Lecture 10: Gaussian Process Regression (GPR) II

Instructor: Wenxiao Pan

In regression, our objective is to find a function (or model) f based on observed data points (the training dataset). Traditional nonlinear regression methods often give a single function that is considered to best fit the dataset. However, there could be more than one function that fits the observed data points equally well.

As shown in Fig. 6 of the prior lecture, we can generate infinite many samples to represent infinite numbers of functions. When we start to have observations, instead of infinite numbers of functions, we only keep functions that fit the observed data points, forming the posterior distribution. This posterior is the prior updated with observed data. When we have new observations, we use the current posterior as prior, and new observed data points to obtain a newer posterior.

Gaussian Process Model

A Gaussian process model describes a probability distribution over possible functions that fit a set of data points. Because we have the probability distribution over all possible functions, we can compute its mean to represent the maximum likelihood estimate of the function, and its variance as an indicator of prediction confidence. The key points can be summarized as:

1. A Gaussian process model is a probability distribution over possible functions;
2. The mean function derived from the posterior distribution of possible functions *is* the function used for regression predictions;
3. The function prior is updated with new observations.

Denote $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ the observed data points and $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]$ the true function values measured at the data points. $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ with k a kernel function (e.g., RBF), from which the assembled matrix $\mathbf{K}$ is positive definite. The Gaussian process model is thus a distribution over functions whose shapes (smoothness) are defined by $\mathbf{K}$.

The regression using Gaussian processes is illustrated in Fig. 1: given observed data (red points), we estimate the mean function (blue line) of the posterior distribution of possible functions, and then predict at new points $\mathbf{X}^*$ as $\mathbf{f}^*$; the variance of the posterior distribution of possible functions (blue shaded region) indicates the prediction confidence.

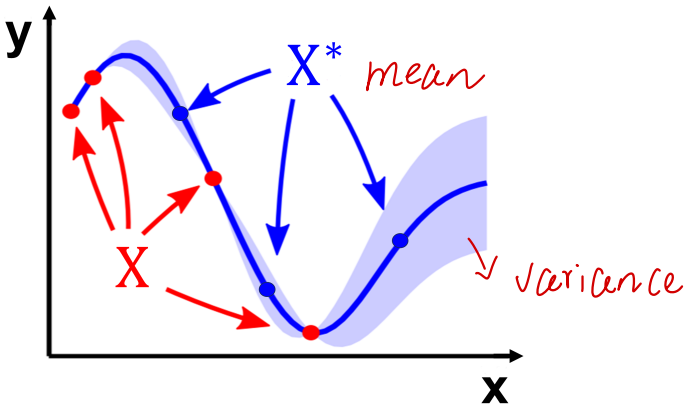


Figure 1: A illustrative process of Gaussian process regression. The red points are observed data, the blue line represents the mean function estimated by the observed data points, and predictions will be made at new blue points.

The joint distribution of $\mathbf{f}$ and $\mathbf{f}^*$ is expressed as:

$$\begin{bmatrix}\mathbf{f}\\ \mathbf{f}^*\end{bmatrix} \sim P(\mathbf{f}, \mathbf{f}^* \,|\, \mathbf{X}, \mathbf{X}^*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{K} & \mathbf{K}^*\\ \mathbf{K}^{*\mathsf{T}} & \mathbf{K}^{**}\end{bmatrix}\right), \tag{1}$$

where the notation "$\sim$" represents "follows and can be sampled from the probability distribution"; the mean $\boldsymbol{\mu} = [m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n)] = \mathbf{0}$; and the co-variances $\mathbf{K} = k(\mathbf{X}, \mathbf{X})$, $\mathbf{K}^* = k(\mathbf{X}, \mathbf{X}^*)$ and $\mathbf{K}^{**} = k(\mathbf{X}^*, \mathbf{X}^*)$.

