POSTECH
INDUSTRIAL AND MANAGEMENT ENGINEERING, POSTECH

Scalable Gaussian process

포항공과대학교 산업경영공학과

Stochastic Systems Lab

Jongwon Kim

July 11, 2022

STOCHASTIC SYSTEMS LAB.

Contents

STOCHASTIC SYSTEMS LAB.

What is Gaussian process?

The Gaussian process is a set of random variables {f (x)|x ∈ X } for which any finite subset follows a Gaussian distribution. Gaussian process regression is a nonparametric regression, that is, no deterministic parametric form is assumed for **the relationship, f between input data X and output data Y**.

We want to estimate the relationship, f by Gaussian process regression which returns a normal distribution.

$$y = f(x) + \epsilon \; where \; \epsilon \sim N(0, \lambda^2 I)$$

We assume that y to be obscured by stochastic noise.

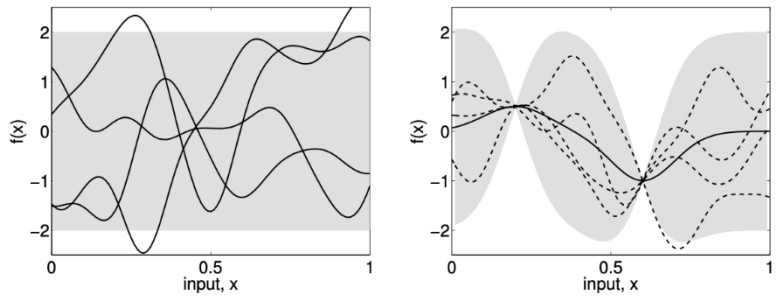


Figure 1.1: Panel (a) shows four samples drawn from the prior distribution. Panel (b) shows the situation after two datapoints have been observed. The mean prediction is shown as the solid line and four samples from the posterior are shown as dashed lines. In both plots the shaded region denotes twice the standard deviation at each input value x.

$$f([x_1, x_2, x_3])|X, Y \sim N(\mu, \Sigma)$$

where $\underline{\mu} = \begin{pmatrix} 4 \\ 2 \\ 6 \end{pmatrix}$ and $\sum = \begin{pmatrix} 16 & -1 & 0 \\ -1 & 4 & 2 \\ 0 & 2 & 9 \end{pmatrix}$

Degenerate Kernel

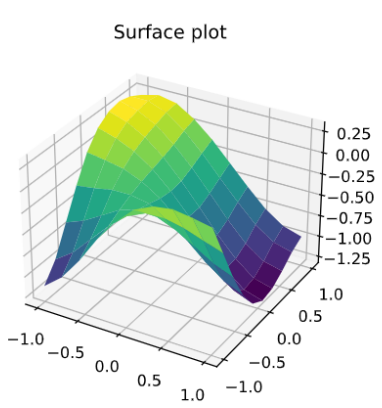
What is Gaussian process?

The Gaussian process is defined completely with Kernel function, K and mean function, m.

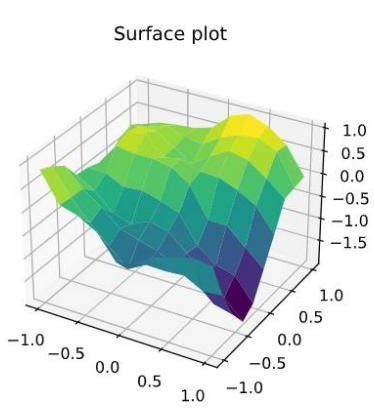
In many research, it is common to consider GPs with a zero mean function.

Kernel function, K represents the distance between $x_1, x_2 \in X$.

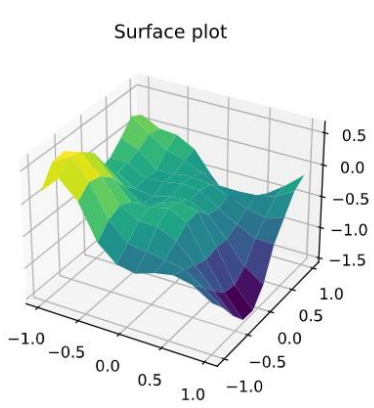
Ex) $K(x_1, x_2) = \sigma\, exp\left(-\frac{(x_1 - x_2)^2}{2l}\right)$ where σ, l are kernel's parameter



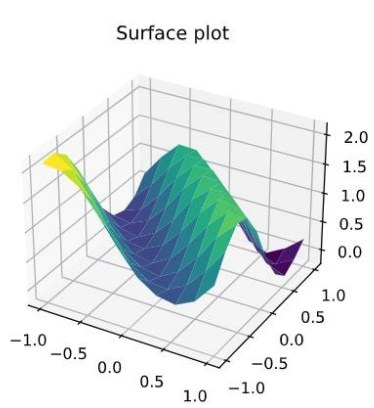
Gaussian Kernel



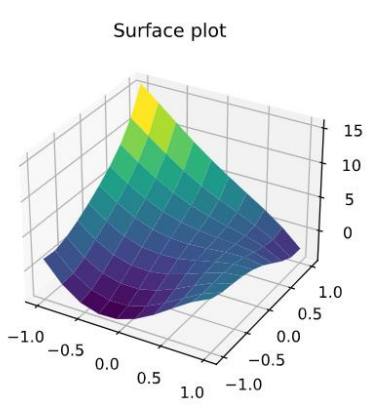
Matern 32 Kernel



Matern 52 Kernel



Gaussian Kernel with direction (1,1).



