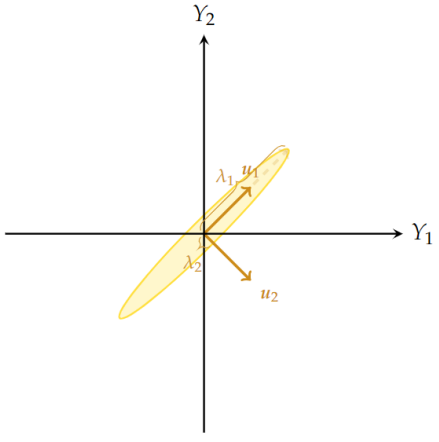
where we denote the two eigenvalues of $\mathbf{K}$ by λ_1 and λ_2 and assume $\lambda_1\geq\lambda_2$ without loss of generality. For each eigenvalue λ_i, its corresponding unit eigenvector is denoted by $\mathbf{u}_i$. The ellipsoid in the figure has a major axis in the same direction as $\mathbf{u}_1$, and the length of its semi-major axis equals to the corresponding eigenvalue λ_1. Regarding the minor axis, it has the same direction as the other eigenvector $\mathbf{u}_2$, and the length of the semi-minor axis equals to λ_2.



(a): $l=0.1$



(b): $l=5$



(c): $l=10$

Figure 4: Ellipsoid of a hat kernel matrix with different length scales.

As can be observed, the larger the length scale, the flatter the ellipsoid. Correspondingly, we can conclude that $|\mathbf{K}|$ is a monotonically non-decreasing function of the length scale l, if the kernel function is the function of d/l and monotonically non-decreasing with respect to l, where d is the sample distance. This is obvious in the two-dimensional case. Given two length scale values l_1 and l_2, and suppose $l_1<l_2$. The kernel matrix can be written as

$$\mathbf{K}_{l_1}=\begin{pmatrix}k_{11,l_1} & k_{12,l_1}\\ k_{21,l_1} & k_{22,l_1}\end{pmatrix},\mathbf{K}_{l_2}=\begin{pmatrix}k_{11,l_2} & k_{12,l_2}\\ k_{21,l_2} & k_{22,l_2}\end{pmatrix} \tag{11}$$

where $k_{11,l_1}=k_{22,l_1}=k_{11,l_2}=k_{22,l_2}$ are all constant with respect to l since $x_1-x_2=0$. Also, we have $k_{12,l_1}\leq k_{12,l_2}$ and $k_{22,l_1}\leq k_{22,l_2}$. Since the kernel matrix is symmetric, we have

$$\begin{aligned}|\mathbf{K}_{l_1}|-|\mathbf{K}_{l_2}| &= \left(k_{11,l_1}^2-k_{21,l_1}^2\right)-\left(k_{11,l_2}^2-k_{21,l_2}^2\right)\\ &= k_{21,l_2}^2-k_{21,l_1}^2\geq 0.\end{aligned} \tag{12}$$

This conclusion can be generalized to more samples. The covariance between sample points is more likely to be zero when a small length scale is adopted, indicating the samples are loosely correlated or even uncorrelated even when they are close to each other. It results in a flexible model and high model complexity.

Conversely, the covariance will increase as the length scale increases, leading to a kernel matrix with highly correlated row vectors and, accordingly, a small determinant. The following proposition and proof will provide a more rigorous presentation of the monotonicity of $|\mathbf{K}|$ with respect to length scale using its derivative.

Proposition 1. *Given a kernel matrix* $\mathbf{K}$ *parameterised by length scale* l*, where the kernel function is a function of* d/l *and monotonically non-decreasing with respect to* l*, the determinant of the kernel matrix* $|\mathbf{K}|$ *is monotonically non-increasing with respect to* l*.*

Proof. The derivative of $|\mathbf{K}|$ is computed by,

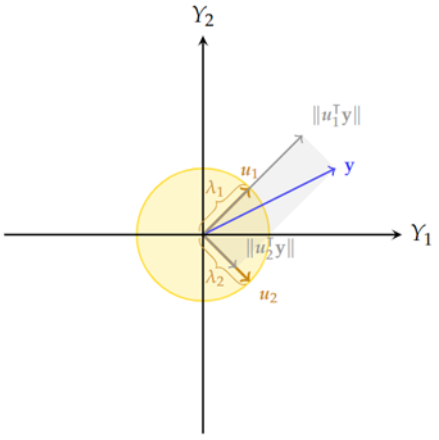
$$\frac{\partial|\mathbf{K}|}{\partial l}=\frac{1}{|\mathbf{K}|}\mathrm{tr}\left(\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial l}\right). \tag{13}$$

As $\mathbf{K}$ is positive definite, we have $|\mathbf{K}|>0$ and $\mathrm{tr}\left(\mathbf{K}^{-1}\right)>0$. Since the kernel function is a function of d/l, its value equals 0 when $d=0$. Hence, $\partial\mathbf{K}/\partial l$ is a hollow matrix, following that $\mathrm{tr}\left(\partial\mathbf{K}/\partial l\right)=0$. Using the trace inequality of symmetric matrices, we have

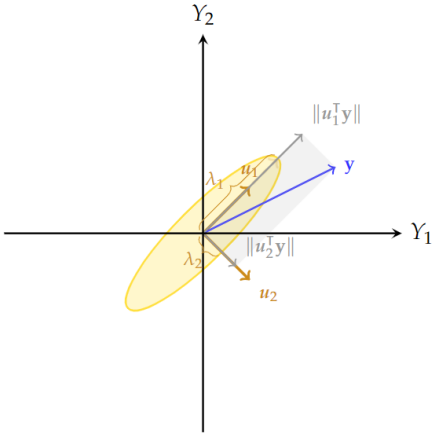
$$\mathrm{tr}\left(\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial l}\right)\le\mathrm{tr}\left(\mathbf{K}^{-1}\right)\mathrm{tr}\left(\frac{\partial\mathbf{K}}{\partial l}\right)=0. \tag{14}$$

Therefore, $\partial|\mathbf{K}|/\partial l\le 0$, indicating that the determinant of the kernel matrix $|\mathbf{K}|$ is monotonically non-increasing with respect to l.

