An Intuitive Tutorial to Gaussian Processes Regression

Jie Wang
Ingenuity Labs Research Institute
jie.wang@queensu.ca

September 25, 2020

Offroad Robotics
c/o Ingenuity Labs Research Institute
Queen's University
Kingston, ON K7L 3N6 Canada

Abstract

This introduction aims to provide readers an intuitive understanding of Gaussian processes regression. Gaussian processes regression (GPR) models have been widely used in machine learning applications because their representation flexibility and inherently uncertainty measures over predictions. The paper starts with explaining mathematical basics that Gaussian processes built on including multivariate normal distribution, kernels, non-parametric models, joint and conditional probability. The Gaussian processes regression is then described in an accessible way by balancing showing unnecessary mathematical derivation steps and missing key conclusive results. An illustrative implementation of a standard Gaussian processes regression algorithm is provided. Beyond the standard Gaussian processes regression, existing software packages to implement state-of-the-art Gaussian processes algorithms are reviewed. Lastly, more advanced Gaussian processes regression models are specified. The paper is written in an accessible way, thus undergraduate science and engineering background will find no difficulties in following the content.

Contents

1 Introduction

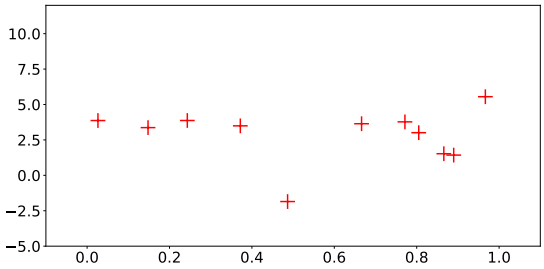
The Gaussian processes is a probabilistic supervised machine learning framework that has been widely used for regression and classification tasks. The Gaussian processes regression (GPR) model can make predictions incorporating prior knowledge (kernels) and provide uncertainty measures over predictions [13]. Besides signal processing [6], GPR is actively used for dynamic system model estimations in reinforcement learning [3] and control [12]. GPR can provide uncertainty estimations not only for one-step prediction but also for multi-step predictions by propagating the uncertainty multi-step ahead [8]. Due to its ability of multi-step uncertainty propagation, GPR is increasingly applied together with model predictive control (MPC) in the past several years [9]. The strength of using GPR is that it can model the unmodeled dynamics and unknown disturbances that are hard to model by the first principle method. More importantly, the GPR model quantifies uncertainty during the model learning to indicate areas with less certain to be explored more. An efficient implementation of GPR makes online learning possible to control tasks [10].

Gaussian processes model is a supervised learning method mainly developed within the computer science and statistics communities. Researchers with engineering backgrounds often find it's difficult to gain a clear understanding of GPR easily. To understand GPR, even to the intuitive level, needs to have mathematics foundation knowledge including multivariate normal distribution (MVN), kernels, non-parametric model, and joint and conditional probability. There is often a gap between using GPR as a tool and feeling comfortable to use it due to the difficulties in understanding these necessary theories. This introduction aims to bring readers an intuitive understanding of GPR in an accessible way. All implementation codes in this paper are provided at: https://github.com/jwangjie/An-Intuitive-Tutorial-to-Gaussian-Processes-Regression.

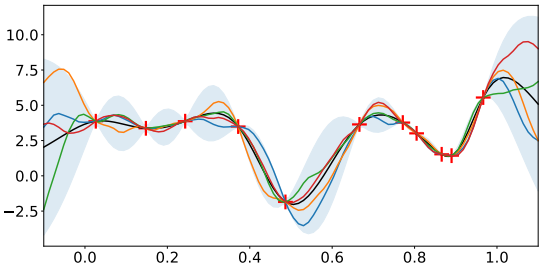
2 Mathematical Basics

This section covers the mathematical basics that GPR models are built on. It starts with Gaussian (normal) distribution, then goes through multivariate normal distribution (MVN), kernels, non-parametric model, and joint and conditional probability. In order to obtain a clear understanding of GPR, unnecessary math is neglected on purpose. First of all, what is regression? Regression is a common machine learn-

ing task that is described as given some observed data points, fitting a function that represents these data point features, then using 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 the data points. In Fig. 1(b), we show five samples of potential infinite numbers of functions that fit the data points in Fig. 1(a).



(a) Data point observations



(b) Five possible regression functions by GPR

Figure 1: A regression example: (a) Some observed data points, (b) Five samples of potential infinite number of functions that fit the observed data points.