Polynomial Kernel

What is Gaussian process?

Problem statement

With observed data X, Y , we want to predict the function value $f(x')$ according to the x'.

Prediction model (Posterior model)

$$f(x')|x', X, Y \sim N(K(x', X)\left[\boldsymbol{K(X, X) + \lambda^2 I}\right]^{-1} Y, K(x', x') - K(x', X)\left[\boldsymbol{K(X, X) + \lambda^2 I}\right]^{-1} \mathrm{K(X, x')})$$

For large-sized data, Full-GP is hardly applicable due to the computational cost of $\left[\boldsymbol{K(X, X) + \lambda^2 I}\right]^{-1}$:
Time complexity: $O(n^3)$
Memory footprint complexity: $O(n^2)$ where n is number of data.

What is Gaussian process?

Problem statement

For simplicity, we will show how to calculate the $[\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})]^{-1}$ efficiently in terms of computation time.

→ **Scalable Gaussian process**

Scalable Gaussian process

1. Scalable Gaussian process
    1. Global approximation
    2. Local approximation
2. Using GPU for Gaussian process
    1. BBMM
    2. KeOps
3. Online Gaussian process

Scalable Gaussian process

1. Scalable Gaussian process
    1. Global approximation

        전체 데이터 중 일부를 선택하거나 대표 점들을 통해 전체를 표현하는 approximation방법이다.

Scalable Gaussian process

Global approximation

$$\mu' = K(x', X)[\boldsymbol{K(X, X)}]^{-1}Y = K_{x'X}(\boldsymbol{K_{XX}})^{-1}Y$$

1. Global approximation
    1. Subset-of-Data

        Select $n' < n$ **training point**, $X' \subset X$ as a training dataset

    2. Using inducing point

        Using $\mathrm{m} < n$ **inducing point**, U, we replace the exact kernel, K by Nystr̈om approximation.

$$(\boldsymbol{K_{XX}}) \approx K_{XU}K_{UU}K_{UX}$$

Scalable Gaussian process

Global approximation

$$\mu' = K(x', X)[\boldsymbol{K(X, X)}]^{-1}Y = K_{x'X}(\boldsymbol{K_{XX}})^{-1}Y$$

1. Global approximation

   3. Using Kronecker product

      Suppose that input data, X lie on a **Cartesian grid**

      $\boldsymbol{X = X^{(1)} \times \cdots \times X^{(d)}}$ where $\boldsymbol{X^{(i)}}$ represents the vector of X along dimension i

      Then we can calculate the inverse $\boldsymbol{K_{XX}^{-1}}$ quickly.

      $\boldsymbol{K_{XX} = K_{X^{(1)}X^{(1)}} \otimes \cdots \otimes K_{X^{(d)}X^{(d)}}}$ where $\otimes$ represents the Kronecker product $\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}$.

Scalable Gaussian process

Local approximation

1. Scalable Gaussian process

    2. Local approximation

        전체 데이터가 나눠진 여러 local 데이터로 각각의 local 모델을 학습하고 각 모델을 연결하는 approximation방법이다.

Scalable Gaussian process

Local approximation

$$\mu' = K(x', X)[\boldsymbol{K(X, X)}]^{-1}Y = K_{x'X}(\boldsymbol{K_{XX}})^{-1}Y$$

