



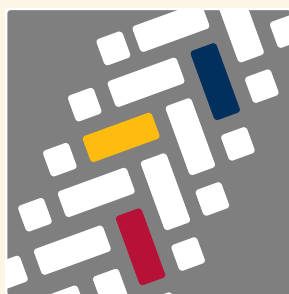
An Intuitive Tutorial to Gaussian Processes Regression

Jie Wang

Ingenuity Labs Research Institute

jie.wang@queensu.ca

February 3, 2021



Offroad Robotics

c/o Ingenuity Labs Research Institute

Queen's University

Kingston, ON K7L 3N6 Canada

Abstract

This tutorial aims to provide an intuitive understanding of the Gaussian processes regression. Gaussian processes regression (GPR) models have been widely used in machine learning applications because of their representation flexibility and inherently uncertainty measures over predictions. The basic concepts that a Gaussian process is built on, including multivariate normal distribution, kernels, non-parametric models, joint and conditional probability were explained first. Next, the GPR was described concisely together with an implementation of a standard GPR algorithm. Beyond the standard GPR, packages to implement state-of-the-art Gaussian processes algorithms were reviewed. This tutorial was written in an accessible way to make sure readers without a machine learning background can obtain a good understanding of the GPR basics.

Contents

1	Introduction	1
2	Mathematical Basics	1
2.1	Gaussian Distribution	1
2.2	Multivariate Normal Distribution	4
2.3	Kernels	5
2.4	Non-parametric Model	8
3	Gaussian Processes	9
4	Illustrative example	11
4.1	Hyperparameters Optimization	12
4.2	Gaussian processes packages	14
5	Summary and Discussion	15
A	Appendix	16

1 Introduction

The Gaussian processes model is a probabilistic supervised machine learning framework that has been widely used for regression and classification tasks. A Gaussian processes regression (GPR) model can make predictions incorporating prior knowledge (kernels) and provide uncertainty measures over predictions [11]. Gaussian processes model is a supervised learning method developed by computer science and statistics communities. Researchers with engineering backgrounds often find it is difficult to gain a clear understanding of it. To understand GPR, even only the basics needs to have knowledge of multivariate normal distribution, kernels, non-parametric model, and joint and conditional probability.

In this tutorial, we present a concise and accessible explanation of GPR. We first review the mathematical concepts that GPR models are built on to make sure readers have enough basic knowledge. In order to provide an intuitive understanding of GPR, plots are actively used. The codes developed to generate plots are provided at <https://github.com/jwangjie/Gaussian-Processes-Regression-Tutorial>.

2 Mathematical Basics

This section reviews the basic concepts needed to understand GPR. We start with the Gaussian (normal) distribution, then explain theories of multivariate normal distribution (MVN), kernels, non-parametric model, and joint and conditional probability. In regression, given some observed data points, we want to fit a function to represent these data points, then use the function to make predictions at new data points. For a given set of observed data points shown in Fig. 1(a), there are infinite numbers of possible functions that fit these data points. In Fig. 1(b), we show five sample functions that fit the data points. In GPR, the Gaussian processes conduct regression by defining a distribution over these infinite number of functions [8].

2.1 Gaussian Distribution

A random variable X is Gaussian or normally distributed with mean μ and variance σ^2 if its probability density function (PDF) is [10]

$$P_X(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right).$$

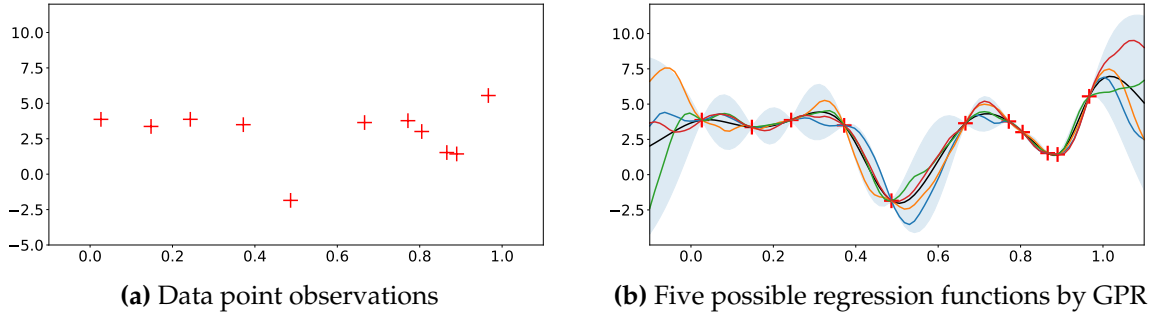


Figure 1: A regression example: (a) The observed data points, (b) Five sample functions that fit the observed data points.

Here, X represent random variables and x is the real argument. The normal distribution of X is usually represented by $P_X(x) \sim \mathcal{N}(\mu, \sigma^2)$. The PDF of a uni-variate normal (or Gaussian) distribution was plotted in Fig. 2. We randomly generated 1000 points from a uni-variate normal distribution and plotted them on the x axis.