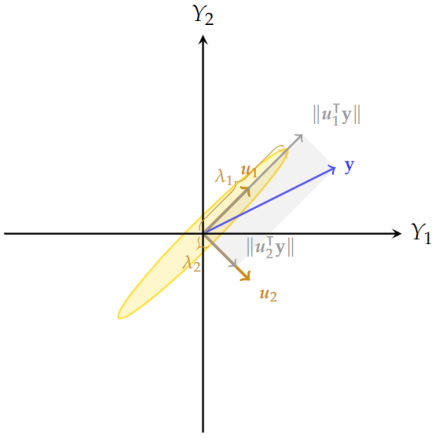
The other term of the LML, $-\frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{y}$, reflects the fitting performance of the model on the training data. Again, consider the simple case with two sample points. The vector of sample labels $\mathbf{y}=[y_1,y_2]^{\mathrm{T}}$ is further added to the figure, as demonstrated in Figure 5.



(a): $l=0.1$



(b): $l=5$



(c): $l=10$

Figure 5: Ellipsoid of a hat kernel matrix and the vector of sample labels.

The blue vector in the figure represents the label vector. The ellipsoid plot can intuitively show whether a kernel matrix is potential to fit the data well. It is expected that the ellipsoid tilts towards the direction of $\mathbf{y}$, otherwise $\mathbf{y}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{y}$ will be large. Based on (7) the term $\mathbf{y}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{y}$ can be rewritten as

$$\mathbf{y}^{\mathrm{T}}\mathbf{K}^{-1}\mathbf{y}=\sum_{i=1}^{n}\frac{1}{\lambda_i}\|\mathbf{u}_i^{\mathrm{T}}\mathbf{y}\|^2. \tag{15}$$

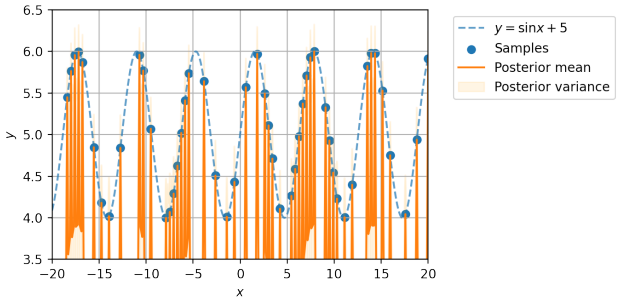
Therefore, it is the weighted sum of squared length of $\mathbf{y}$'s projection on each eigenvector. For instance, in Figure 5, if length scale l keeps growing to a large value, the projection of $\mathbf{y}$ on $\mathbf{u}_2$ will be penalised due to the large eigenvalue of $\mathbf{K}^{-1}$, i.e., $1/\lambda_2$. Among the three subfigures of Figure 5, (b) gives the best fitting as the kernel matrix achieves a reasonable balance between squared projected length.

B. Deformation and over-fitting

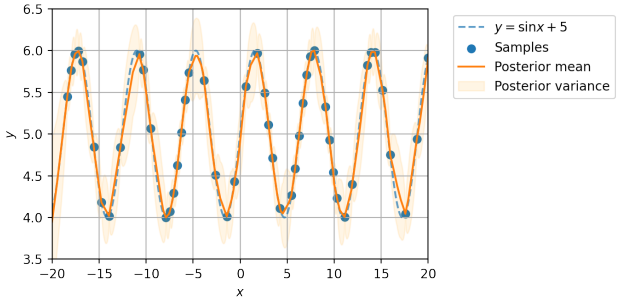
1) Intuitive demonstration of deformation

It can be concluded from the previous section that, when the length scale l of the kernel function is small, the model can fit the training data well while the model complexity is high, indicating the over-fitting issue of the model. If we plot the posterior mean curve of the model, as demonstrated in Figure 6 (a), we can observe that a small length scale corresponds with an extremely

distorted curve, which stays near the prior mean except the neighbourhood of training samples. Many jerks show up in the posterior mean curve as the model is too complex and overly flexible. Therefore, this phenomenon is referred to as the deformation of GPR, meaning that severe over-fitting occurs. In comparison, an undeformed curve can better restore the shape of the ground truth behind training samples, as demonstrated in Figure 6 (b). The deformation phenomon is usually revealed in transportation practice, in which the inputs data set is generally large and with noise, leading to inappropriate value of length scales.



(a): Hat kernel, $l = 0.3$



(b): Hat kernel, $l = 10$

Figure 6: Posterior mean curve using the hat kernel with different length scales.

2) Bayesian generalization error

More formally, we will look into the deformation and over-fitting issue in terms of the generalization error. In the example above, this phenomenon is described by observing whether the fitted curve is distorted or not, which does not give a rigorous and universal criterion. Instead, the generalization error is a better indicator, which measures the estimation error of the model on unseen samples. Denote the posterior mean estimate at x_* of a GPR trained on dataset D as $h_D(x_*)$. The empirical generalization error at x_* of this model is written as,

$$\mathsf{E}_D(x_*) = \ell\left(h_D(x_*), y_*\right) = \left[h_D(x_*) - y_*\right]^2 \tag{16}$$

where the loss function ℓ is set as mean squared error (MSE), and y_* denotes the true label at x_*.

Before acquiring the ground truth labels of all test samples, we are not able to compute the exact empirical generalization error. It should be aware that it is impossible and meaningless to evaluate the model over all samples in the input space. The indicator of more interest in real applications is the expected generalization error. The Bayesian generalization error, i.e., the expectation of the empirical generalization error, at x_* of this model is written as,

$$\mathsf{E}\left[\mathsf{E}_D(x_*)\right] = \mathsf{E}\left[\ell\left(h_D(x_*), y_*\right)\right] = \mathsf{E}\left[\left[h_D(x_*) - y_*\right]^2\right] \tag{17}$$