2.1 Gaussian Distribution

A random variable X is said to be Gaussian or normally distributed with mean μ and variance σ^2 if its probability density function (PDF) is [11]

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

Here, X represents random variables and x is the real argument. The normal distribution of X is usually represented by $P(x) \sim \mathcal{N}(\mu, \sigma^2)$. A one dimensional (1-d) Gaussian PDF was plotted in Fig. 2. We generated $n = 1000$ numbers of random sample points from a 1-d Gaussian distribution and plotted them on the x axis.

These generated data can be viewed as a vector $X_1 = [x^{(1)}, x^{(2)}, \ldots, x^{(n)}]$. By plotting the vector X_1 on a new X axis at $X = 0$, we projected all points of X_1 to another space shown in Fig. 3. We did nothing but vertically plot points of the vector X_1 in the $X - f(X)$ coordinates space. We can plot another independent Gaussian vector X_2 within the same coordinates at $X = 1$. Keep in mind that each vector X_1 and X_2 is normal distributed shown in Fig. 3. Next, we random selected 10 point of X_1 and X_2 respectively and connected points of these 10 points in order

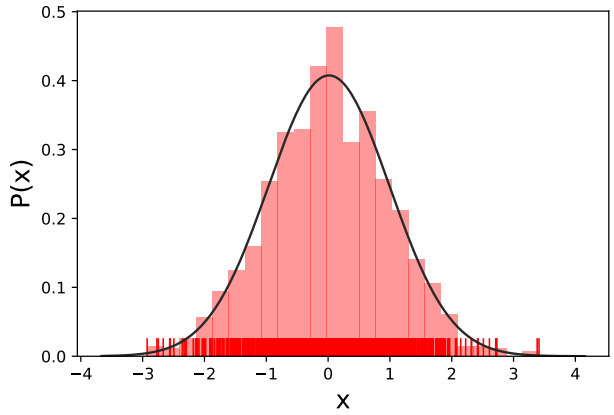


Figure 2: One thousand random sample points from a 1-d Gaussian was plotted as blue vertical bars on the x axis. The PDF of these data points was plotted as a two dimensional bell curve.

by lines shown in Fig. 4(a). These lines look like linear functions spanning within the $[0, 1]$ domain. Going back to think about regression, we can use these functions

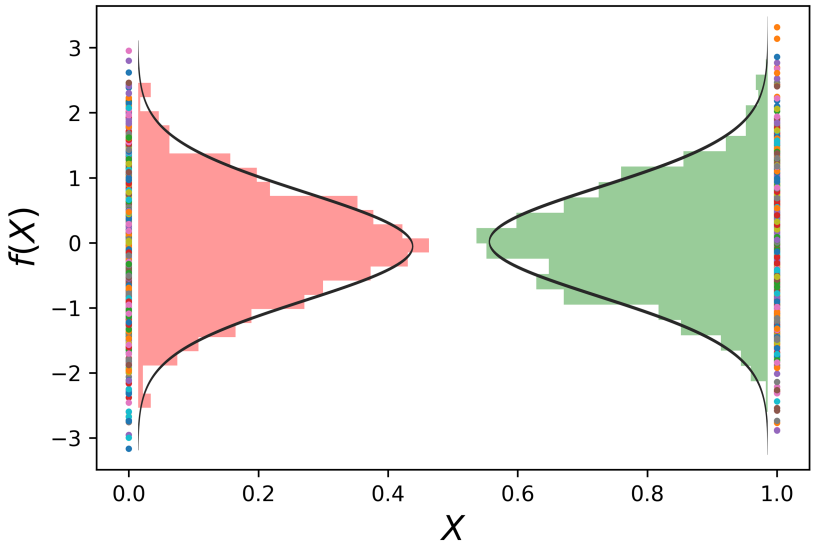
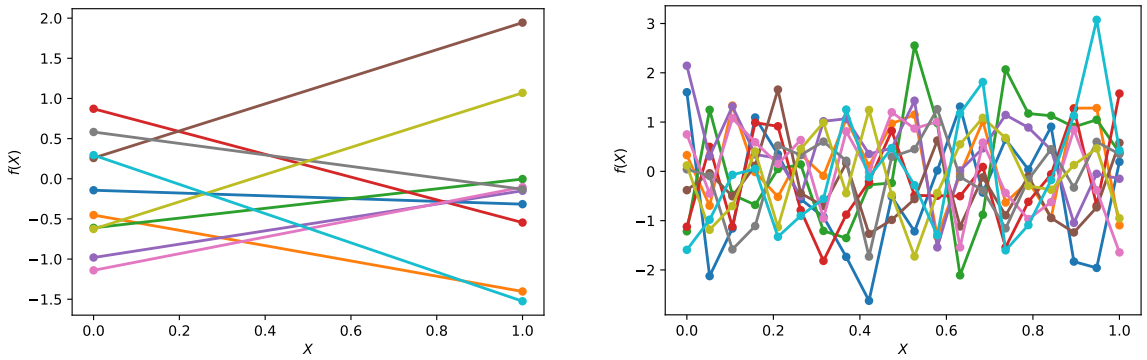