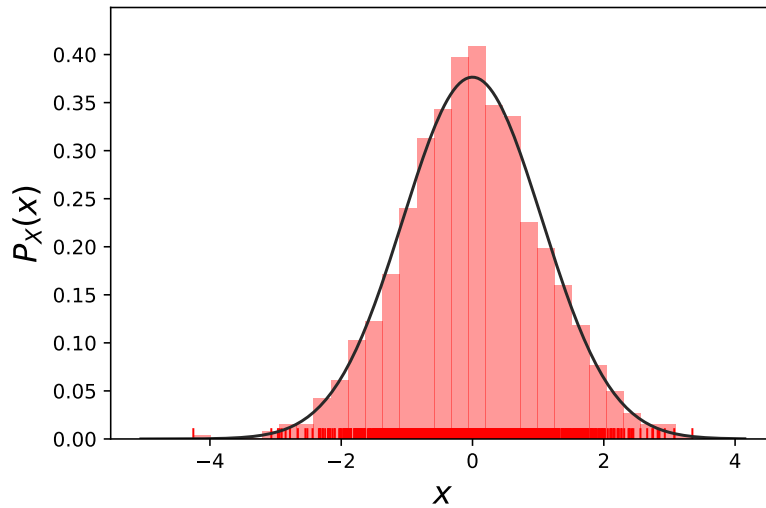


Figure 2: One thousand normal distributed data points were plotted as red vertical bars on the x axis. The PDF of these data points was plotted as a two-dimensional bell curve.

These random generated data points can be expressed as a vector $x_1 = [x_1^1, x_1^2, \dots, x_1^n]$. By plotting the vector x_1 on a new Y axis at $Y = 0$, we projected points $[x_1^1, x_1^2, \dots, x_1^n]$ into another space shown in Fig. 3. We did nothing but vertically plot points of the vector x_1 in a new Y, x coordinates space. We can plot another independent

Gaussian vector $x_2 = [x_2^1, x_2^2, \dots, x_2^n]$ in the same coordinates at $Y = 1$ shown in Fig. 3. Keep in mind that either x_1 or x_2 is a uni-variate normal distribution shown in Fig. 2.

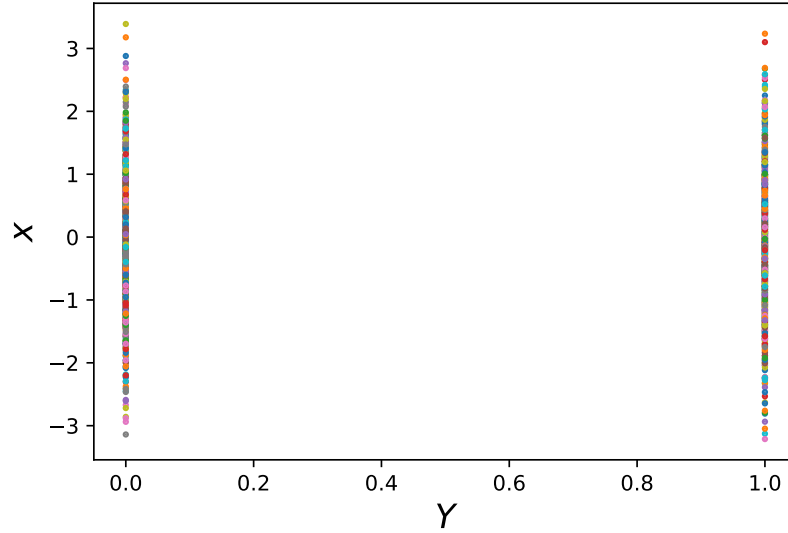


Figure 3: Two independent uni-variate Gaussian vector points were plotted vertically in the Y, x coordinates space.

Next, we randomly selected 10 points in vector x_1 and x_2 respectively and connected these 10 points in order by lines as shown in Fig. 4(a). These connected lines look like linear functions spanning within the $[0, 1]$ domain. We can use these functions to make predictions for regression tasks if the new data points are on (or close enough to) these linear lines. However, in most cases, the assumption that new data points are always on the connected linear functions is not held. If we plot more random generated uni-variate Gaussian vectors, for example, 20 vectors x_1, x_2, \dots, x_{20} in $[0, 1]$, and connect 10 random selected sample points of each vector as lines, we get 10 lines that look more like functions within $[0, 1]$ shown in Fig. 4(b). We still cannot use these lines to make predictions for regression tasks because they are too noisy. These functions must be smoother, meaning input points that are close to each other should have similar output values. The “functions” generated by connecting independent Gaussian vector points are not smooth enough for regression tasks, we need these independent Gaussian correlated to each other as a joint Gaussian distribution. The joint Gaussian distribution is described by the multivariate normal distribution theory.

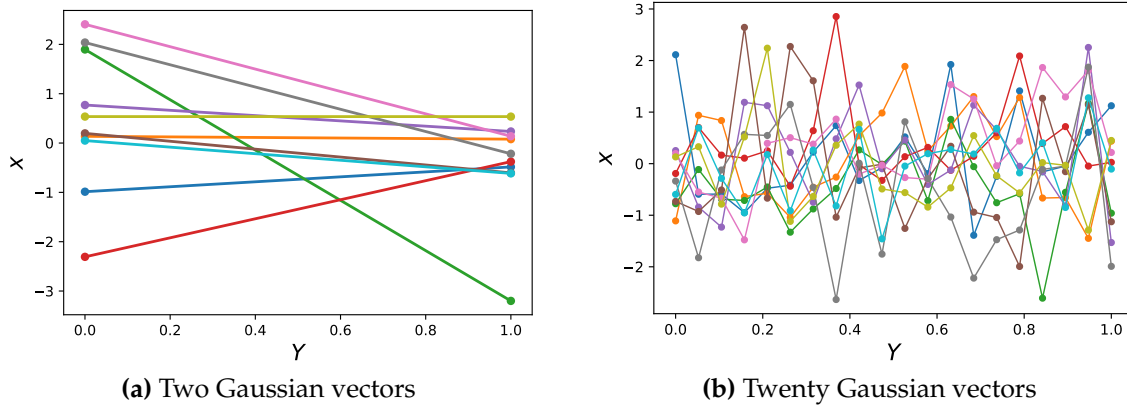


Figure 4: Connecting points of independent Gaussian vectors by lines: (a) Ten random selected points in two vector x_1 and x_2 , (b) Ten random selected points in twenty vectors x_1, x_2, \dots, x_{20} .