Recall the expression for the posterior mean estimate of GPR in (4), and substitute $h_D(x_*)$ with it, which yields,

$$\begin{aligned}
\mathsf{E}\left[\mathsf{E}_D(x_*)\right] &= \mathsf{E}\left[\left(h_D(x_*) - y_*\right)^2\right] \\
&= \mathsf{E}\left[\left(\boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathbf{y} - y_*\right)^2\right] \\
&= \mathsf{E}\left[\boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathbf{y}\mathbf{y}^{\acute{u}}\mathbf{K}^{-1}\boldsymbol{\kappa}_* - 2\boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathbf{y}y_* + y_*^2\right] \\
&= \mathsf{E}\left[\boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathbf{y}\mathbf{y}^{\acute{u}}\mathbf{K}^{-1}\boldsymbol{\kappa}_*\right] - 2\mathsf{E}\left[\boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathbf{y}y_*\right] \\
&\quad + \mathsf{E}\left[y_*^2\right].
\end{aligned} \tag{18}$$

Here, the training labels follow the prior distribution $\mathbf{Y} \sim \mathsf{N}(\mathbf{0}, \mathbf{K})$. Therefore, $\mathsf{E}\left(\mathbf{y}\mathbf{y}^{\acute{u}}\right) = \mathbf{K}$, and (18) can be further simplified as,

$$\begin{aligned}
\mathsf{E}\left[\boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathbf{y}\mathbf{y}^{\acute{u}}\mathbf{K}^{-1}\boldsymbol{\kappa}_*\right] &= \mathrm{tr}\left[\mathbf{K}^{-1}\boldsymbol{\kappa}_*\boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathsf{E}\left(\mathbf{y}\mathbf{y}^{\acute{u}}\right)\right] \\
&= \mathrm{tr}\left[\mathbf{K}^{-1}\boldsymbol{\kappa}_*\boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathbf{K}\right] \\
&= \mathrm{tr}\left[\mathbf{K}^{-1}\boldsymbol{\kappa}_*\boldsymbol{\kappa}_*^{\acute{u}}\right] \\
&= \mathrm{tr}\left[\boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\boldsymbol{\kappa}_*\right] \\
&= \boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\boldsymbol{\kappa}_*.
\end{aligned} \tag{19}$$

Concerning the second term, it can be rewritten as,

$$\mathsf{E}\left[\boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathbf{y}y_*\right] = \boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathsf{E}\left[\mathbf{y}y_*\right] \tag{20}$$

Similarly, according to the joint prior in (3), we have $\mathsf{E}\left[\mathbf{y}y_*\right] = \boldsymbol{\kappa}_*$. Then, (20) can be further simplified,

$$\begin{aligned}
\mathsf{E}\left[\boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathbf{y}y_*\right] &= \boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\mathsf{E}\left[\mathbf{y}y_*\right] \\
&= \boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\boldsymbol{\kappa}_*.
\end{aligned} \tag{21}$$

Concerning the third term, it can be rewritten as,

$$\begin{aligned}
\mathsf{E}\left[y_*^2\right] &= \mathsf{E}(y_*)^2 + \mathsf{V}(y_*) \\
&= 0 + \kappa(x_*, x_*) \\
&= \kappa_{**}
\end{aligned} \tag{22}$$

where $\mathsf{V}(\cdot)$ denotes the variance of a random variable.

Combine all three terms together, the Bayesian generalization error is,

$$\mathsf{E}\left[\mathsf{E}_D(x_*)\right] = \kappa_{**} - \boldsymbol{\kappa}_*^{\acute{u}}\mathbf{K}^{-1}\boldsymbol{\kappa}_* \tag{23}$$

Nonetheless, the Bayesian generalization error above is the value in ideal case. In most common cases, we do not know the exact covariances between each two input vectors; if we know, they can hardly be accurately described by simple kernel functions such as the RBF kernel and the hat kernel. Our prior about the use of the kernel function, which might be biased, only encodes our belief in the relationships between different inputs.

Denote the Bayesian generalization error at x_* using an unbiased kernel function and a biased kernel function as $\mathsf{E}\left[\mathsf{E}_D^u(x_*)\right]$ and $\mathsf{E}\left[\mathsf{E}_D^b(x_*)\right]$, respectively. It was argued that the use of a biased kernel function leads to an increased Bayesian generalization error [37]. We first write the empirical generalization error $\mathsf{E}_D^u(x_*)$,

$$\mathsf{E}_D^u(x_*)=\left(\boldsymbol{\kappa}_{*(u)}^{\text{ú}}\mathbf{K}_{(u)}^{-1}\mathbf{y}-y_*\right)^2 \tag{24}$$

where the subscript (u) indicates the use of an unbiased kernel function, i.e., the latent true kernel function that determines the covariance between input vectors. And its Bayesian generalization error is,

$$\mathsf{E}\left[\mathsf{E}_D^u(x_*)\right]=\kappa_{**(u)}-\boldsymbol{\kappa}_{*(u)}^{\text{ú}}\mathbf{K}_{(u)}^{-1}\boldsymbol{\kappa}_{*(u)} \tag{25}$$

The other empirical generalization error $\mathsf{E}_D^u(x_*)$ can be expressed as,

$$\mathsf{E}_D^b(x_*)=\left(\boldsymbol{\kappa}_{*(b)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\mathbf{y}-y_*\right)^2 \tag{26}$$

where the subscript (b) indicates the use of a biased kernel function, i.e., the designated prior kernel function. And its Bayesian generalization error is,

$$\begin{aligned}\mathsf{E}\left[\mathsf{E}_D^b(x_*)\right]=&\kappa_{**(u)}-2\boldsymbol{\kappa}_{*(u)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\\&+\boldsymbol{\kappa}_{*(b)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\mathbf{K}_{(u)}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\end{aligned} \tag{27}$$

where detailed reduction is provided in Appendix A.

Proposition 2: $\mathsf{E}\left[\mathsf{E}_D^u(x_*)\right]\le\mathsf{E}\left[\mathsf{E}_D^b(x_*)\right]$.

