Gaussian Process Flavours

Davide Carminati

1 Kernel Eigenvalue Decomposition

The kernel series approximation relies on *Mercer's expansion*:

$$K_{(\mathbf{x},\mathbf{x}')} = \sum_{n=0}^{\infty} \lambda_n \varphi_n(\mathbf{x}) \varphi_n(\mathbf{x}') \tag{1}$$

in which λ_n and φ_n are the *n-th* eingenvalue and eigenfunction, respectively. Using a finite series approximation, a Gaussian Process kernel can be written as:

$$K_{(\mathbf{x},\mathbf{x}')} \approx \sum_{n=0}^{m} \lambda_n \varphi_n(\mathbf{x}) \varphi_n(\mathbf{x}') \tag{2}$$

where m is the number of eigenvalues considered. m is much smaller than the number of datapoints $\mathbf{x}_1 \dots \mathbf{x}_N$. The advantage is that we consider a much smaller kernel $K_{approx} \in \mathrm{R}^{m \times m}$ to be inverted, instead of the classical $K \in \mathrm{R}^{N \times N}$.

1.1 Squared Exponential kernel decomposition

The *Squared Exponential* kernel is defined as:

$$K_{(x,x')} = \exp^{-\varepsilon^2 (x-x')^2} \tag{3}$$

Following [1], we set:

$$\beta = \left(1 + \left(\frac{2\varepsilon}{\alpha}\right)^2\right)^{\frac{1}{4}}, \quad \gamma_n = \sqrt{\frac{\beta}{2^{n-1}\Gamma_{(n)}}}, \quad \delta^2 = \frac{\alpha}{2}(\beta^2 - 1), \tag{4}$$

the *Squared Exponential* kernel can be decomposed using Equation 1 and one obtains the eigenfunctions:

$$\varphi_n(x) = \gamma_n \exp^{-\delta^2 x^2} H_{n-1}(\alpha \beta x) \tag{5}$$

where H_{n-1} is the *classical* Hermite polynomial of degree $n-1$. Their corresponding *eigenvalues* are defined as:

$$\lambda_n = \sqrt{\frac{\alpha^2}{\alpha^2 + \delta^2 + \varepsilon^2}} \left(\frac{\varepsilon^2}{\alpha^2 + \delta^2 + \varepsilon^2}\right)^{n-1} \tag{6}$$

1.2 Multivariate expansion

The d-variate squared exponential kernel is defined as:

$$K_{(\mathbf{x},\mathbf{x}')} = \exp^{-\varepsilon_1^2 (x_1 - x_1')^2 - \dots - \varepsilon_d^2 (x_d - x_d')^2} \tag{7}$$

For d-variate kernels it holds the following expansion:

$$K_{(\mathbf{x},\mathbf{x}')} = \sum_{\mathbf{n} \in \mathbb{N}^d} \lambda_{\mathbf{n}} \varphi_{\mathbf{n}}(x) \varphi_{\mathbf{n}}(x') \tag{8}$$

where $\mathbf{n}$ is the set of all n^d combination of the considered number of eigenvalues. The eigenvalues $\lambda_{\mathbf{n}}$ and eigenfunctions $\varphi_{\mathbf{n}}(\mathbf{x})$ are defined as [1]:

$$\lambda_{\mathbf{n}} = \prod_{j=1}^{d} \lambda_{n_j} \tag{9}$$

$$\varphi_{\mathbf{n}}(\mathbf{x}) = \prod_{j=1}^{d} \varphi_{n_j}(x_j) \tag{10}$$

where d is the number of dimensions.

2 Gaussian Process Implicit Surface

Gaussian Process Implicit Surface (GPIS) [2][3] allows the modeling of obstacles in environments by imposing a suitable y value to the regression problem. In particular, each $\mathbf{x}$ point has a value

$$y = \begin{cases} -1 & \text{outside obstacle} \\ 0 & \text{on the edge} \\ 1 & \text{inside obstacle} \end{cases} \tag{11}$$

Given a train dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, with $\mathbf{x}_i \in \mathbf{X}$ and $y_i \in \mathbf{y}$ and a test dataset $\{(\mathbf{x}_{*i}, y_{*i})\}_{i=1}^{M}$, with $\mathbf{x}_{*i} \in \mathbf{X}_*$, the kernels are written as:

$$\mathbf{K} = K(\mathbf{X}, \mathbf{X}), \quad \mathbf{k}_* = K(\mathbf{X}_*, \mathbf{X}), \quad \mathbf{k}_{**} = K(\mathbf{X}_*, \mathbf{X}_*) \tag{12}$$