2.2 Multivariate Normal Distribution

It's common that a system is described by more than one feature variables (x_1, x_2, \dots, x_D) that are correlated to each other. If we want to model these variables all together as one Gaussian model, we need to use a multivariate Gaussian/normal (MVN) [10] distribution model. The PDF of an MVN with D dimension is defined as [10]

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp \left[-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right],$$

where D is the number of the dimension, x represents the variable, $\mu = \mathbb{E}[x] \in \mathbb{R}^D$ is the mean vector, and $\Sigma = \text{cov}[x]$ is the $D \times D$ covariance matrix. The Σ is a symmetric matrix that stores the pairwise covariance of all jointly modeled random variables with $\Sigma_{ij} = \text{cov}(y_i, y_j)$ as its (i, j) element.

We use a bi-variate normal (BVN) distribution as a simplified example to understand the MVN theory. A BVN distribution can be visualized as a three-dimensional (3-d) bell curve with heights represent the probability density shown in Fig. 5a. The projections of the 3-d bell curve on the x_1, x_2 plane are ellipse contours plotted in Fig. 5a and 5b. The shape of ellipses shows the correlations between x_1 and x_2 points, i.e. how much one variable of x_1 is related to another variable of x_2 . The $P(x_1, x_2)$ is the joint probability of x_1 and x_2 . For a BVN, the mean vector μ is a two-dimensional vector $\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$, where μ_1 and μ_2 are the independent mean of x_1

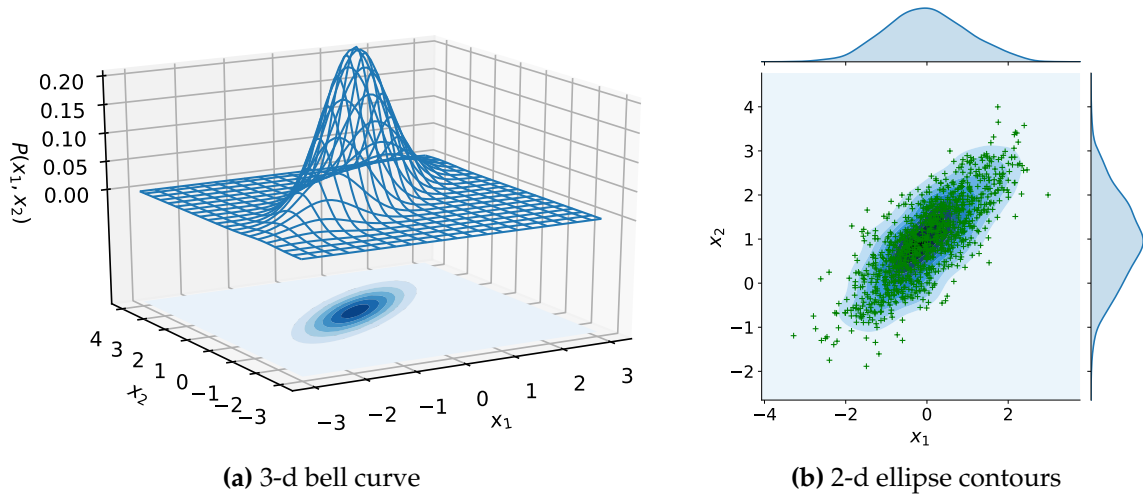


Figure 5: The PDF of a BVN visualization: (a) a 3-d bell curve with height represents the probability density, (b) ellipse contour projections showing the correlation between x_1 and x_2 points.

and x_2 respectively. The covariance matrix is $\begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}$, where the diagonal terms σ_{11} and σ_{22} are the independent variance of x_1 and x_2 respectively. The off-diagonal terms σ_{12} and σ_{21} represent correlations between x_1 and x_2 . The BVN is expressed as

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} \right) \sim \mathcal{N}(\mu, \Sigma).$$

It is intuitively understandable that we need the conditional probability rather than the joint probability for regression tasks. If we cut a slice on the 3-d bell curve shown in Fig. 5, we get the conditional probability distribution $P(x_1 | x_2)$ shown in Fig. 6. The conditional distribution is also Gaussian [11].

2.3 Kernels

After reviewing the MVN concept, we want to smooth the functions in Fig. 4(b). This can be done by defining covariance functions, which reflect our ability to express prior knowledge of the form of the function we are modelling. In regression, outputs of the function should be similar when two input are close to each other. One possible equation format is the dot product $A \cdot B = \|A\| \|B\| \cos \theta$, where θ is

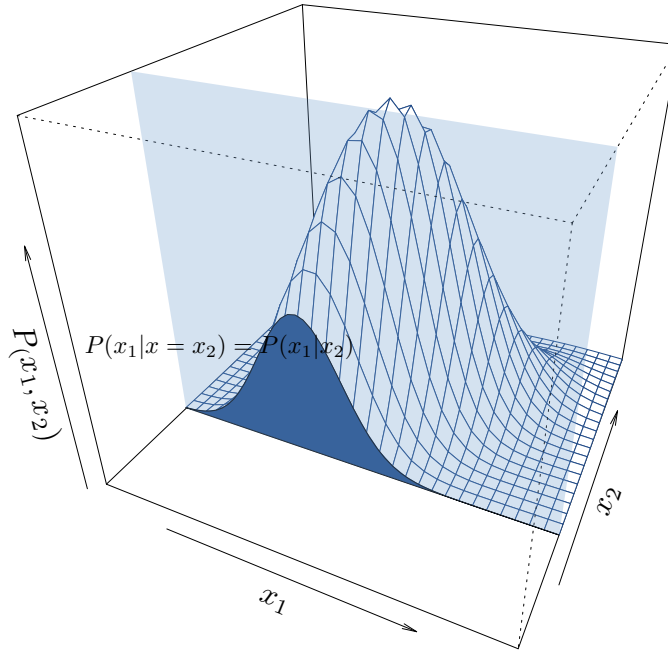


Figure 6: The conditional probability distribution $P(x_1|x_2)$ by cutting a slice on the PDF 3-d bell curve of a BVN.

the angle between two input vectors. When two input vectors are similar, their dot product output value is high.