Proof: Denote the difference between two generalization errors as $\Delta\mathsf{E}_D=\mathsf{E}_D^b(x_*)-\mathsf{E}_D^u(x_*)$. Therefore, we have,

$$\begin{aligned}\mathsf{E}\left[\Delta\mathsf{E}_D\right]&=\mathsf{E}\left[\mathsf{E}_D^b(x_*)-\mathsf{E}_D^u(x_*)\right]\\&=\kappa_{**(u)}-2\boldsymbol{\kappa}_{*(u)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\\&\quad+\boldsymbol{\kappa}_{*(b)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\mathbf{K}_{(u)}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\\&\quad-\left(\kappa_{**(u)}-\boldsymbol{\kappa}_{*(u)}^{\text{ú}}\mathbf{K}_{(u)}^{-1}\boldsymbol{\kappa}_{*(u)}\right)\\&=\kappa_{**(u)}-2\boldsymbol{\kappa}_{*(u)}^{\text{ú}}\mathbf{K}_{(u)}^{-1}\mathbf{K}_{(u)}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\\&\quad+\boldsymbol{\kappa}_{*(b)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\mathbf{K}_{(u)}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\\&\quad-\kappa_{**(u)}+\boldsymbol{\kappa}_{*(u)}^{\text{ú}}\mathbf{K}_{(u)}^{-1}\mathbf{K}_{(u)}\mathbf{K}_{(u)}^{-1}\boldsymbol{\kappa}_{*(u)}\\&=\boldsymbol{\kappa}_{*(b)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\mathbf{K}_{(u)}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\\&\quad-2\boldsymbol{\kappa}_{*(u)}^{\text{ú}}\mathbf{K}_{(u)}^{-1}\mathbf{K}_{(u)}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\\&\quad+\boldsymbol{\kappa}_{*(u)}^{\text{ú}}\mathbf{K}_{(u)}^{-1}\mathbf{K}_{(u)}\mathbf{K}_{(u)}^{-1}\boldsymbol{\kappa}_{*(u)}\\&=\left(\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}-\mathbf{K}_{(u)}^{-1}\boldsymbol{\kappa}_{*(u)}\right)^{\text{ú}}\mathbf{K}_{(u)}\left(\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}-\mathbf{K}_{(u)}^{-1}\boldsymbol{\kappa}_{*(u)}\right).\end{aligned} \tag{28}$$

Considering that $\mathbf{K}_{(u)}$ is a positive-semidefinite matrix, it can be concluded that $\mathsf{E}\left[\Delta\mathsf{E}_D\right]\ge 0$. Hence, $\mathsf{E}\left[\mathsf{E}_D^u(x_*)\right]\le\mathsf{E}\left[\mathsf{E}_D^b(x_*)\right]$.

3) Definition of deformation

It can be noticed that the Bayesian generalization error in the unbiased case, which is a lower bound of the Bayesian generalization error in real applications, precisely equals the posterior variance, as shown in (4). Hence, a large posterior variance is related to a higher risk of deformation. Moreover, we also notice that the posterior variance $\kappa_{**}-\boldsymbol{\kappa}_*^{\text{ú}}\mathbf{K}^{-1}\boldsymbol{\kappa}_*$ can never exceed the prior variance κ_{**}. Therefore, the GPR is more likely to be deformed if the posterior variance is close to the prior variance.

Similar to over-fitting, it seems that we can only approach deformation of GPR as a qualitative notion. However, based on the findings above, a threshold-based quantitative definition can be provided.

Definition 3 (Deformation). *A GPR trained on dataset D is deformed with threshold δ_d when the following two conditions hold,*

(1) $\mathsf{E}_D(x_i)=0,\ \forall(x_i,y_i)\in D$.

(2) $\min\limits_{x_*\in\mathsf{X}}\left(\kappa_{**}-\mathsf{E}\left[\mathsf{E}_D(x_*)\right]\right)\le\delta_d$.

where $\mathsf{E}_D(\cdot)$ refers to the error corresponding to a biased kernel function, and X denotes the input space.

The first condition in Definition 3 ensures that the GPR is not under-fitting, as all training samples are perfectly fitted. The second condition limits the minimum gap δ_d between the posterior and prior variance. If a strict minimum gap $\delta_d=0$ is adopted, the GPR becomes completely deformed.

Definition 4 (Complete deformation). *A GPR trained on dataset D is completely deformed when it is deformed with threshold $\delta_d=0$.*

Since we can never know the exact covariances between each two input vectors, almost all kernels suffer the risk of deformation in GPR. Selecting proper values for hyperparameters is critical for GPR. As has been shown in Figure 2 and 3, the LML curve of GPR stays constant when the length scale of the kernel function is not adequately large. As a result, the model becomes deformed and over-fitted for a wide range of small length scale values. In addition, the gradient of the LML curve in this range corresponding to deformation is close to zero, making gradient-based optimization algorithms easily stuck in local optimum when the initial point of the algorithm locates here. Therefore, it is necessary to find a lower bound of length scale to avoid the appearance of deformation.

C. Lower bounds of length scale to avoid deformation

To jump over the region that corresponds with deformation, it is natural to seek for a lower bound of length scale values, which can help narrow the range where we place the initial point of gradient-based optimizers. The property of being compactly supported indicates the possibility that the use of hat kernel may lead to complete deformation of a GPR.

The hat kernel function is a function with a compact support, whose value will be constantly zero when the sample distance exceeds a specific threshold, and we can write its support, i.e., the influencing range of a sample, as follows,

$$\begin{aligned}\operatorname{supp}\left(\kappa_{\text{hat}}\left(x_0,\cdot\right)\right)&=\left\{x\mid\kappa_{\text{hat}}\left(x_0,x\right)>0\right\}\\&=\left\{x\mid\left|x-x_0\right|<l\right\}\end{aligned} \tag{29}$$

For each unobserved point of interest x_*, its posterior mean can be written as $\boldsymbol{\kappa}_*^{\text{ú}}\mathbf{K}^{-1}\mathbf{y}$ according to (4). Since $\mathbf{K}^{-1}\mathbf{y}$ is constant with respect to $\mathbf{x}_*$, we denote it by $\mathbf{a}$. This expression can be rewritten as follows,

$$\begin{aligned}m_{\text{post}}&=\boldsymbol{\kappa}_*^{\text{ú}}\mathbf{K}^{-1}\mathbf{y}=\boldsymbol{\kappa}_*^{\text{ú}}\mathbf{a}\\&=\sum_{i=1}^{n}a_i\kappa_{\text{hat}}\left(x_i,x_*\right).\end{aligned} \tag{30}$$

The equation above expresses the posterior mean as the weighted sum of kernel function values between the new point and every sample point.

