GPR and Inverse Problem

Dr. Nono Saha

June 14, 2024

Max Planck Institute for Mathematics in the Sciences
University of Leipzig/ScaDS.AI
Lancaster University of Leipzig

Sommer 2024

Baysian Opt. and Gaussian Process

The goal of every optimization algorithm is to identify the minimum (or maximum) of an unknown objective function f in a certain design space $\chi \subset \mathbb{R}^d$,

$$\mathbf{x}_{min} = \arg\min_{\mathbf{x}\in\chi} f(\mathbf{x}) \tag{1}$$

Basic idea: treating the unknown objective as a random function, i.e. a stochastic model on a continuous domain χ.

A stochastic process $(X_x)_{x\in\chi}$ is a Gaussian process if for any N points $x_1^*, \cdots x_N^* \in \chi$ the probability of the objective to be equal to $Y = (y_1, \cdots, y_N)$ follows a multivariate Gaussian distribution

$$P(Y^*) = \frac{1}{(2\pi)^{N/2}|\mathbf{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{Y}-\mathbf{m})^T\mathbf{\Sigma}^{-1}(\mathbf{Y}-\mathbf{m})\right] \tag{2}$$

with a mean vector $\mathbf{m}$ and a covariance matrix $\mathbf{\Sigma}$.

Multivariate Gaussian Theorem [1]

Theorem (Marginals and conditionals of a MVN). Suppose $\mathbf{x} = (x_1, x_2)$ is jointly Gaussian with parameters.

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}, \boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{pmatrix}$$

Then, the marginals are given by

$$\begin{cases} p(x_1) = \mathcal{N}(x_1 | \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}) \\ p(x_2) = \mathcal{N}(x_2 | \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}) \end{cases}$$

and the posterior conditional is given by

$$p(x_1 | x_2) = \mathcal{N}(x_1 | \boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2})$$

where

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\Sigma}_{1|2}(\Lambda_{11}\boldsymbol{\mu}_1 - \Lambda_{12}(x_2 - \boldsymbol{\mu}_2))$$

and

$$\boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} = \Lambda_{11}^{-}1$$

[1] Christopher M. Bishop F.R.Eng, Pattern Recognition andMachine Learning, Spring 2006.

Gaussian Processes (GP)

GP is a Gaussian distribution over functions defined as:

$$f(\mathbf{x}) \sim GP(\mathbf{m}(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'))$$

where

$$\begin{cases} \mathbf{m}(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \\ \kappa(\mathbf{x}, \mathbf{x}') = \mathbb{E}\left[(f(\mathbf{x}) - \mathbf{m}(\mathbf{x}))(f(\mathbf{x}') - \mathbf{m}(\mathbf{x}'))^T\right] \end{cases}$$

How do you sample such functions?

1. given a set of points $\mathbf{X}_{1:N}$
2. Create $\mu = \mathbf{0}_N$ and compute $\mathbf{K} = \kappa(\mathbf{x}, \mathbf{x})$
3. find $\mathbf{L}$ such that $\mathbf{K} = \mathbf{L}\mathbf{L}^T$
4.

$$\begin{aligned} f^i &\sim \mathcal{N}(\mu, \mathbf{K}) \\ &\sim \mathcal{N}(\mu, \mathbf{I})\mathbf{L} \end{aligned}$$

Example: let $\kappa(x, x') = e^{\left(-\frac{1}{2}(x-x')^2\right)}$, write a python code that samples 10 functions.

Mean and Variance Function

In Bayesian methods, a prior is the initial belief about the function to be estimated before observing any data. This prior is updated when data is observed to get a posterior. With GPRs, the prior is defined over functions. Specifically, the prior consists of:

1. A mean function $\mathbf{m}(x)$: It is typically set to zero, but in principle, it can be set based on prior knowledge about the data.
2. A covariance function $\kappa(\mathbf{x}, \mathbf{x})$: The covariance function (or kernel function) represents the belief about the similarity or relationship between data points in the input space.

Remarks

- Once data is observed, the GPR only keeps functions that fit the data points
- This is the posterior, the prior updated with the observed data.
- For regression tasks, the mean function calculated by the posterior distribution of possible functions is the function used for predictions.

Kernel Functions

Kernel functions (denoted as $\kappa(\mathbf{x}, \mathbf{x})$), also known as covariance functions, are central to GPRs. They define the covariance or similarity between points in the input space, governing how function values at different points relate to one another. The kernel's choice affects the GP's ability to capture functions effectively.

1. Squared Exponential (SE) or Radial Basis Function (RBF) Kernel

$$\kappa(x, x') = \sigma^2 \exp\left(\frac{||x - x'||^2}{2l^2}\right)$$

2. Matern Kernel

$$\kappa(x, x') = \frac{1}{\Gamma(v)2^{v-1}} \left(\sqrt{2v}\frac{||x - x'||}{l}\right)^v K_v \left(\sqrt{2v}\frac{||x - x'||}{l}\right)$$

3. White Kernel

$$\kappa(x, x') = \begin{cases} \sigma^2 & \text{if } x == x' \\ 0 & \text{else} \end{cases}$$

GP given a set of observed points

Now, the Gaussian process 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 calculate the variance to show how confident we are when we make predictions using the function.

Keep in mind,

- The functions (posterior) update with new observations.
- The mean calculated by the posterior distribution of the possible functions is the function used for regression.