If a function is defined solely in terms of inner products in the input space, then the function $k(x, x')$ is a kernel function [11]. The most widely used kernel or covariance function is the squared exponential (SE) kernel function. The SE kernel is de-facto default kernel for Gaussian processes [5]. This is because it can be integrated against most functions that you need to due to its universal property. And every function in its prior has infinitely many derivatives. It's also known as the radial basis function (RBF) kernel or Gaussian kernel function that is defined as ¹

$$\text{cov}(x_i, x_j) = \exp \left(- \frac{(x_i - x_j)^2}{2} \right) .$$

In Fig. 4(b), we plotted 20 independent Gaussian vectors by connecting 10 random selected sample points of each vector in order by lines. Instead of plotting 20

¹This is a simplified RBF without hyperparameters for simplicity. A general RBF will be further explained in section 4.1.

independent Gaussian, we generated 10 twenty-variate normal (20-VN) distribution with an identity covariance function shown in 7(a). It's the same as 4(b) due to there is no correlations between points by using an identity as its kernel function. When a RBF was used as the covariance function, we got smooth lines shown in 7(b).

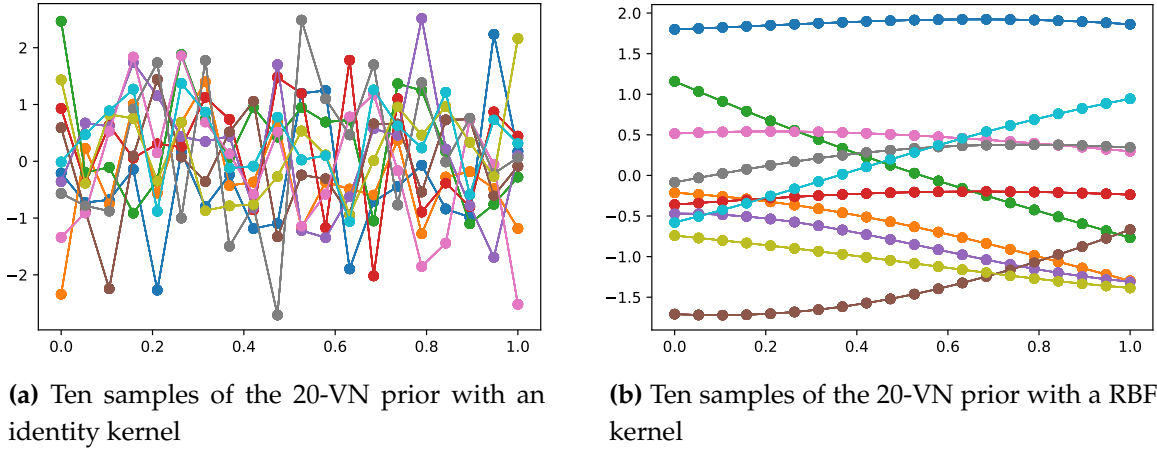


Figure 7: Samples of twenty-variate normal (20-VN) distribution kernelized prior functions: (a) Ten 20-VN with an identity function covariance, (b) Ten 20-VN with a RBF covariance.

By adding covariance functions, we obtain smoother lines, and they start to look like functions. It is natural to consider continuing to increase the dimension of MVN. Here, dimension refers to the number of multi-variables of MVN. When the dimension of MVN gets larger, the region of interest will be filled up with more points. When the dimension becomes infinity, there will be a point to represent any possible input point. By using MVN with infinity dimensions, we fit functions with infinity parameters to do regression tasks. We can make predictions anywhere in the region of interest. In Fig. 8, We plotted 200 samples of a two hundred-variate normal (200-VN) distribution to get a feeling of functions with infinity parameters. We call these functions as kernelized prior functions because there are no observed data points yet. All functions are random generated by the MVN model incorporating with kernel functions as prior knowledge before having any observed data points.