Similarly, the second term of the posterior variance $\sigma_{\text{post}}^2 = \kappa_{**} - \boldsymbol{\kappa}_*^{\mathrm{T}}\mathbf{K}^{-1}\boldsymbol{\kappa}_*$ is a quadratic form. It can be rewritten as follows,

$$\boldsymbol{\kappa}_*^{\mathrm{T}}\mathbf{K}^{-1}\boldsymbol{\kappa}_* = \text{poly}_n\left(\kappa_{\text{hat}}\left(x_1, x_*\right), \kappa_{\text{hat}}\left(x_2, x_*\right), \ldots, \kappa_{\text{hat}}\left(x_n, x_*\right)\right) \tag{31}$$

where $\text{poly}_n(\cdot)$ denotes a n-order polynomial function. Thus, the posterior variance at x_* will equal the prior variance when all kernel function values equal zero.

Without loss of generality, we assume all training samples are sorted in ascending order, i.e., $x_1 < x_2 < \ldots < x_n$, and we limit the input space of our interest to $\mathsf{X} = [x_1, x_n]$. To avoid the appearance of complete deformation, we should ensure that, for all $x_* \in [x_1, x_n] \setminus \mathbf{x}$, $\max_i \kappa_{\text{hat}}\left(x_i, x_*\right) > 0$.

Proposition 3. *For all* $x_* \in [x_1, x_n] \setminus \mathbf{x}$, $l > \sup_i \min_j \left|x_i - x_j\right|/2$ *is a sufficient condition of* $\max_i \kappa_{\text{hat}}\left(x_i, x_*\right) > 0$.

Proof. Given an arbitrary point $x_* \in [x_1, x_n] \setminus \mathbf{x}$, find its nearest sample point

$$x_{*,\text{nr}} = \arg\min_{x_i \in \mathbf{x}} \left|x_i - x_*\right|. \tag{32}$$

And their distance is $d_{*,\text{nr}} = \left|x_{*,\text{nr}} - x_*\right|$. Due to the monotonicity of the hat kernel function with respect to sample distance, for any point $x \in \mathbf{x} \setminus \left\{x_{*,\text{nr}}\right\}$, we have

$$\kappa_{\text{hat}}\left(x, x_*\right) \le \kappa_{\text{hat}}\left(x_{*,\text{nr}}, x_*\right). \tag{33}$$

Additionally, $d_{*,\text{nr}} \le \max_{i \in [1, n-1]}\left(x_{i+1} - x_i\right)$, which can be generalized to an unsorted $\mathbf{x}$, i.e., $d_{*,\text{nr}} \le \sup_i \min_j \left|x_i - x_j\right|/2$.

According to the inequality in (13), $\forall x_* \in [x_1, x_n] \setminus \mathbf{x}, \max_i \kappa_{\text{hat}}\left(x_i, x_*\right) > 0$ is equivalent to $\forall x_* \in [x_1, x_n] \setminus \mathbf{x}, \kappa_{\text{hat}}\left(x_{*,\text{nr}}, x_*\right) > 0$. This requires that $d_{*,\text{nr}} < l$, which means that $l > \sup_i \min_j \left|x_i - x_j\right|/2$. □

Although vector $\mathbf{a}$ is constant with respect to $\mathbf{x}_*$, it is associated with the length scale of the kernel function. Consequently, the posterior estimate of a specific point can be influenced by sample points far away. In the case of deformation other than complete deformation, each sample point has limited influencing range, and the posterior mean curve can still show jerks near some of sample points. A stronger condition can be adopted to reduce the risk of deformation; that is, for all $x_* \in [x_1, x_n] \setminus \mathbf{x}$, its posterior mean $m_{\text{post}}\left(x_*\right)$ can be affected by the label of any sample point.

The crucial point of this condition lies in vector $\mathbf{a}$, since the value of the hat kernel function between two points with a larger distance than l will be zero, and requiring that $l > x_n - x_1$ is apparently unreasonable. Considering that $\mathbf{y}$ is a constant vector, we focus on the inverse of kernel matrix $\mathbf{K}^{-1}$. A comparison between two hat kernel matrices with different length scale values is demonstrated in Figure 7, where both ordinate and abscissa represent the index of the sample point, and a bluer colour in the figure indicates a larger covariance between two sample points. When the length scale value is small, e.g., Figure 7 (a), the kernel matrix is block diagonal and can be written in the following form,

$$\begin{aligned}\mathbf{K} &= \begin{bmatrix} \kappa_{\text{hat}}\left(x_1, x_1\right) & \kappa_{\text{hat}}\left(x_1, x_2\right) & \cdots & \kappa_{\text{hat}}\left(x_1, x_n\right) \\ \kappa_{\text{hat}}\left(x_2, x_1\right) & \kappa_{\text{hat}}\left(x_2, x_2\right) & \cdots & \kappa_{\text{hat}}\left(x_2, x_n\right) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa_{\text{hat}}\left(x_n, x_1\right) & \kappa_{\text{hat}}\left(x_n, x_2\right) & \cdots & \kappa_{\text{hat}}\left(x_n, x_n\right) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{K}_1 & 0 & \cdots \\ 0 & \mathbf{K}_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}. \end{aligned} \tag{34}$$

To compute $\mathbf{K}^{-1}$ in vector $\mathbf{a}$, one may compute the inverse of each sub-matrix,

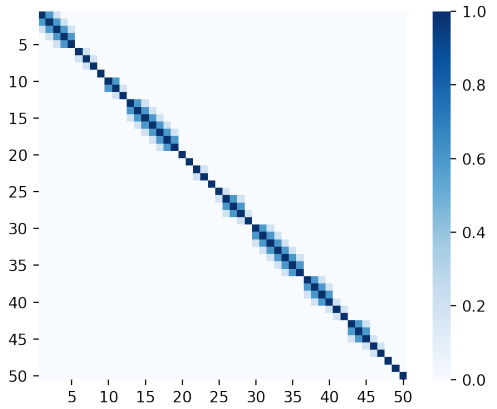
$$\mathbf{K}^{-1} = \begin{bmatrix} \mathbf{K}_1^{-1} & 0 & \cdots \\ 0 & \mathbf{K}_2^{-1} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}. \tag{35}$$

Suppose the shape of $\mathbf{K}_1$ is $n_1 \times n_1$, label $\{y_i \mid i \le n_1\}$ will have no influence on the posterior estimate of points $\left\{x \mid x > x_{n_1} + l\right\}$. Therefore, to ensure that the posterior mean $m_{\text{post}}\left(x_*\right), \forall x_* \in [x_1, x_n] \setminus \mathbf{x}$ can be affected by the label of any sample point, we have to ensure that the kernel matrix $\mathbf{K}$ is not block diagonal.