While this equation describes the joint probability distribution $P(\mathbf{f}, \mathbf{f}^* \,|\, \mathbf{X}, \mathbf{X}^*)$ over $\mathbf{f}$ and $\mathbf{f}^*$, in regressions, we actually need the conditional distribution $P(\mathbf{f}^* \,|\, \mathbf{f}, \mathbf{X}, \mathbf{X}^*)$ over $\mathbf{f}^*$ only. The derivation of the conditional distribution $P(\mathbf{f}^* \,|\, \mathbf{f}, \mathbf{X}, \mathbf{X}^*)$ from the joint distribution $P(\mathbf{f}, \mathbf{f}^* \,|\, \mathbf{X}, \mathbf{X}^*)$ is achieved by using the Marginal and conditional distributions of MVN theorem [1, Sec. 2.3.1]. The result is:

$$\mathbf{f}^* \,|\, \mathbf{f}, \mathbf{X}, \mathbf{X}^* \sim P(\mathbf{f}^* \,|\, \mathbf{f}, \mathbf{X}, \mathbf{X}^*) = \mathcal{N}\left(\mathbf{K}^{*\mathsf{T}}\mathbf{K}^{-1}\mathbf{f},\ \mathbf{K}^{**} - \mathbf{K}^{*\mathsf{T}}\mathbf{K}^{-1}\mathbf{K}^*\right). \tag{2}$$

In realistic scenarios, we typically have access only to **noisy measurements** of true function values, denoted as $\mathbf{y} = \mathbf{f} + \boldsymbol{\epsilon}$, where each element of $\boldsymbol{\epsilon}$ represents additive independent and identically distributed (i.i.d.) Gaussian noise with variance σ_n^2. Then we have $\mathrm{cov}(\mathbf{y}) = \mathrm{cov}(\mathbf{f} + \boldsymbol{\epsilon}) = \mathbf{K} + \sigma_n^2\mathbf{I}$. The joint distribution of $\mathbf{y}$ and $\mathbf{f}^*$ is then given by:

$$\begin{bmatrix}\mathbf{y}\\ \mathbf{f}^*\end{bmatrix} \sim P(\mathbf{y}, \mathbf{f}^* \,|\, \mathbf{X}, \mathbf{X}^*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{K} + \sigma_n^2\mathbf{I} & \mathbf{K}^*\\ \mathbf{K}^{*\mathsf{T}} & \mathbf{K}^{**}\end{bmatrix}\right). \tag{3}$$

By deriving the conditional distribution, we get the predictive equations for Gaussian process regression:

$$\mathbf{f}^* \,|\, \mathbf{X}, \mathbf{y}, \mathbf{X}^* \sim P(\mathbf{f}^* \,|\, \mathbf{y}, \mathbf{X}, \mathbf{X}^*) = \mathcal{N}\left(\mathrm{mean}(\mathbf{f}^*), \mathrm{cov}(\mathbf{f}^*)\right), \tag{4}$$

where

$$\begin{aligned}\mathrm{mean}(\mathbf{f}^*) &= \mathbf{K}^{*\mathsf{T}}[\mathbf{K} + \sigma_n^2\mathbf{I}]^{-1}\mathbf{y},\\ \mathrm{cov}(\mathbf{f}^*) &= \mathbf{K}^{**} - \mathbf{K}^{*\mathsf{T}}[\mathbf{K} + \sigma_n^2\mathbf{I}]^{-1}\mathbf{K}^*.\end{aligned} \tag{5}$$

In Eq. (5), the mean function $\mathrm{mean}(\mathbf{f}^*)$ is the mean of possible functions and represents the maximum likelihood estimate of the function that can be used for regression predictions. The variance function $\mathrm{cov}(\mathbf{f}^*)$ indicates the prediction confidence and reveals that the uncertainty in predictions depends solely on the input values $\mathbf{X}$ and $\mathbf{X}^*$, not on the observed outputs $\mathbf{y}$. This characteristic is a distinctive property of Gaussian distributions [2].