The GP regression problem is as follows:

$$\begin{aligned} \bar{\mathbf{f}}_* &= \mathbf{k}_* [\mathbf{K} + \sigma_N^2 \mathrm{I}]^{-1} \mathbf{y} \\ \mathrm{cov}(\mathbf{f}_*) &= \mathbf{k}_{**} - \mathbf{k}_* [\mathbf{K} + \sigma_N^2 \mathrm{I}]^{-1} \mathbf{k}_*^\top \end{aligned} \tag{13}$$

where $\bar{\mathbf{f}}_*$ is the prediction on the test dataset and $\mathrm{cov}(\mathbf{f}_*)$ is the associated uncertainty on the prediction. In the classic GPIS algorithms, the *Square Exponential* kernel (Equation 14) and the *Thin Plate* [3] kernel (Equation 15) are used.

$$K_{SE(\mathbf{x},\mathbf{x}')} = \sigma^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|}{2l^2}\right) \tag{14}$$

$$K_{(\mathbf{x},\mathbf{x}')} = 2\|\mathbf{x} - \mathbf{x}'\|^3 - 3R\|\mathbf{x} - \mathbf{x}'\|^2 + R^3 \tag{15}$$

where l is the kernel *length scale* and R is the maximum distance between datapoints.

3 GP with gradient information

In a GP regression problem, it is possible to provide for each d-dimensional point $\mathbf{x} \in \mathrm{R}^d$ both the observation

y and its derivative ∇y [4]. Let $\mathbf{y}^+ \in \mathrm{R}^{N(d+1)}$ be the vector of all observations and their derivatives organized as:

$$\mathbf{y}^+ = \begin{bmatrix} \mathbf{y} \\ \nabla\mathbf{y} \end{bmatrix}, \qquad \nabla\mathbf{y} = \begin{bmatrix} \nabla_{x_1} y_1 \\ \vdots \\ \nabla_{x_1} y_N \\ \vdots \\ \nabla_{x_d} y_1 \\ \vdots \\ \nabla_{x_d} y_N \end{bmatrix}_{Nd\times 1} \tag{16}$$

To describe the covariance between observations and their gradients, an augmented kernel $K^+_{(\mathbf{x},\mathbf{x}')} \in \mathrm{R}^{N(d+1)\times N(d+1)}$ with gradients is used:

$$K^+_{(\mathbf{x},\mathbf{x}')} = \begin{bmatrix} K_{(\mathbf{x},\mathbf{x}')} & K_{(\mathbf{x},\mathbf{x}')}\nabla^\top_{\mathbf{x}'} \\ \nabla_{\mathbf{x}} K_{(\mathbf{x},\mathbf{x}')} & \nabla_{\mathbf{x}} K_{(\mathbf{x},\mathbf{x}')}\nabla^\top_{\mathbf{x}'} \end{bmatrix} \tag{17}$$

where $\nabla_{\mathbf{x}} = \begin{bmatrix} \frac{\partial}{\partial x_1} \dots \frac{\partial}{\partial x_d} \end{bmatrix}^\top$.
As a consequence, the GP regression problem can be rearranged from Equation 13 as:

$$\begin{aligned} \begin{bmatrix} \bar{\mathbf{f}}_* \\ \nabla\bar{\mathbf{f}}_* \end{bmatrix} &= \mathbf{k}^+_*[\mathbf{K}^+ + \sigma^2_{N(d+1)}\mathrm{I}]^{-1}\mathbf{y}^+ \\ \begin{bmatrix} \mathrm{cov}(\mathbf{f}_*) \\ \nabla\mathrm{cov}(\mathbf{f}_*) \end{bmatrix} &= \mathbf{k}^+_{**} - \mathbf{k}^+_*[\mathbf{K}^+ + \sigma^2_{N(d+1)}\mathrm{I}]^{-1}\left(\mathbf{k}^+_*\right)^\top \end{aligned} \tag{18}$$