A multivariable Gaussian models the function as

$$p(\mathbf{f}|\mathbf{x}) = \mathcal{N}(\mathbf{f}|\mu, \mathbf{K})$$

So, we have observations and estimated functions f with these observations. Now, say we have some new points $\mathbf{x}_*$ where we want to predict $f(\mathbf{x}_*)$

GP given a set of observed points

The joint distribution of f and f^* can be modeled as:

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

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

This is modelling a joint distribution $p(\mathbf{f}, \mathbf{f}_*|\mathbf{X}, \mathbf{X}_*)$, but we want the conditional distribution over only, which is $p(\mathbf{f}_*|\mathbf{f}, \mathbf{X}, \mathbf{X}_*)$.

How can we derive the posterior from the joint distribution?

We obtain:

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

GP given a set of observed points

Now let us consider a more realistic situation where $y = f(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.

The prior on the noisy observations becomes $cov(y) = \mathbf{K} + \sigma_n^2 \mathbf{I}$.

The joint distribution of the observed target values and the function values at the test locations under the prior as

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

Deriving the conditional distribution corresponding to our previous equation, we get the following predictive equations

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

where,

$$\bar{\mathbf{f}}_* = \mathbb{E}\left[\bar{\mathbf{f}}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*\right] = \mathbf{K}_*^T \left[\mathbf{K} + \sigma_y^2 \mathbf{I}\right]^{-1} \mathbf{y}$$

$$cov(\bar{\mathbf{f}}_*) = \mathbf{K}_{**} - \mathbf{K}_*^T \left[\mathbf{K} + \sigma_y^2 \mathbf{I}\right]^{-1} \mathbf{K}_*$$

GPR Algorithm (Hipster version!)

Inputs: $\mathbf{X}$ (inputs), $\mathbf{y}$ (targets), κ (covariance function), σ_n^2 (noise level), $\mathbf{X}_*$ (test input)

1. $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X})$
2. $\mathbf{v} := (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$
3. $\mathbf{K}_* = \kappa(\mathbf{X}, \mathbf{X}_*)$
4. $\mathbb{E}[\bar{\mathbf{f}}_*] := \mathbf{K}_*^T \mathbf{v}$
5. $\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*)$
6. $\mathbf{v}_* := (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{K}_*$
7. $\text{cov}[\bar{\mathbf{f}}_*] = K_{**} - \mathbf{K}^T \mathbf{v}_*$
8. $\sigma_{\bar{\mathbf{f}}_*} = \sqrt{\text{diag}\left(\text{cov}[\bar{\mathbf{f}}_*]\right)}$

Outputs: $\mathbb{E}[\bar{\mathbf{f}}_*]$ (mean posterior), $\sigma_{\bar{\mathbf{f}}_*}$ (standard deviation)

GPR Algorithm (Improved version)[2]

Inputs: $\mathbf{X}$ (inputs), $\mathbf{y}$ (targets), κ (covariance function), σ_n^2 (noise level), $\mathbf{x}_*$ (test input)

1. $\mathbf{K} = \kappa(\mathbf{X}, \mathbf{X})$
2. $\mathbf{L} = \text{cholesky}(\mathbf{K} + \sigma_n^2 \mathbf{I})$
3. $\mathbf{m} = \mathbf{L}^{-1}\mathbf{y}$
4. $\mathbf{K}_* = \kappa(\mathbf{X}, \mathbf{X}_*)$
5. $\mathbf{L}_k := \mathbf{L}^{-1}\mathbf{K}_*$
6. $\mathbb{E}[f_*] := \mathbf{L}_k^T \mathbf{m}$
7. $\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*)$
8. $\text{var}[\bar{\mathbf{f}}_*] = \text{diag}\,(\mathbf{K}_{**}) - \sum_i L_k^2$
9. $\sigma_{\bar{\mathbf{f}}_*} = \sqrt{\text{var}[\bar{\mathbf{f}}_*]}$

Outputs: $\mathbb{E}[\bar{\mathbf{f}}_*]$ (mean posterior), $\sigma_{\bar{\mathbf{f}}_*}$ (standard deviation)

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

GP: Summary

- GPR is based on the Bayesian inference principle
- GPR provide not just the prediction function (as in standard regression methods) but also a measure of uncertainty, which is helpful with inverse design.
- The variance function calculated by the posterior distribution of possible functions quantifies the uncertainty of the predictions made by the GPR.
- The parameters of the kernel are hyperparameters to be tuned, e.g. σ, l
- What we have seen is know as **Standard/Plain Gaussian Process**. It has two main constraints:
  1. The overall computation complexity is $O(N^3)$, N, is the dimension of covariance matrix $\mathbf{K}$
  2. The memory consumption is quadratic
- The standard GP loses its efficiency quickly when there are more than 5000 for CPU and 13000 data points for GPU. For big datasets, **sparse GP** is needed.

RNA Inverse Problem

Given a target 3D RNA structure, $\mathcal{S}^*$ of a length L with a set of desired properties, the goal is to find one or many RNA sequences of length L that fold into the target structure $\mathcal{S}^*$. More formally, this is done by solving the following optimization problem:

$$\min_{\phi} I(\phi)$$

$$\text{where } I(\phi) = ||f(\phi) - \mathcal{S}^*|| \text{ and } \phi \in \{A, C, G, U\}^L$$

Remarks: A key prerequisite to addressing the RNA inverse problem is a reliable solution to the RNA 3D structure prediction problem, i.e., computing the function f.

Challenge: Traditionally, predicting a 3D structure of an RNA molecule is done using coarse-grained methods or Finite Element Methods (FEM), which are computationally costly and complex mathematical procedures.

Idea: Replace the function f by $\bar{f}_*$