Figure 3: Two independent 1-d Gaussian vector points were plotted vertically.

to make predictions if the new data points are on (or close enough to) these lines. The assumption that new data points are on the connected linear functions is too strong to be guaranteed because new data points can be anywhere within $[0, 1]$. If we plot more random generated normal distributed vectors, for example, 20 vectors $X_1, X_2, \ldots, X_{20}$ between $[0, 1]$, and connect 10 random selected sample points

of each vector as lines shown in Fig. 4(b), we got 10 lines that look like functions within $[0, 1]$. However, we can't 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 function values. In a word, "functions" by connecting random generated independent Gaussian points are not smooth enough for regression tasks, we need these independent Gaussian distributions correlated to each other as joint Gaussian distributions. The joint Gaussian distribution is described by the multi-variable Gaussian theory.



(a) Two Gaussian vectors **(b)** Twenty Gaussian vectors

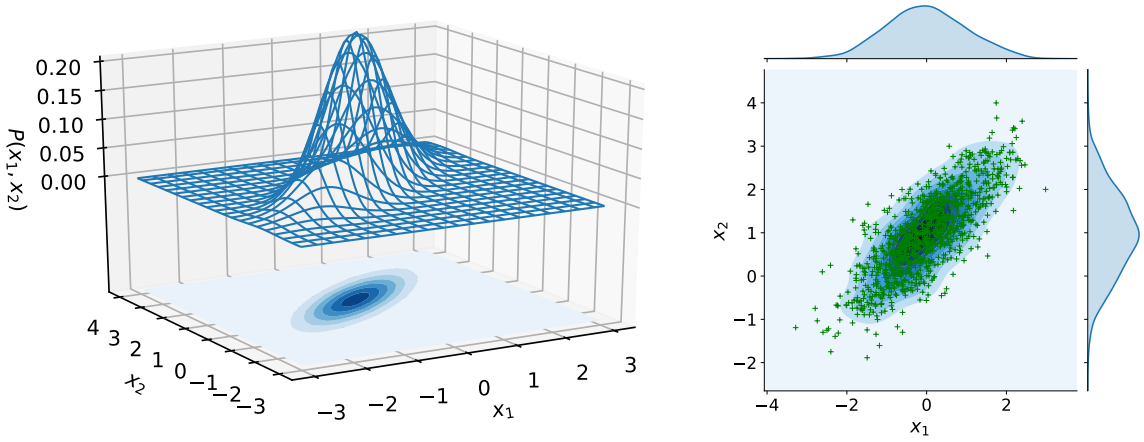
Figure 4: Connecting points of independent Gaussian vectors by lines.

2.2 Multivariate Normal Distribution

In many situations, a system (set of data) is described by more than one feature variables $[X_1, X_2, \ldots, X_D]$ that are correlated to each other. If we want to model the variables all together as one Gaussian model, it's a multivariate Gaussian/normal (MVN) [11] distribution model. We use a two dimensional (2-d) MVN model as an example. A 2-d MVN can be visualized as a three dimensional (3-d) bell curve with height represents the probability density shown in Fig. 5a. The 3-d bell curve projection on the $x_1 - x_2$ plane plotted in Fig. 5a and 5b are ellipse contours showing the co-relationship between $x_1$and x_2 points. The $P(x_1, x_2)$ is the joint probability of X_1 and X_2.

Formally, the PDF of the MVN with D dimension is defined as [11]

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



(a) 3-d bell curve **(b)** 2-d ellipse contours

Figure 5: The PDF of a 2-d MVN visualization: (a) a 3-d bell curve with height represents the probability density, (b) ellipse contour projections showing the co-relationship between $x_1$and x_2 points.

where D is the number of 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 the jointly modeled random variables with its (i, j) element as $\Sigma_{ij} = \text{cov}(y_i, y_j)$. For a 2-d MVN, the mean vector μ is a 2-d vector $\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$, which are independent mean of each variable x_1 and x_2. The covariance matrix is $\begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix}$. The diagonal terms are independent variance of each variable x_1 and x_2. The off-diagonal terms represents correlations between the two variables that represents how much one variable x_1 is related to another variable x_2. A 2-d Gaussian is expressed as

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix} \right) \sim \mathcal{N}(\mu, \Sigma) \, .$$

Instead of the joint probability, we are more interested to the conditional probability for the regression tasks. If we cut a slice on the 3-d bell curve or draw a line on the ellipse contours 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 [13].