From Eq. (5), we can follow these steps to determine the 95% confidence intervals:

1. Extract the variances from the diagonal elements of the covariance matrix (i.e., $\mathrm{cov}(\mathbf{f}^*)$ in Eq. (5));
2. Take the square root of the variances (extracted above) to get the standard deviations;
3. Determine the 95% confidence interval from the mean prediction and the standard deviation as: [mean_prediction $-1.96\times$ std_prediction, mean_prediction $+1.96\times$ std_prediction].

Kernel Selection

The selection of the kernel function is critical, as it greatly affects the model's ability to generalize [3]. Kernel functions range from well-established options like the RBF to custom designs tailored to specific needs based on model requirements such as smoothness, sparsity, drastic changes, and differentiability [4]. Selecting an appropriate kernel function for a specific GPR task is detailed in Duvenaud (2014) [4].

For demonstration purposes, consider a linear kernel, one of the simplest kernel functions, parameterized by a variance (σ^2) parameter. The construction of the kernel is as follows:

$$k^{\mathrm{Linear}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 + \mathbf{x}_i^{\mathsf{T}}\mathbf{x}_j, \tag{6}$$

which is called a homogeneous linear kernel when $\sigma^2 = 0$. For comparison, we also consider the widely used RBF kernel in its general form:

$$k^{\mathrm{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \exp\left(-\frac{1}{2l}(\mathbf{x}_i - \mathbf{x}_j)^{\mathsf{T}}(\mathbf{x}_i - \mathbf{x}_j)\right), \tag{7}$$

where σ and l are hyperparameters. Fig. 2 illustrates the effect of the kernel. It's clear that the linear kernel predicts a purely linear relation between the input and the output, which gives an underfitted model. The RBF kernel interpolates the data points quite well, but isn't very good at extrapolation (i.e., predicting unseen data points). As we can see, by trying to pass through as many data points as possible, the RBF kernel is fitted to the noise, creating an overfitted model. The combination of linear and RBF kernel, in comparison, has the best balance between interpolation and extrapolation—it interpolates the data points well enough, while at the same time extrapolating unseen data points at the two ends reasonably.

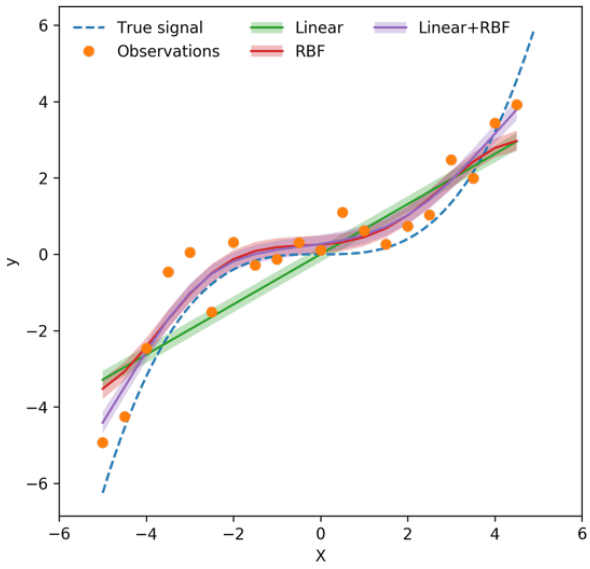


Figure 2: Kernel selection. The green, red, and purple lines demonstrate the predictions made by GPR when using the Linear, RBF, and summation of the linear and RBF kernels, respectively. The shaded region around each curve represents the 95-percent confidence interval associated with each prediction.

Hyperparameters Optimization

Additionally, hyperparameter optimization plays an essential role in kernel-based methods. For example, consider the widely used RBF kernel in its general form:

$$k^{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \exp\left(-\frac{1}{2l}(\mathbf{x}_i - \mathbf{x}_j)^\top(\mathbf{x}_i - \mathbf{x}_j)\right), \tag{8}$$