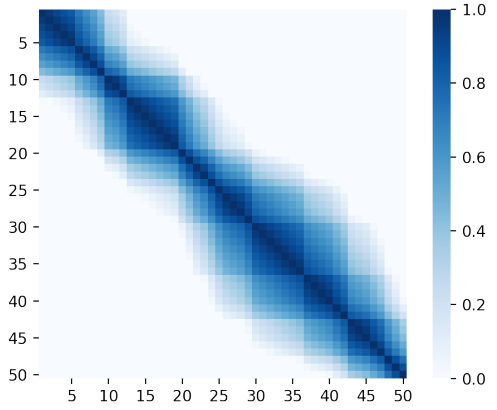
Proposition 4. $l > \sup_i \min_j \left|x_i - x_j\right|$ *is a sufficient condition of the kernel matrix* $\mathbf{K}$ *being not block diagonal.*

Proof. The kernel matrix $\mathbf{K}$ is block diagonal means that there exists $s \in [1, n-1]$ such that, for all $i \in [1, s]$ and $j \in [s+1, n]$, $\kappa_{\text{hat}}\left(x_i, x_j\right) = 0$. This is equivalent to $l > x_{s+1} - x_s$.

Therefore, one sufficient condition of the kernel matrix $\mathbf{K}$ being not block diagonal is that $l > \max_{s \in [1, n-1]}\left(x_{s+1} - x_s\right)$, which can be generalized to an unsorted $\mathbf{x}$, i.e., $l > \sup_i \min_j \left|x_i - x_j\right|$. □



(a): $l = 1$



(b): $l = 10$

Figure 7: Hat kernel matrices with different length scales.

V. Conclusion

Gaussian process regression is an emerging and efficient model in transportation system estimation and prediction problems. This paper investigates the hyperparameter optimization problem of GPR. We focus on monotonic kernel functions widely used in a variety of common tasks, such as the hyperparameter optimization of deep neural networks and simulation-based optimization. The most frequently adopted monotonic kernel function is the RBF kernel, which is infinitely differentiable and can yield smooth estimation results. However, one issue often neglected in practical applications is the over-fitting and under-fitting of GPR due to inappropriate hyperparameter of the RBF kernel. The hyperparameter of the kernel function in GPR is often determined by optimizing the nonconvex LML function, where plateaus can be observed in the function curve of LML when the hyperparameter is either too small or too large. A question arises that whether we can find a kernel function similar to the RBF kernel that can alleviate the risk of causing the issue above at the same time. Towards this question, our study suggests the use of the hat kernel instead of the RBF function. It is found that the hat kernel can avoid under-fitting of GPR due to overly large hyperparameter, but the over-fitting issue due to overly small hyperparameter is still unattended.

To further address the over-fitting issue, a lower bound analysis on the hyperparameter of the hat kernel is performed. Graphical interpretations on the LML are first presented to help understanding the objective of the hyperparameter optimization problem of GPR. Then, we show that a too small hyperparameter of the hat kernel leads to the deformation of the posterior mean curve. Two lower bounds are derived, which are hopeful to reduce the risk that the hyperparameter optimization is stuck in the local optimum.

The proposed hat kernel as well as the lower bounds of the hyperparameter overcome the overfitting/underfitting issues of GPR. As an alternative/improvement of the RBF kernel, the hat kernel is of considerable significance in that it helps to further improve the use of GPR in solving transportation problems. For future work, it is worth studying how the hat kernel can be generalized to input vectors with higher feature dimensions and whether a stronger bound can be found for the kernel function. This study has not employed the hat kernel to address transportation problems. Applying GPR with hat kernel to solve traffic volume prediction problems with noisy data or high-dimensional transportation optimization problems is suggested as a meaningful extension of this study.

Acknowledgments

This study is supported by the Distinguished Young Scholar Project (No. 71922007) and Key Project (No. 52131203) of the National Natural Science Foundation of China.

Reference

[1] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," *Transportation Research Part C: Emerging Technologies,* vol. 79, pp. 1-17, 2017.

[2] R. Yan, S. Wang, and K. Fagerholt, "A semi-“smart predict then optimize”(semi-SPO) method for efficient ship inspection," *Transportation Research Part B: Methodological,* vol. 142, pp. 100-125, 2020.

[3] Z. Liu, Y. Liu, C. Lyu, and J. Ye, "Building personalized transportation model for online taxi-hailing demand prediction," *IEEE Transactions on Cybernetics,* 2020.

[4] Y. Liu, C. Lyu, Y. Zhang, Z. Liu, W. Yu, and X. Qu, "DeepTSP: Deep traffic state prediction model based on large-scale empirical data," *Communications in Transportation Research,* vol. 1, p. 100012, 2021.

[5] Z. Liu, Y. Liu, Q. Meng, and Q. Cheng, "A tailored machine learning approach for urban transport network flow estimation," *Transportation Research Part C: Emerging Technologies,* vol. 108, pp. 130-150, 2019.

[6] H. Zhang, Y. Wu, H. Tan, H. Dong, F. Ding, and B. Ran, "Understanding and modeling urban mobility dynamics via disentangled representation learning," *IEEE Transactions on Intelligent Transportation Systems,* 2020.

[7] C. Lyu, X. Wu, Y. Liu, and Z. Liu, "A Partial-Fréchet-Distance-Based Framework for Bus Route Identification," *IEEE Transactions on Intelligent Transportation Systems,* 2021.

[8] Q. Cheng, Z. Liu, Y. Lin, and X. S. Zhou, "An s-shaped three-parameter (S3) traffic stream model with consistent car following relationship," *Transportation Research Part B: Methodological,* vol. 153, pp. 246-271, 2021.