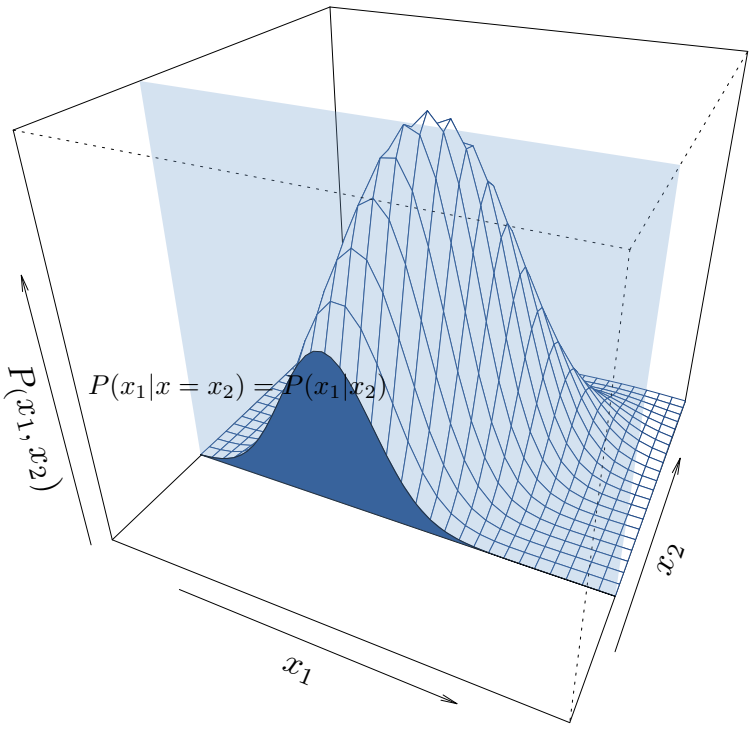


Figure 6: The conditional probability distribution $P(x_1|\, x_2)$.

2.3 Kernels

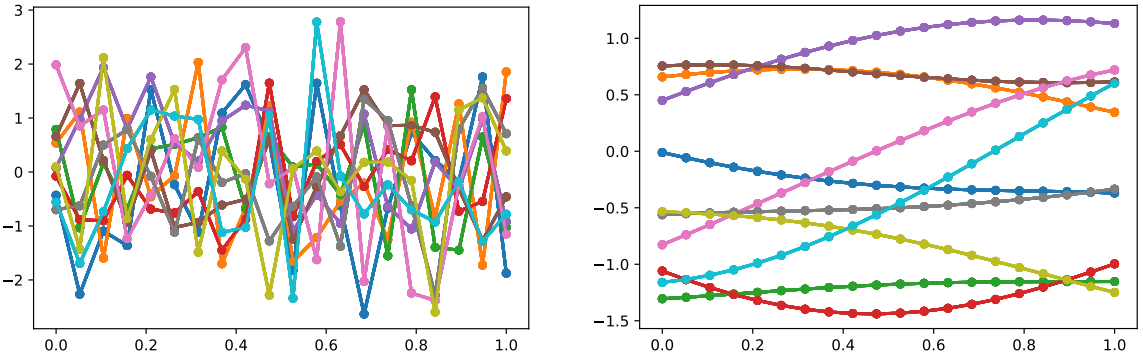
After reviewing MVN, we smooth the functions in Fig. 4(b) by defining the covariance functions. For regression tasks, outputs of the function should be similar when two input points are similar to each other. One possible function format is dot product function due to the fact that when two input vectors are similar, their dot product output value is high. It is clear in the dot product equation $A \cdot B = \|A\|\|B\| \cos\theta$, where θ is the angle between two input vectors. In fact, if a function is defined solely in terms of inner products in input space as $k(x,\ x')$; we call $k(\cdot, \cdot)$ a kernel function [13]. The most widely used covariance function (kernel function) is the squared exponential kernel function. It's also called the radial basis function (RBF) kernel or Gaussian kernel function that is defined as [1]

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

Let's re-plot 20 independent Gaussian vector points and connecting points in

[1] This is a simplified RBF without hyperparameters for simplicity. A general RBF is explained in section 4.1.

order by lines shown in Fig. 4(b). Instead of 20 independent Gaussian that has 10 sample points in each vector, we generated 10 twenty dimensional (20-d) MVN with an identity covariance matrix function shown in 7(a). It's the same as 4(b) because there is no smoothness added by using an identity covariance matrix as its kernel function. When the RBF was used as the covariance function to generate 10 samples of 20-d MVN, we got smooth lines shown in 7(b).