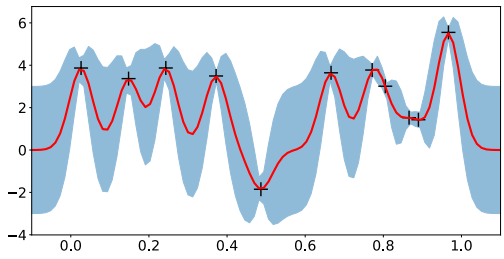
The variance parameter σ^2 controls the spread of the distribution; and l indicates the rate at which the correlation between two points decreases with increasing distance. The influence of the hyperparameter l on the smoothness of the function is demonstrated in Fig. 3. Increasing the value of l results in a smoother function, while a smaller l value leads to a function with more 'wiggles' or more overfitted to the noise. Likewise, a smaller variance results in a smoother curve, whereas a larger variance results in a more overfitted model.

The optimal hyperparameters $\boldsymbol{\Theta}^*$ can be determined by maximizing the log marginal likelihood [2] as:

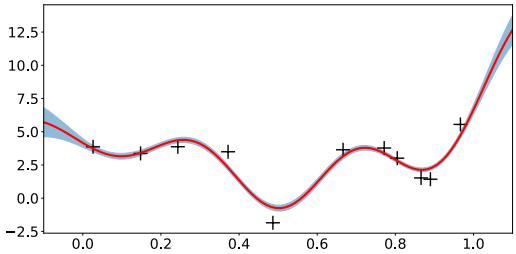
$$\boldsymbol{\Theta}^* = \arg\max_{\Theta} \left[\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\Theta})\right], \tag{9}$$

where the log marginal likelihood is defined as: $\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\Theta}) = -\frac{1}{2}\mathbf{y}^\top(\mathbf{K}+\sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log\det(\mathbf{K}+\sigma_n^2\mathbf{I})$. Marginal likelihood is the probability of the data given a probabilistic model and its parameters. By maximizing it, we can estimate the unknown parameters that are the most likely to have generated the observed sample data. Note that after optimizing the hyperparameters, the prediction variance $\text{cov}(\mathbf{f}^*)$ depends on not only the inputs $\mathbf{X}$ and $\mathbf{X}^*$ but also the outputs $\mathbf{y}$ [5].

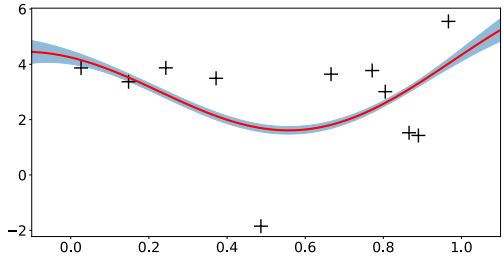
With the optimized hyperparameters, $\sigma = 0.0067$ and $l = 0.0967$, the regression result of the observed data points shown in Fig. 3 is depicted in Fig. 4.



(a) l = small



(b) l = medium



(c) l = large

Figure 3: Effect of the hyperparameter l: A larger l yields a smoother function, while a smaller l produces a more wiggly function.

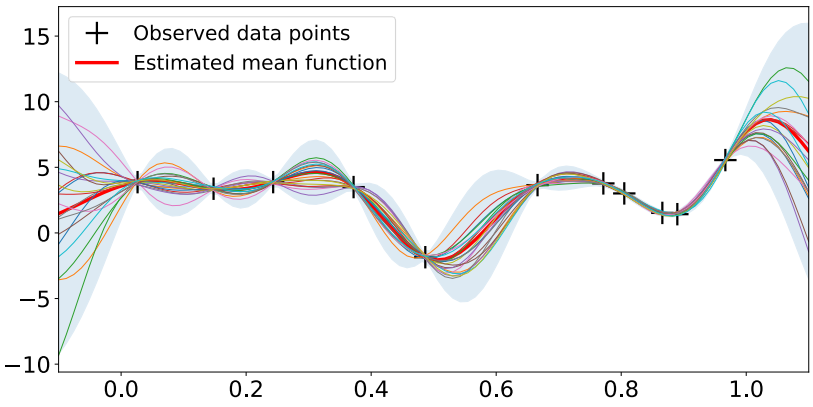


Figure 4: Regression result with the optimized hyperparameters σ and l.