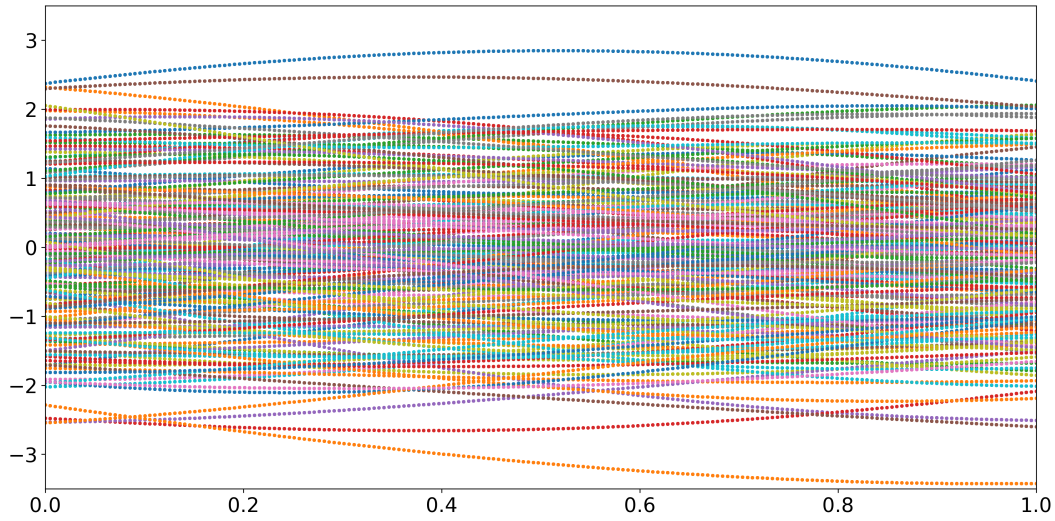


Figure 8: Two hundred kernelized prior functions of a two hundred-variate normal distribution.

2.4 Non-parametric Model

This section explains the concept of parametric and nonparametric model [10]. Parametric models assume that the data distribution can be modeled in terms of a set of finite number of parameters. In regression, given some data points, we would like to make predictions of the function value $y = f(x)$ with a new specific x . If we assume a linear regression model, $y = \theta_1 + \theta_2 x$, we need to find the parameters θ_1 and θ_2 to define the function. In many cases, the linear model assumption isn't hold, a polynomial model with more parameters, such as $y = \theta_1 + \theta_2 x + \theta_3 x^2$ is needed. We use the training dataset D with n observed points, $D = [(x_i, y_i) | i = 1, \dots, n]$ to train the model, i.e. mapping x to y through basis functions $f(x)$. After the training process, we assume all information of the data are captured by the feature parameters θ , thus predictions are independent to the training dataset D . It can be expressed as $P(f_* | X_*, \theta, D) = P(f_* | X_*, \theta)$, in which f_* are predictions made at unobserved data points X_* . Thus, when conducting regressions using parametric models, the complexity or flexibility of models are limited by the number of parameters. If the number of parameters of a model grows with the size of observed dataset, it's a non-parametric model. A non-parametric model does not imply that there are no parameters, but rather infinite parameters.

3 Gaussian Processes

Before explaining the Gaussian process, let's do a quick review of the basic concepts we covered. In regression, there is a function f we are trying to model given observed data points D (training dataset) from the unknown function f . The traditional nonlinear regression methods typically give one function that is considered to fit the dataset best. However, there may be more than one function that fit the observed data points equally well. We saw that when the dimension of MVN was infinite, we could make predictions at any point with these infinite number of functions. These functions are MVN because it's our assumption (prior). More formally speaking, the prior distribution of these infinite functions is MVN. The prior distribution represents the expected outputs of f over inputs x without observing any data. When we start to have observations, instead of infinite number of functions, we only keep functions that fit the observed data points. Now we have the posterior, which is the prior updated with observed data. When we have new observations, we will use the current posterior as prior, and new observed data points to obtain a new posterior.

Here is a definition of **Gaussian processes**: A Gaussian processes model is a probability distribution over possible functions that fit a set of points. Because we have the probability distribution over all possible functions, we can calculate the means as the function, and the variances to indicate how confident the predictions are. The key points are summarized as: 1) the function (posteriors) updates with new observations, 2) a Gaussian process model is a probability distribution over possible functions, and any finite sample of functions are jointly Gaussian distributed, 3) the mean function calculated by the posterior distribution of possible functions is the function used for regression predictions.

It is time to describe the standard Gaussian processes model. All the parameter definitions follow the classic textbook [11]. Besides the covered basic concepts, the Appendix A.1 and A.2 of [11] are also recommended to read. The regression function modeled by a multivariate Gaussian is given as

$$P(f | \mathbf{X}) = \mathcal{N}(f | \boldsymbol{\mu}, \mathbf{K}) ,$$

where $\mathbf{X} = [x_1, \dots, x_n]$, $f = [f(x_1), \dots, f(x_n)]$, $\boldsymbol{\mu} = [m(x_1), \dots, m(x_n)]$ and $K_{ij} = k(x_i, x_j)$. \mathbf{X} are the observed data points, m represents the mean function, and k represents a positive definite kernel function. With no observation, the mean function is default to be $m(\mathbf{X}) = 0$ given that the data is often normalized to a zero mean. The Gaussian processes model is a distribution over functions whose shape

(smoothness) is defined by \mathbf{K} . If points \mathbf{x}_i and \mathbf{x}_j are considered to be similar by the kernel, function outputs of the two points, $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$, are expected to be similar. The process of conducting regressions by Gaussian processes model is illustrated by Fig. 9: given the observed data (red points) and a mean function \mathbf{f} (blue line) estimated by these observed data points, we make predictions at new points \mathbf{X}_* as $\mathbf{f}(\mathbf{X}_*)$.

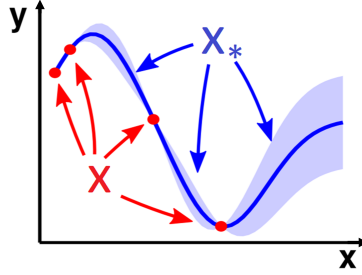


Figure 9: A illustrative process of conducting regressions by Gaussian processes. The red points are observed data, blue line represents the mean function estimated by the observed data points, and predictions will be made at new blue points.