(a) 10 samples of the 20-d MVN prior with identity kernel

(b) 10 samples of the 20-d MVN prior with RBF kernel

Figure 7: Ten samples of 20-d MVN kernelized prior functions.

We get smooth lines and they look more like functions. It's natural to consider to continue increasing the dimension of MVN. When the dimension of MVN gets larger, the region of interest is filled up with more points. When the dimension becomes infinity, there is a point that represents any possible input. By using infinity dimensional MVN, we are fitting functions with infinity parameters for regression tasks. We can make predictions anywhere in the region of interest. We can plot 'm=200' samples of a 'n=200' dimensional MVN to get a feeling of functions with infinity parameters shown in Fig. 8. We call these functions as kernelized prior functions because there are no observed data points yet. All functions are generated by the MVN model incorporating kernel function as prior knowledge before having any observed data points.

2.4 Nonparametric model

This section explains the basics of parametric and nonparametric models [11]. Parametric models assume that the data distribution can be modeled in terms of a set of finite numbers of parameters. In regression, given some data points, we

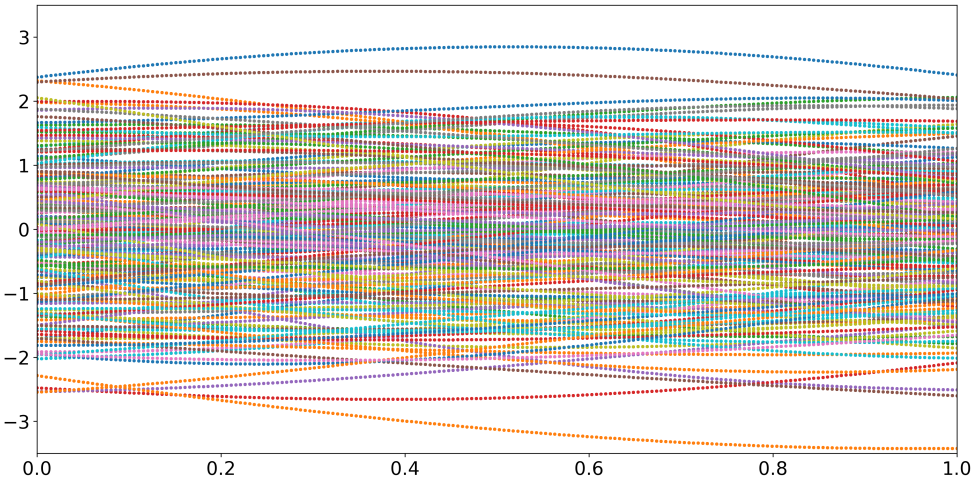


Figure 8: 200 samples of a 200 dimensional MVN kernelized prior functions.

would like to make predictions of the function value $y = f(x)$ with a 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, $D = [(x_i, y_i) \,|\, i = 1, \ldots, n]$ to train the model, i.e. mapping x to y through parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$. After the training process, we assume all the information of the data is captured by the feature parameters $\boldsymbol{\theta}$, thus predictions are independent of the training dataset D. It can be expressed as $P(f_* \,|\, X_*, \boldsymbol{\theta}, D) = P(f_* \,|\, X_*, \boldsymbol{\theta})$, in which f_* are predictions made at unobserved data points X_*. Thus, conducting regression using parametric models, the complexity or flexibility of the model is limited by the parameter numbers. It's natural to think to use a model that the number of parameters grows with the size of the dataset, and it's a Bayesian nonparametric model. The Bayesian nonparametric model doesn't imply that there are no parameters, but rather infinite parameters.