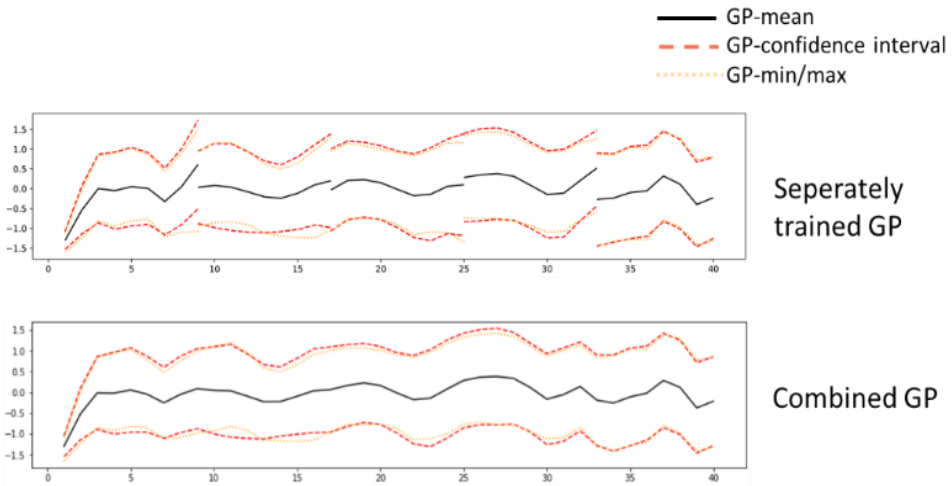
2. Local approximation

$$O(n^3) \rightarrow O\left(n_1^3 + \cdots + n_k^3\right) + O(*) \; where \sum n_i = n$$

   1. Patch Kriging

      Divide the domain into k local domain and learn the model for each domain.

      1. How to smooth the local expert for each overlapped domain.
      2. How to divide the domain into local domain



Scalable Gaussian process

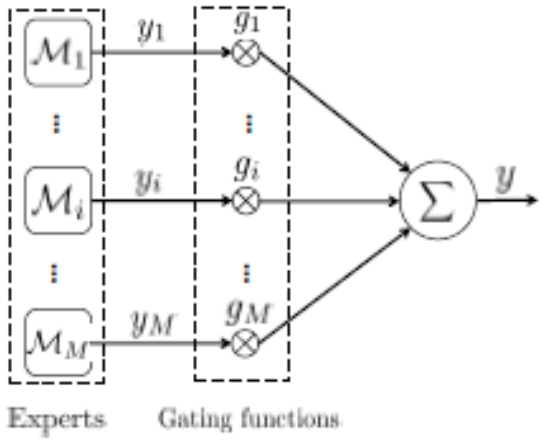
Local approximation

$$\mu' = K(x', X)[\boldsymbol{K(X, X)}]^{-1}Y = K_{x'X}(\boldsymbol{K_{XX}})^{-1}Y$$

2. Local approximation

    2. Mixture of Experts

        Joint learning of probabilistic partition of input space and diverse experts for different but overlapped sub-regions



$$p(y|\boldsymbol{x}) = \sum_{i=1}^{M} g_i(\boldsymbol{x}) p_i(y|\boldsymbol{x})$$

Gaussian process with GPU

Black-box Matrix multiplication

$$\mu' = K_{x'X}(\boldsymbol{K_{XX}})^{-1}Y$$

Acutally, we don't need the inverse, K^{-1}, we just need the matrix-vector multiplication $K^{-1}Y$

Conjugate gradients (CG) is an alternative method for computing $\widehat{K}_{XX}^{-1}\mathbf{y}$. CG frames $\widehat{K}_{XX}^{-1}\mathbf{y}$ as the solution to an optimization problem: $\mathbf{v}^* = \arg\min_{\mathbf{v}} \frac{1}{2}\mathbf{v}^\top \widehat{K}_{XX}\mathbf{v} - \mathbf{v}^\top \mathbf{y}$, which is convex by the positive-definiteness of $\widehat{K}_{XX}$. The optimization is performed iteratively, with each step requiring

Change the inversion problem into optimization problem and solve that problem by multiple matrix multiplication.

Matrix multiplication = GPU's work

Gardner, Jacob, et al. "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration." *Advances in neural information processing systems* 31 (NIPS), 2018.

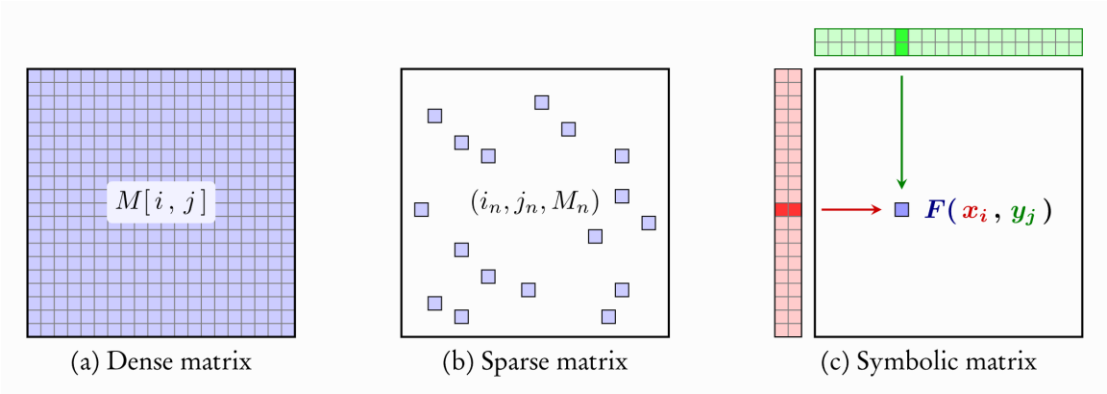
Gaussian process with GPU

KeOps

$$\mu' = K_{x'X}(\boldsymbol{K_{XX}})^{-1}Y$$

Three types of matrix:

- Dense matrix
- Sparse matrix
- Symbolic matrix
  - We can express the $K(X_1, X_2)$ with only three elements K, X_1, X_2.