Non-parametric Model

This section explains the distinction between parametric and non-parametric models [6].

Parametric models assume that the data can be modeled (or fitted) in terms of a set of finite numbers of parameters. In regression, given some data points, we would like to predict the function value $y^* = f(x^*)$ for a new specific x^*. If we assume a linear regression model, $y = \theta_1 + \theta_2 x$, we need to identify the parameters θ_1 and θ_2 to define the function. Often, a linear model is insufficient, and a polynomial model with more parameters, like $y = \theta_1 + \theta_2 x + \theta_3 x^2$ is needed. We use the training dataset comprising n observed points, $[(x_i, y_i) \,|\, i = 1, \ldots, n]$ to train (or fit) the model, i.e. establish a mapping x to y through some basis functions (e.g. polynomials, or activation function in DNN). After the fitting (or training) process, all information in the dataset is assumed to be captured by the model parameters $\boldsymbol{\theta}$, and then the predictions are **independent** of the training dataset. Thus, when conducting regressions using parametric models, the complexity or flexibility of models is inherently limited by **the number of parameters**.

Conversely, the structure of a **non-parametric model** is not specified a priori by a set of parameters, but is instead determined from data. The term non-parametric is not meant to imply that such models completely lack parameters but that the number and nature of the parameters are flexible and not fixed in advance.

Discussion

A Gaussian process is a probability distribution over possible functions that fit a set of points [2]. A Gaussian process regression model provides prediction values together with uncertainty estimates. The model incorporates prior knowledge about the nature of the functions through the use of kernel functions.

The GPR model discussed here is the standard or "vanilla" approach to Gaussian processes [7]. There are two primary limitations with it:

- The computational complexity is $O(N^3)$, where N represents the dimension of the covariance matrix K.
- The memory consumption increases quadratically with data size.

Due to these limitations, standard GPR models become impractical for large datasets. In such cases, sparse Gaussian Processes are employed to alleviate computational complexity [8].

Gaussian Process Regression Packages

Three Python packages for implementing Gaussian process regression:

1. GPy is a mature and well-documented package in development since 2012 [9]. It utilizes NumPy for computations, offering sufficient stability for tasks that are not computationally intensive. The hyperparameters' optimization can be conducted in the package.
2. For complex and computationally intense tasks, packages incorporating advanced algorithms and GPU acceleration are especially preferable. GPflow [9] originates from GPy with a similar interface. It leverages TensorFlow as its computational backend.
3. GPyTorch [10] is a more recent package that provides GPU acceleration through PyTorch. Like GPflow, GPyTorch supports automatic gradients, which simplifies the development of complex models, such as those embedding deep neural networks within GP frameworks.

In addition, https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpr_noisy_targets.html provides basic introductory examples for GPR with source code. GPML (https://gaussianprocess.org/gpml/code/matlab/doc/) is a package for Matlab.

Reference

[1] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006.

[2] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[3] D. Duvenaud, "The Kernel Cookbook," Available at https://www.cs.toronto.edu/~duvenaud/cookbook, 2016.

[4] ——, "Automatic model construction with Gaussian processes," Ph.D. dissertation, University of Cambridge, 2014.

[5] Z. Chen and B. Wang, "How priors of initial hyperparameters affect Gaussian process regression models," *Neurocomputing*, vol. 275, pp. 1702–1710, 2018.

[6] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[7] R. Frigola, F. Lindsten, T. B. Schön, and C. E. Rasmussen, "Bayesian Inference and Learning in Gaussian Process State-Space Models with Particle MCMC," in *Advances in Neural Information Processing Systems*, 2013, pp. 3156–3164.

[8] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gaussian process meets big data: A review of scalable GPs," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[9] A. G. De G. Matthews, M. Van Der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman, "GPflow: A Gaussian process library using TensorFlow," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1299–1304, 2017.

[10] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, "GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration," in *Advances in Neural Information Processing Systems*, 2018.