The joint distribution of \mathbf{f} and \mathbf{f}_* is expressed as

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{bmatrix} \right),$$

where $\mathbf{K} = K(\mathbf{X}, \mathbf{X})$, $\mathbf{K}_* = K(\mathbf{X}, \mathbf{X}_*)$ and $\mathbf{K}_{**} = K(\mathbf{X}_*, \mathbf{X}_*)$. And $(m(\mathbf{X}), m(\mathbf{X}_*)) = \mathbf{0}$.

This is the joint probability distribution equation $P(\mathbf{f}, \mathbf{f}_* | \mathbf{X}, \mathbf{X}_*)$ over \mathbf{f} and \mathbf{f}_* , but regressions need the conditional distribution $P(\mathbf{f}_* | \mathbf{f}, \mathbf{X}, \mathbf{X}_*)$ over \mathbf{f}_* only. The derivation from the joint distribution $P(\mathbf{f}, \mathbf{f}_* | \mathbf{X}, \mathbf{X}_*)$ to the conditional $P(\mathbf{f}_* | \mathbf{f}, \mathbf{X}, \mathbf{X}_*)$ uses the theorem in Appendix A. The result is

$$\mathbf{f}_* | \mathbf{f}, \mathbf{X}, \mathbf{X}_* \sim \mathcal{N} \left(\mathbf{K}_*^\top \mathbf{K}^{-1} \mathbf{f}, \mathbf{K}_{**} - \mathbf{K}_*^\top \mathbf{K}^{-1} \mathbf{K}_* \right).$$

In more realistic situations, we don't have access to true function values but noisy versions thereof $y = f(x) + \epsilon$. Assuming there is an additive independent and identically distributed (i.i.d.) Gaussian noise with variance σ_n^2 , the prior on the noisy observations becomes $\text{cov}(y) = \mathbf{K} + \sigma_n^2 \mathbf{I}$. The joint distribution of the observed values and the function values at new testing points becomes

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix} \sim \mathcal{N} \left(\mathbf{0}, \begin{bmatrix} \mathbf{K} + \sigma_n^2 \mathbf{I} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{bmatrix} \right).$$

By deriving the conditional distribution, we get the predictive equations for Gaussian processes regression as

$$\bar{\mathbf{f}}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*)) ,$$

where

$$\begin{aligned} \bar{\mathbf{f}}_* &\triangleq \mathbb{E}[\bar{\mathbf{f}}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*] \\ &= \mathbf{K}_*^T [\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y} , \\ \text{cov}(\mathbf{f}_*) &= \mathbf{K}_{**} - \mathbf{K}_*^T [\mathbf{K} + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{K}_* . \end{aligned}$$

In the variance function $\text{cov}(\mathbf{f}_*)$, it can be noted that the variance does not depend on the observed output \mathbf{y} but only on the inputs \mathbf{X} and \mathbf{X}_* . This is a property of the Gaussian distribution [11].

4 Illustrative example

An implementation of the standard GPR is shown in this section. The example follows the algorithm in [11] as follows.

$$\begin{aligned} L &= \text{cholesky}(K + \sigma_n^2 I) \\ \boldsymbol{\alpha} &= L^\top \setminus (L \setminus \mathbf{y}) \\ \bar{f}_* &= \mathbf{k}_*^\top \boldsymbol{\alpha} \\ \mathbf{v} &= L \setminus \mathbf{k}_* \\ \mathbb{V}[f_*] &= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v} . \\ \log p(\mathbf{y} | \mathbf{X}) &= -\frac{1}{2} \mathbf{y}^\top (K + \sigma_n^2 I)^{-1} \mathbf{y} - \frac{1}{2} \log \det(K + \sigma_n^2 I) - \frac{n}{2} \log 2\pi \end{aligned}$$

The inputs are \mathbf{X} (inputs), \mathbf{y} (targets), k (covariance function), σ_n^2 (noise level), and \mathbf{x}_* (test input). The outputs are \bar{f}_* (mean), $\mathbb{V}[f_*]$ (variance), and $\log p(\mathbf{y} | \mathbf{X})$ (log marginal likelihood).

An example result was plotted shown in Fig. 10. We conducted regression within the $[-5, 5]$ region of interest. The observed data points (training dataset) were generated from a uniform distribution between -5 and 5. Any point within the given interval $[-5, 5]$ was equal likely to be drawn by a uniform. The functions were evaluated at evenly spaced points between -5 and 5. The regression function is composed of mean values estimated by a GPR model. Twenty samples of posterior mean functions were also plotted within 3μ variances.