3 Math

3.1 Definition

Before diving into equations, let's review the covered mathematical basics. In regression, there is a function $\mathbf{f}$ we are trying to model given a set of data points $\mathbf{X}$ (training data/existing observed data) from the unknown function $\mathbf{f}$. The tradi-

tional nonlinear regression methods typically give one function that is considered to fit the dataset best. However, there may be more than one function that can fit the observed data points equally well. We saw that when the dimension of MVN is infinite, we can make predictions at any point with these infinite number of functions. These functions are MVN because it's our assumption (prior). More formally, the prior distribution of these infinite functions is MVN. The prior distribution representing the outputs $\mathbf{f}$ that we expect to see over some inputs $\mathbf{x}$ without observing any data. When we start to have observation points, instead of infinite functions, we keep functions that fit the observed points. Now we got the posterior, the current belief based on existing observations. When we have new observation points, we use the current posterior as the prior, use new observation points to update the posterior.

This is **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 functions(posterior) updates with new observations. 2) The mean function calculated by the posterior distribution of the possible infinity functions is the function used for regression predictions. 3) A Gaussian processes model is a probability distribution over possible functions, and any finite sample of functions is jointly Gaussian distributed.

3.2 Predictions

This section provides a concise description of the standard Gaussian processes algorithm. The parameter definition follows [13]. Besides the covered mathematics foundation, it's highly recommended to read Appendix A.1 and A.2 of [13] before continuing to read. The regression function is modeled by a multivariate Gaussian as

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

where $\mathbf{X} = [x_1, ..., x_n]$, $\mathbf{f} = [f(\mathbf{x}_1), ..., f(\mathbf{x}_n)]$, $\boldsymbol{\mu} = [m(\mathbf{x}_1), ..., m(\mathbf{x}_n)]$ and $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. $\mathbf{X}$ is the observed data points, m represents the mean function and it is common to set $m(\mathbf{x}) = 0$ as Gaussian processes is flexible enough to model the mean arbitrarily well. k represents a positive definite kernel function. 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,

the function output values at the two points, $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$, are expected to be similar. The regression by Gaussian processes is illustrated in Fig. 9: given some observed data points (red points) and a mean function $\mathbf{f}$ estimated by these points (blue line), we make predictions at some new points $\mathbf{X}_*$ as $f(\mathbf{X}_*)$.

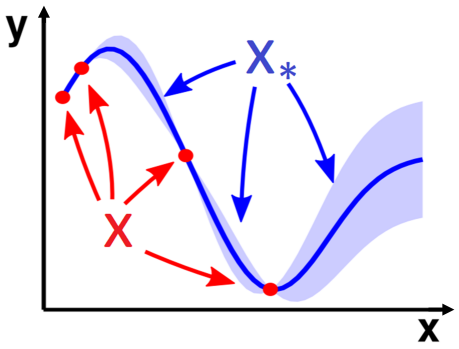


Figure 9: The illustrative process of Gaussian processes regression.

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

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

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

This is the joint probability distribution equation $P(\mathbf{f}, \mathbf{f}_* \,|\, \mathbf{X}, \mathbf{X}_*)$ over $\mathbf{f}$ and $\mathbf{f}_*$, but regression tasks 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

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

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 $\mathrm{cov}(y) = \mathbf{K} + \sigma_n^2 I$. The joint distribution of the observed values and the function values at the new testing points becomes

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

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

$$\bar{\mathbf{f}}_* \mid \mathbf{X}, \mathbf{y}, \mathbf{X}_* \sim \mathcal{N}\left(\bar{\mathbf{f}}_*, \operatorname{cov}(\mathbf{f}_*)\right) \ ,$$

where

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

It's clear from above equations that the training data are explicitly used for the new test data to construct the prediction distribution [14], thus also proves that GP is a nonparametric model.

4 Illustrative example

In this section, a example implementation of the Gaussian processes regression is shown. The implementation follows the algorithm in [13] as follows.

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

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

The regression results are shown in Fig. 10. We do regression within the [-5, 5] domain. The observed data points (training dataset) are generated from a uniform distribution between -5 and 5. It means any point value within the given interval [-5, 5] is equally likely to be drawn by a uniform. The functions will be evaluated at evenly spaced points between -5 and 5. The function values compose the estimated mean function. Twenty samples of posterior mean functions were also plotted within 3μ variance.