[9] S. Wang, X. Chen, and X. Qu, "Model on empirically calibrating stochastic traffic flow fundamental diagram," *Communications in Transportation Research,* vol. 1, p. 100015, 2021.

[10] J. Bao, P. Liu, and S. V. Ukkusuri, "A spatiotemporal deep learning approach for citywide short-term crash risk prediction with multi-source data," *Accident Analysis & Prevention,* vol. 122, pp. 239-254, 2019.

[11] B. Sifringer, V. Lurkin, and A. Alahi, "Enhancing discrete choice models with representation learning," *Transportation Research Part B: Methodological,* vol. 140, pp. 236-261, 2020.

[12] J. Huo, X. Fu, Z. Liu, and Q. Zhang, "Short-Term Estimation and Prediction of Pedestrian Density in Urban Hot Spots Based on Mobile Phone Data," *IEEE Transactions on Intelligent Transportation Systems,* 2021.

[13] D. Huang, Z. Liu, P. Liu, and J. Chen, "Optimal transit fare and service frequency of a nonlinear origin-destination based fare structure," *Transportation Research Part E: Logistics and Transportation Review,* vol. 96, pp. 1-19, 2016.

[14] J. Zhao and S. Sun, "High-order Gaussian process dynamical models for traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems,* vol. 17, no. 7, pp. 2014-2019, 2016.

[15] M. Liang, X. Huang, C.-H. Chen, X. Chen, and A. Tokuta, "Counting and classification of highway vehicles by regression analysis," *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, no. 5, pp. 2878-2888, 2015.

[16] C. Ramussen and C. Williams, "Gaussian processes for machine learning (adaptive computation and machine learning)," ed: The MIT Press, Boston, 2006.

[17] A. Garriga-Alonso, C. E. Rasmussen, and L. Aitchison, "Deep convolutional networks as shallow gaussian processes," *arXiv preprint arXiv:1808.05587,* 2018.

[18] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as gaussian processes," *arXiv preprint arXiv:1711.00165,* 2017.

[19] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012.

[20] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems,* vol. 25, 2012.

[21] Y. Xie, K. Zhao, Y. Sun, and D. Chen, "Gaussian processes for short-term traffic volume forecasting," *Transportation Research Record,* vol. 2165, no. 1, pp. 69-78, 2010.

[22] C. K. Williams and C. E. Rasmussen, "Gaussian processes for regression," 1996.

[23] Y. Yuan, Z. Zhang, X. T. Yang, and S. Zhe, "Macroscopic traffic flow modeling with physics regularized Gaussian process: A new insight into machine learning applications in transportation," *Transportation Research Part B: Methodological,* vol. 146, pp. 88-110, 2021.

[24] S. Zhong, Y. Gong, Z. Zhou, R. Cheng, and F. Xiao, "Active learning for multi-objective optimal road congestion pricing considering negative land use effect," *Transportation Research Part C: Emerging Technologies,* vol. 125, p. 103002, 2021.

[25] A. Aboudina and B. Abdulhai, "A bi-level distributed approach for optimizing time-dependent congestion pricing in large networks: A simulation-based case study in the Greater Toronto Area," *Transportation Research Part C: Emerging Technologies,* vol. 85, pp. 684-710, 2017.

[26] D. Huang, J. Xing, Z. Liu, and Q. An, "A multi-stage stochastic optimization approach to the stop-skipping and bus lane reservation schemes," *Transportmetrica A: Transport Science,* vol. 17, no. 4, pp. 1272-1304, 2021.

[27] Z. Liu, Z. Wang, Q. Cheng, R. Yin, and M. Wang, "Estimation of urban network capacity with second-best constraints for multimodal transport systems," *Transportation research part B: methodological,* vol. 152, pp. 276-294, 2021.

[28] M. Svensén and C. M. Bishop, "Pattern recognition and machine learning," ed: Springer, 2007.

[29] M. G. Genton, "Classes of kernels for machine learning: a statistics perspective," *Journal of machine learning research,* vol. 2, no. Dec, pp. 299-312, 2001.

[30] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proceedings of the 14th annual conference on neural information processing systems*, 2001, no. CONF, pp. 682-688.

[31] M. Titsias, "Variational learning of inducing variables in sparse Gaussian processes," in *Artificial intelligence and statistics*, 2009: PMLR, pp. 567-574.

[32] J. M. Tomczak, "Gaussian process regression with categorical inputs for predicting the blood glucose level," in *International Conference on Systems Science*, 2016: Springer, pp. 98-108.

[33] Y. Liu, P. Bansal, R. Daziano, and S. Samaranayake, "A framework to integrate mode choice in the design of mobility-on-demand systems," *Transportation Research Part C: Emerging Technologies,* vol. 105, pp. 648-665, 2019.

[34] X. Ying, "An overview of overfitting and its solutions," in *Journal of Physics: Conference Series*, 2019, vol. 1168, no. 2: IOP Publishing, p. 022022.

[35] T. Dietterich, "Overfitting and undercomputing in machine learning," *ACM computing surveys (CSUR),* vol. 27, no. 3, pp. 326-327, 1995.

[36] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *the Journal of machine Learning research,* vol. 12, pp. 2825-2830, 2011.

[37] F. Vivarelli, "Studies on the generalisation of Gaussian processes and Bayesian neural networks," Aston University, 1998.