Matrix type on GP framework

This paper use the idea that low Kolomogorov complexity of the dataset can improve model efficiency.

Charlier, B., Feydy, J., Glaunès, J. A., Collin, F.-D. & Durif, G. "Kernel Operations on the GPU, with Autodiff, without Memory Overflows". Journal of Machine Learning Research 22, 1–6 (JMLR), 2021.

Online Gaussian process

Online Gaussian process

1. JIT(Just-in-time) model

    Use whole history for training with short learning time.

2. Moving average

    Update the model or utilize (latest) k data point for training model.

Online Gaussian process

Online Gaussian process

Moving average

Utilize latest k data point for training model

1. Decide whether to include new data in the training dataset
2. Select k data that best describes all previous data

Update the model

1. Calculate the mean, likelihood, and variance of t+1 time using the previous ones.

Thank you

Appendix

- **Notation**

y $\in R$: Output data (Dependent variable)

x $\in A$: Input data (Independent variable)

$D = \{(x_i, y_i)|i = 1, \dots, n\}$: Data set of n observation,

w : weight of the function (weight space view)

$f : A \to R$ function which we want to optimize (function space view)

All vectors are column vectors.

Capital letters refer to matrix

- $\mathrm{X} = (x_1, x_2, \dots, x_n)^T \ where\ x_i \in \Omega$

Global approximation

FITC, PITC

$$\mu' = K(x', X)[\boldsymbol{K(X, X)}]^{-1}Y = K_{x'X}(\boldsymbol{K_{XX}})^{-1}Y$$

Using m **inducing point**, U, we replace the exact kernel, K by Nystr¨om approximation, Q.

$$\mathrm{Q_{ab}} = \mathrm{K_{au}K_{UU}^{-1}K_{ub}}$$

Common assumption

$$\mathrm{p}(y', Y|U) = \mathrm{p}(Y|U)\mathrm{p}(y'|U)$$

FITC

$$\mathrm{p}(y_1, y_2|U) = \mathrm{p}(y_1|U)\mathrm{p}(y_2|U) \;\; for\ any \;\; y_1, y_2 \in Y$$

$$\mu' = Q_{x'X}\left(\widehat{\boldsymbol{K_{XX}}}\right)^{-1}Y$$

PITC

$$\mathrm{p}(y_1, y_2|U) = \mathrm{p}(y_1|U)\mathrm{p}(y_2|U) \;\; for\ any \;\; y_1 \in Y_i, y_2 \in Y_j, i \neq j$$

$$\widehat{\boldsymbol{K_{XX}}} = \boldsymbol{Q_{XX}} + \boldsymbol{di}\mathrm{ag}(\mathrm{K_{XX}} - \mathrm{Q_{XX}})$$

$$\widehat{\boldsymbol{K_{XX}}} = \boldsymbol{Q_{XX}} + \boldsymbol{blockdi}\mathrm{ag}(\mathrm{K_{XX}} - \mathrm{Q_{XX}})$$

Global approximation

Variational free energy

$$\mu' = K(x', X)[\boldsymbol{K(X, X)}]^{-1}Y = K_{x'X}(\boldsymbol{K_{XX}})^{-1}Y$$

Using m inducing point, u, we replace the exact kernel by Nystr¨om approximation. $\mathrm{Q_{ab}} = \mathrm{K_{au}K_{uu}^{-1}K_{ub}}$

$$\mu' = Q_{x'X}\left(\widehat{\boldsymbol{K_{XX}}}\right)^{-1}Y$$

$$\widehat{\boldsymbol{K_{XX}}} = \boldsymbol{Q_{XX}} + \boldsymbol{di}\mathrm{ag}(\mathrm{K_{XX}} - \mathrm{Q_{XX}})$$

$$p(f_i, f_j|\boldsymbol{f_u}) = p(f_i|\boldsymbol{f_u})p(f_j|\boldsymbol{f_u}),$$

$$\widehat{\boldsymbol{K_{XX}}} = \boldsymbol{Q_{XX}} + \boldsymbol{blockdi}\mathrm{ag}(\mathrm{K_{XX}} - \mathrm{Q_{XX}})$$

Time complexity: $\mathrm{O}(\mathrm{nm}^2)$, Memory complexity: $\mathrm{O}(\mathrm{nm}^2)$

Appendix

Appendix

Appendix

Thank you