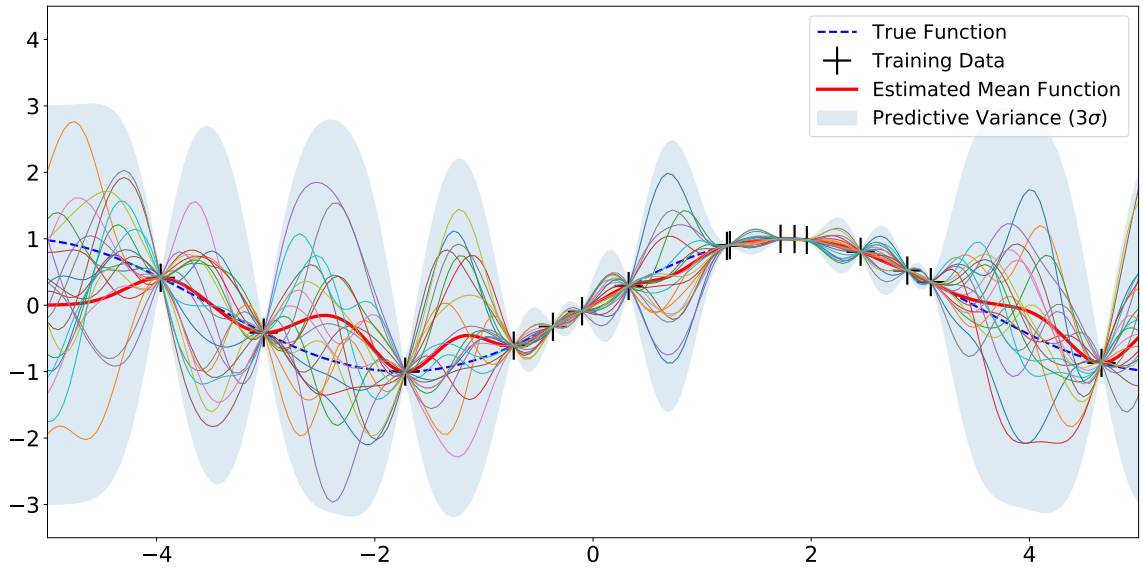


Figure 10: An example of Gaussian processes regression. The observed data points are generated by the blue dotted line (the true regression function) plotted as black crosses, using the generated data as the observed/training data points, infinite possible posterior functions are obtained. We plotted 20 samples of these infinite functions with sorted colors. The mean function is 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 3μ prediction variance.

4.1 Hyperparameters optimization

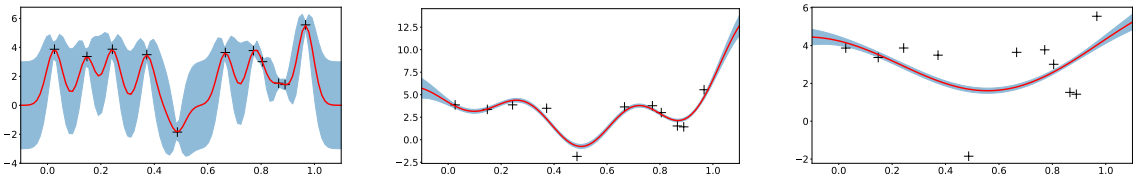
So far, we talked about the basics and implemented a simple GPR example. While a practical GPR model is more complex than it. The kernel functions play significant roles in GPR. Kernel functions are selected depending on the specific tasks using criteria including if the model is smooth, if it is sparse, if it can change drastically, and if it needs to be differentiable [4]. More information about choosing a kernel/covariance function for a Gaussian processes regression can be found in [4].

In kernels, proper hyper-parameters selection are essential. Let's use the most commonly used kernel, RBF, as an example. The general RBF function is

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

where σ_f and l are hyperparameters [4]. The vertical scale σ_f describes how much

vertically the function can span. The horizontal scale l indicate how quickly the correlation relationship between two points drops as their distance increase. A higher l provides a smooth function and smaller l results a more wiggly function. The function smoothness affected by l is shown in Fig. 11.



(a) l = small **(b)** l = medium **(c)** l = large

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

The optimized hyperparameters $\boldsymbol{\theta}$ are determined by the Maximum Likelihood Estimate as

$$\boldsymbol{\theta}^* = \arg\max_{\theta} \log p(y \mid \mathbf{X}, \boldsymbol{\theta}) \ .$$

Thus, considering hyperparameters optimization, a more general equation of predictions at the testing points is [1]

$$\bar{\mathbf{f}}_* \mid \mathbf{X}, \mathbf{y}, \mathbf{X}_*, \boldsymbol{\theta} \sim \mathcal{N}\left(\bar{\mathbf{f}}_*, \operatorname{cov}(\mathbf{f}_*)\right) \ .$$

The regression result of the data points in 11 with the optimized hyperparameters $\sigma_f = 0.0067$ and $l = 0.0967$ is shown in Fig. 12.