APPENDIX A

Proposition 5. The Bayesian generalization error using a biased kernel function is,

$$\mathsf{E}\left[\mathsf{E}_{\mathcal{D}}^{b}(x_*)\right] = \kappa_{**(u)} - 2\boldsymbol{\kappa}_{*(u)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)} + \boldsymbol{\kappa}_{*(b)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\mathbf{K}_{(u)}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}. \quad (36)$$

Proof.

$$\begin{aligned}\mathsf{E}\left[\mathsf{E}_{\mathcal{D}}^{b}(x_*)\right] &= \mathsf{E}\left[\left(\boldsymbol{\kappa}_{*(b)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\mathbf{y} - y_*\right)^2\right] \\ &= \mathsf{E}\left[\boldsymbol{\kappa}_{*(b)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\mathbf{y}\mathbf{y}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)} - 2\boldsymbol{\kappa}_{*(b)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\mathbf{y}y_* + y_*^2\right] \\ &= \mathsf{E}\left[\boldsymbol{\kappa}_{*(b)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\mathbf{y}\mathbf{y}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\right] - 2\mathsf{E}\left[\boldsymbol{\kappa}_{*(b)}^{\text{ú}}\mathbf{K}_{(b)}^{-1}\mathbf{y}y_*\right] \\ &\quad + \mathsf{E}\left[y_*^2\right].\end{aligned} \quad (37)$$

Concerning the first term, it can be rewritten as,

$$
\begin{aligned}
&\mathrm{E}\left[\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathbf{y}\mathbf{y}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\right]\\
&=\mathrm{E}\left[\mathrm{tr}\left(\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathbf{y}\mathbf{y}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\right)\right]\\
&=\mathrm{E}\left[\mathrm{tr}\left(\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathbf{y}\mathbf{y}^{\acute{u}}\right)\right]\\
&=\mathrm{tr}\left[\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathrm{E}\left(\mathbf{y}\mathbf{y}^{\acute{u}}\right)\right].
\end{aligned}
\tag{38}
$$

Here, the training labels follow the unbiased prior distribution $\mathbf{Y}\sim\mathrm{N}(\mathbf{0},\mathbf{K}_{(u)})$. Therefore, $\mathrm{E}\left(\mathbf{y}\mathbf{y}^{\acute{u}}\right)=\mathbf{K}_{(u)}$, and (38) can be further simplified as,

$$
\begin{aligned}
&\mathrm{E}\left[\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathbf{y}\mathbf{y}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\right]\\
&=\mathrm{tr}\left[\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathrm{E}\left(\mathbf{y}\mathbf{y}^{\acute{u}}\right)\right]\\
&=\mathrm{tr}\left[\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathbf{K}_{(u)}\right]\\
&=\mathrm{tr}\left[\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathbf{K}_{(u)}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\right]\\
&=\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathbf{K}_{(u)}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}.
\end{aligned}
\tag{39}
$$

Concerning the second term, it can be rewritten as,

$$
\mathrm{E}\left[\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathbf{y}y_{*}\right]=\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathrm{E}\left[\mathbf{y}y_{*}\right]
\tag{40}
$$

The joint prior distribution of the training labels and the new sample label uses the biased kernel function, so we have $\mathrm{E}\left[\mathbf{y}y_{*}\right]=\boldsymbol{\kappa}_{*(b)}$. Then, (40) can be further simplified,

$$
\begin{aligned}
\mathrm{E}\left[\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathbf{y}y_{*}\right]&=\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathrm{E}\left[\mathbf{y}y_{*}\right]\\
&=\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}.
\end{aligned}
\tag{41}
$$

Concerning the third term, similar to (22), it still equals $\kappa_{**(u)}$.

Combine all three terms together, the Bayesian generalization error is,

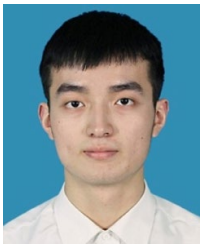
$$
\begin{aligned}
\mathrm{E}\left[\mathrm{E}_{D}^{b}(x_{*})\right]&=\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\mathbf{K}_{(u)}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}\\
&\quad-2\boldsymbol{\kappa}_{*(b)}^{\acute{u}}\mathbf{K}_{(b)}^{-1}\boldsymbol{\kappa}_{*(b)}+\kappa_{**(u)}
\end{aligned}
\tag{42}
$$

APPENDIX B

Table I Notation Table

Notations	
$\mathbf{x}$	The input vector
y	The output label
$\mathbf{x}_{*}$	An arbitrary input vector
$\kappa(\cdot,\cdot)$	The kernel function
l	The hyperparameter of the kernel
δ_{d}	the minimum gap between the posterior and prior variance.
D	Dataset contains n samples $D=(\mathbf{x}_{1:n},\mathbf{y}_{1:n})$, where $\mathbf{x}_{1:n}=[\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n]^{\acute{u}}$, $\mathbf{y}_{1:n}=[y_1,y_2,\ldots,y_n]^{\acute{u}}$
$\mathbf{K}$	The covariance matrix, whose element on the i th row and j th column is $\kappa(\mathbf{x}_i,\mathbf{x}_j)$
$\boldsymbol{\kappa}_{*}$	The vector denotes covariances between $\mathbf{x}_{*}$ an $\mathbf{x}$, $\boldsymbol{\kappa}_{*}=[\kappa(\mathbf{x}_1,\mathbf{x}_{*}),\kappa(\mathbf{x}_2,\mathbf{x}_{*}),\ldots,\kappa(\mathbf{x}_n,\mathbf{x}_{*})]^{\acute{u}}$

Zhiyuan Liu received his Ph.D. degree in Transportation Engineering in 2011 from National University of Singapore. He is currently a professor at School of Transportation in Southeast University, and director of the Research Center for Complex Transport Networks.
His research interests include transport network modelling, public transport, and intelligent transport system. In these areas, Dr. Liu has published over 70 journal papers.



Cheng Lyu received his B.S. and M.S. degree in Transportation Engineering in the School of Transportation at Southeast University, Nanjing, China. His research interests include transportation big data analysis and modelling, machine learning, data mining and intelligent transportation systems.

Jinbiao Huo received the B.S. degree in transportation engineering from the School of Transportation, Southeast University, Nanjing, China. He is currently pursuing the M.S. degree in transportation engineering with the School of Transportation, Southeast University.
His research interests include big data analytics and transport network modelling.

Shuaian (Hans) Wang is a Professor at The Hong Kong Polytechnic University (PolyU). His research interests include shipping operations management, green shipping, big data in shipping, port planning and operations, urban transport network modeling, and logistics and supply chain management. He dedicates to rethinking and proposing innovative solutions to improve the efficiency of maritime and urban transportation systems, to promote environmental friendly and sustainable practices, and to transform business and engineering education.

Jun Chen is a professor at School of Transportation in Southeast University, and also the dean of School of Transportation in Southeast University. His research interests include transportation planning, transportation management, public transportation and parking management.