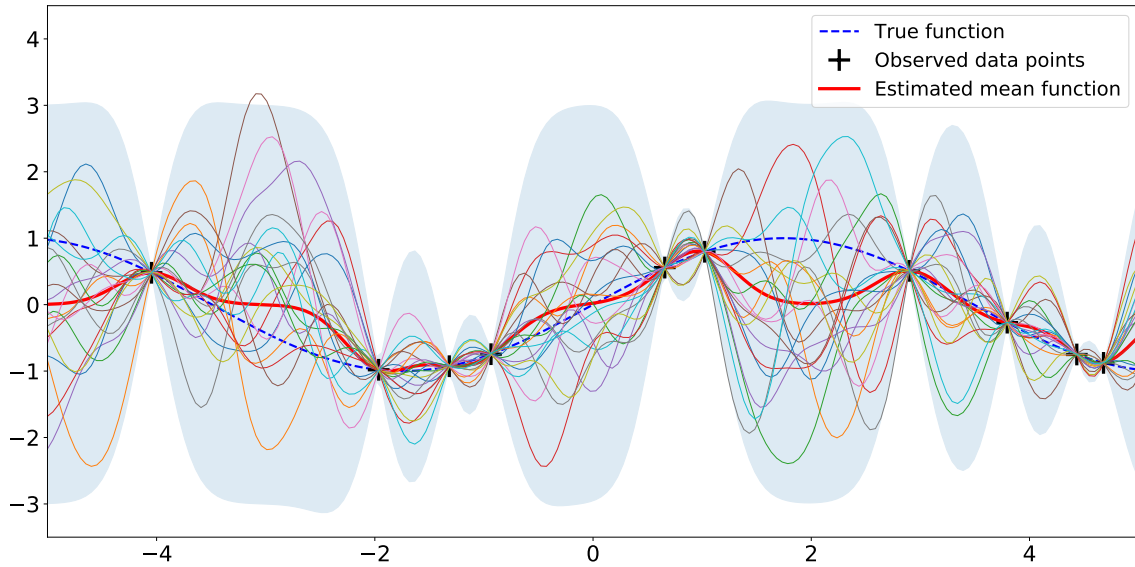


Figure 10: A illustrative example of the standard GPR. The observed data points generated by the blue dotted line (the true function) were plotted as black crosses. Given the observed/training data points, infinite possible posterior functions were obtained. We plotted 20 samples of these infinite functions with sorted colors. The mean function was obtained by the probability distribution of these functions and plotted as a red solid line. The blue shaded area around the mean function indicates the 3σ prediction variances.

4.1 Hyperparameters Optimization

We explained the basics of GPR and implemented a simple example. While in practical, GPR models are more complex than it. Kernel functions play significant roles in GPR. The choice of kernel functions determines almost all the generalization properties of a GP model [4]. There are many covariance functions to choose or make your own for a Gaussian process depending on your specific problem. These criteria include if the model is smooth, if it is sparse, if it can change drastically, and if it needs to be differentiable [5]. More depth information on choosing a kernel/covariance function for a Gaussian process can be found in [5]. In kernels, the hyperparameters optimization is essential. Here we will use the most widely used kernel, RBF, as an example to explain the hyperparameters optimization. The general RBF function is given by

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp \left(-\frac{1}{2l} (\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j) \right),$$

where σ_f and l are hyperparameters [5]. The vertical scale σ_f describes how much vertically the function can span. The horizontal scale l indicates how quickly the correlation relationship between two points drops as their distance increases. The effect of l was shown in Fig. 11. A higher l provided a smoother function and a smaller l gave a wigglier function.

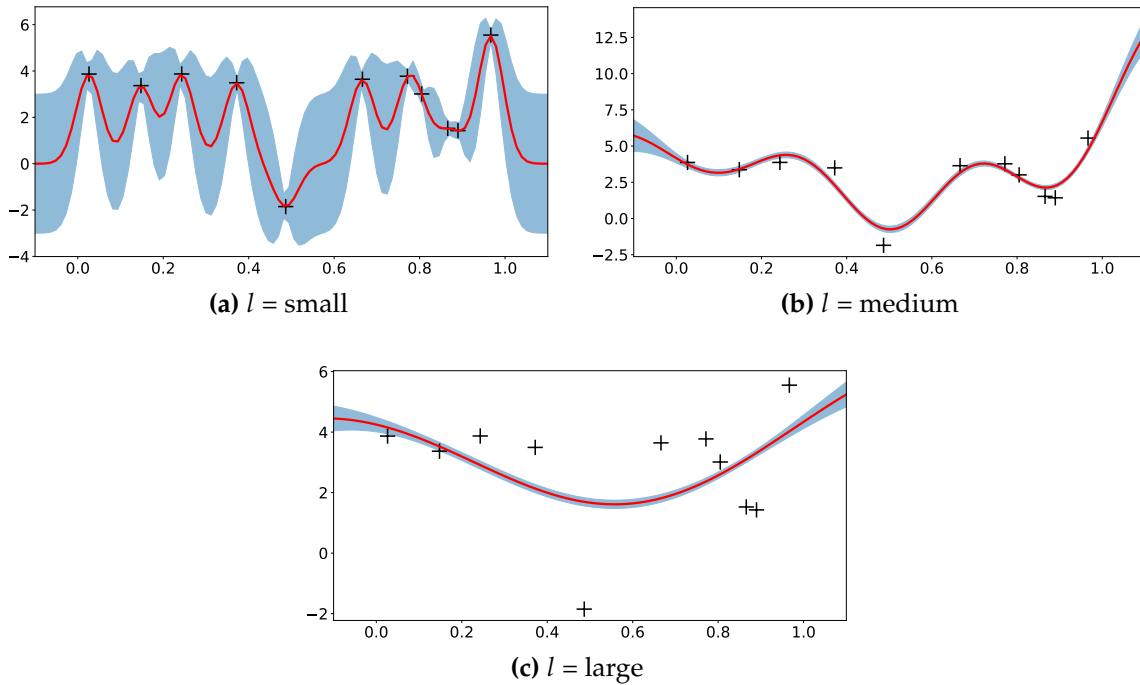


Figure 11: The function smoothness affected by the horizontal scale l of hyperparameters.

The optimized hyperparameters Θ^* are determined by the log marginal likelihood [11] as

$$\Theta^* = \arg \max_{\Theta} \log p(\mathbf{y} | \mathbf{X}, \Theta).$$

Thus, considering hyperparameters, a more general equation of predictions at the new testing points is [2]

$$\bar{\mathbf{f}}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*, \Theta \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*)) .$$

Note that after learning/tuning the hyperparameters, the predictive variance $\text{cov}(\mathbf{f}_*)$ depends on not only the inputs \mathbf{X} and \mathbf{X}_* but also the outputs \mathbf{y} [1]. With the optimized hyperparameters $\sigma_f = 0.0067$ and $l = 0.0967$, the regression result of the

observed data points in Fig. 11 was shown in Fig. 12. Here, the hyperparameters optimization was conducted by the GPy package, which will be introduced in the next section.