4.2 Gaussian processes packages

This section reviews three packages written in Python for Gaussian processes implementations. One of the most well-known GP packages is GPy [2]. GPy is a maturely developed package with well-documented explanations since 2012. GPy uses NumPy to perform all its computations. For tasks that don't require heavy computations implementations, GPy is sufficient and stable to use. However, GPR is computationally expensive in high dimensional spaces (features more than a few dozens) due to the fact it uses the whole samples/features to do the predictions. The more observations, the more computations are needed for predictions. A package that includes state-of-the-art algorithms is preferred for efficient implementation of complex GPR tasks. For GPR tasks that are computationally expensive, GPU acceleration is especially useful. GPflow [2] origins from GPy with

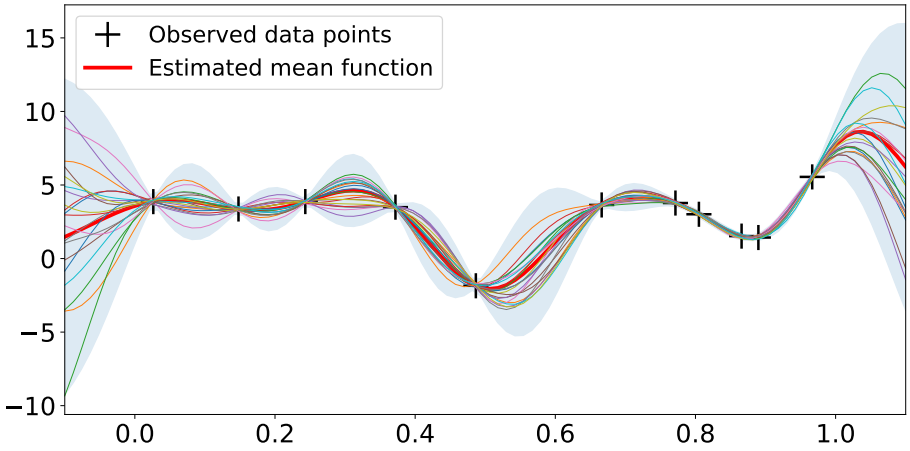


Figure 12: Regression result with the optimized hyperparameters σ_f and l.

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 contains very up-to-date GP algorithms. Similar to GPflow, GPyTorch provides automatic gradients. So complex models such as embedding deep neural networks in GP models can be easier developed.

5 Summary

A Gaussian processes model is a probability distribution over possible functions that fit a set of points [13]. Gaussian process regression provides a distribution with uncertainty estimations for predictions rather than one prediction value. The prior knowledge and specifications about the shape of the function model can be integrated by selecting different kernel functions.

The Gaussian processes format in this introduction is called the standard GP or vanilla GP [5]. There are two main constraints with standard Gaussian processes model: 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 [5].

A Appendix

The Marginal and conditional distributions of MVN theorem: suppose $X = (x_1, x_2)$ is jointly 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

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

and the posterior conditional is given by

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

References

[1] Zhenwen Dai. Gpss2019 - computationally efficient gps.

[2] Alexander G De G. Matthews, Mark Van Der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. Gpflow: A gaussian process library using tensorflow. *The Journal of Machine Learning Research*, 18(1):1299–1304, 2017.

[3] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, feb 2015.

[4] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[5] Roger Frigola, Fredrik Lindsten, Thomas B Schön, 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.

[6] R Fuentes, P Gardner, C Mineo, TJ Rogers, SG Pierce, K Worden, N Dervilis, and EJ Cross. Autonomous ultrasonic inspection using bayesian optimisation and robust outlier analysis. *Mechanical Systems and Signal Processing*, 145:106897, 2020.

[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] Agathe Girard, Carl Edward Rasmussen, Joaquin Quiñonero Qui~, Quiñonero Candela, and Roderick Murray-Smith. Gaussian Process Priors With Uncertain Inputs Application to Multiple-Step Ahead Time Series Forecasting. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, 2002.

[9] Lukas Hewing, Kim P. Wabersich, Marcel Menner, and Melanie N. Zeilinger. Learning-Based Model Predictive Control: Toward Safe Learning in Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1), may 2020.

[10] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. Learning-Based Model Predictive Control for Autonomous Racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, oct 2019.

[11] Kevin P Murphy. *Machine Learning A Probabilistic Perspective*. 2012.

[12] Chris J. Ostafew, Angela P. Schoellig, Timothy D. Barfoot, and Jack Collier. Learning-based Nonlinear Model Predictive Control to Improve Vision-based Mobile Robot Path Tracking. *Journal of Field Robotics*, 33(1):133–152, jan 2016.

[13] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. 2006.

[14] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2006.