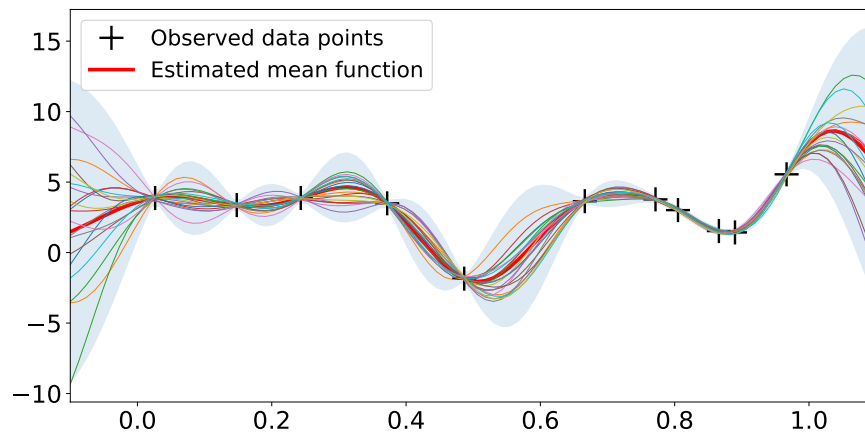


Figure 12: Regression result with the optimized hyperparameters σ_f and l .

4.2 Gaussian processes packages

This section will review three software packages written in Python for Gaussian processes implementations. One of the most well-known GP packages is GPy [3]. GPy is a maturely developed package with well-documented explanations since 2012. GPy uses NumPy to perform all its computations. For tasks that do not require heavy computations, GPy is sufficient and stable to use. However, GPR is computationally expensive in high dimensional spaces (features more than a few dozens) because it uses the whole samples/features to do predictions. The more observations, the more computations are needed for predictions. A package that includes state-of-the-art algorithms is preferred for efficient implementations of complex GPR tasks. For GPR tasks that are computationally expensive, GPU acceleration is especially useful. GPflow [3] originates from GPy with much of a similar interface. GPflow leverages TensorFlow as its computational backend. GPyTorch [7] is a recently developed package that provides GPU acceleration through PyTorch. It also contains many state-of-the-art Gaussian processes algorithms. Similar to GPflow, GPyTorch provides automatic gradients, thus complex models such as embedding deep neural networks in GP models can be easier developed.

5 Summary and Discussion

A Gaussian process is a probability distribution over possible functions that fit a set of points [11]. A Gaussian processes regression model provides uncertainty estimations together with prediction values. The prior knowledge of the form of functions can be integrated by using kernel functions.

The GPR model in this tutorial is the standard or vanilla Gaussian processes [6]. There are two main constraints with it: 1) The overall computation complexity is $O(N^3)$, where N is the dimension of the covariance matrix K . 2) The memory consumption is quadratic. Because of the computation complexity and memory consumption, the standard Gaussian processes model gets struck quickly. For regression tasks with a big dataset, sparse GP is used to reduce computational complexity [9].

References

- [1] Zexun Chen and Bo Wang. How priors of initial hyperparameters affect Gaussian process regression models. *Neurocomputing*, 275:1702–1710, 2018.
- [2] Zhenwen Dai. Computationally efficient GPs. In *Gaussian Process Summer School (GPSS)*, 2019.
- [3] Alexander G De G. Matthews, Mark Van Der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagr , Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *The Journal of Machine Learning Research*, 18(1):1299–1304, 2017.
- [4] David Duvenaud. The Kernel Cookbook. Available at <https://www.cs.toronto.edu/~duvenaud/cookbook>.
- [5] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.
- [6] Roger Frigola, Fredrik Lindsten, Thomas B Sch n, and Carl Edward Rasmussen. Bayesian Inference and Learning in Gaussian Process State-Space Models with Particle MCMC. In *Advances in Neural Information Processing Systems*, pages 3156–3164, 2013.

- [7] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. In *Advances in Neural Information Processing Systems*, 2018.
- [8] Zoubin Ghahramani. A Tutorial on Gaussian Processes (or why I don't use SVMs). In *Machine Learning Summer School (MLSS)*, 2011.
- [9] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When Gaussian process meets big data: A review of scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
- [10] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
- [11] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

A Appendix

The Marginal and Conditional Distributions of MVN theorem: suppose $X = (x_1, x_2)$ is a joint Gaussian with parameters

$$\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}, \Lambda = \Sigma^{-1} = \begin{bmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{bmatrix},$$

then the marginals are given by

$$\begin{aligned} p(x_1) &= \mathcal{N}(x_1 | \mu_1, \Sigma_{11}), \\ p(x_2) &= \mathcal{N}(x_2 | \mu_2, \Sigma_{22}), \end{aligned}$$

and the posterior conditional is given by

$$\begin{aligned} p(x_1 | x_2) &= \mathcal{N}(x_1 | \mu_{1|2}, \Sigma_{1|2}) \\ \mu_{1|2} &= \mu_1 + \Sigma_{12} \Sigma_{22}^{-1} (x_2 - \mu_2) \\ &= \mu_1 - \Lambda_{11}^{-1} \Lambda_{12} (x_2 - \mu_2) \\ &= \Sigma_{1|2} (\Lambda_{11} \mu_1 - \Lambda_{12} (x_2 - \mu_2)) \\ \Sigma_{1|2} &= \Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21} = \Lambda_{11}^{-1} \end{aligned}$$