For the 2D case, the augmented covariance matrix turns out to be:

$$K^+_{(\mathbf{x},\mathbf{x}')} = \begin{bmatrix} K_{(\mathbf{x},\mathbf{x}')} & \frac{\partial K_{(\mathbf{x},\mathbf{x}')}}{\partial x'_1} & \frac{\partial K_{(\mathbf{x},\mathbf{x}')}}{\partial x'_2} \\ \frac{\partial K_{(\mathbf{x},\mathbf{x}')}}{\partial x_1} & \frac{\partial^2 K_{(\mathbf{x},\mathbf{x}')}}{\partial x_1 \partial x'_1} & \frac{\partial^2 K_{(\mathbf{x},\mathbf{x}')}}{\partial x_1 \partial x'_2} \\ \frac{\partial K_{(\mathbf{x},\mathbf{x}')}}{\partial x_2} & \frac{\partial^2 K_{(\mathbf{x},\mathbf{x}')}}{\partial x_2 \partial x'_1} & \frac{\partial^2 K_{(\mathbf{x},\mathbf{x}')}}{\partial x_2 \partial x'_2} \end{bmatrix} \tag{19}$$

3.1 Kernel derivatives

A list of kernel partial derivatives is provided.

3.1.1 Squared Exponential covariance function

The Squared Exponential kernel is presented in Equation 14. Differentiation leads to the following expressions [2]:

$$\frac{\partial K_{SE(\mathbf{x},\mathbf{x}')}}{\partial x_i} = -\frac{x_i - x'_i}{l^2} K_{SE(\mathbf{x},\mathbf{x}')} \tag{20}$$

$$\frac{\partial^2 K_{SE(\mathbf{x},\mathbf{x}')}}{\partial x_i \partial x_j} = \left(\frac{1}{l^2}\delta_{ij} - \frac{(x_i - x'_i)(x_j - x'_j)}{l^4}\right) K_{SE(\mathbf{x},\mathbf{x}')} \tag{21}$$

3.1.2 Thin Plate covariance function

Deriving Equation 15 with respect to each component of $\mathbf{x}$, one obtains [2]:

$$\frac{\partial K_{TP(\mathbf{x},\mathbf{x}')}}{\partial x_i} = 6(x_i - x'_i)\left(\|\mathbf{x} - \mathbf{x}'\| - R\right) \tag{22}$$

$$\frac{\partial K_{TP(\mathbf{x},\mathbf{x}')}}{\partial x_i \partial x_j} = -6\left(\frac{(x_i - x'_i)(x_j - x'_j)}{\|\mathbf{x} - \mathbf{x}'\|} + \delta_{ij}(\|\mathbf{x} - \mathbf{x}'\| - R)\right) \tag{23}$$

3.1.3 Matérn 5/2 covariance function

Setting $d = \|\mathbf{x} - \mathbf{x}'\|$, the *Matérn* $\frac{5}{2}$ covariance function is of the form:

$$K_{M(\mathbf{x},\mathbf{x}')} = \left(1 + \frac{\sqrt{5}}{l}d + \frac{5}{3l^2}d^2\right)\exp\left(-\frac{\sqrt{5}}{l}d\right) \tag{24}$$

Its partial derivatives are:

$$\begin{aligned} \frac{\partial K_{M(\mathbf{x},\mathbf{x}')}}{\partial x_i} &= \exp\left(-\frac{\sqrt{5}}{l}d\right)\cdot \\ &\quad \cdot\left[\frac{5}{l^2}(x_i - x'_i)\left(\left(\frac{2}{3} - \frac{\sqrt{5}}{3l}\right)d - 1\right)\right] \end{aligned} \tag{25}$$

$$\begin{aligned} \frac{\partial^2 K_{M(\mathbf{x},\mathbf{x}')}}{\partial x_i \partial x_j} &= \left(\frac{\sqrt{5}}{l}\frac{x_j - x'_j}{d}\right)\frac{\partial K_{M(\mathbf{x},\mathbf{x}')}}{\partial x_i} + \\ &+ \exp\left(-\frac{\sqrt{5}}{l}d\right)\left[\frac{(x_i - x'_i)(x_j - x'_j)}{d}\left(\frac{5\sqrt{5}}{3l^3} - \frac{10}{3l^2}\right) + \right. \\ &\left. + \delta_{ij}\left(\frac{5\sqrt{5}}{3l^3}d - \frac{10}{3l^2}d + \frac{5}{l^2}\right)\right] \end{aligned} \tag{26}$$

4 Fast Approximate GPIS

The Fast Approximate GPIS algorithm [5] relies on the Kernel Decomposition algorithm presented in section 1, which allows to rewrite the GP problem with an approximated – but smaller – kernel matrix. The advantage is that the kernel matrix inversion is faster.
Regrouping the terms in Equation 2, one obtains:

$$K_{(\mathbf{x},\mathbf{x}')} \approx \mathbf{\Phi}_{(\mathbf{x})}\Lambda\mathbf{\Phi}^\top_{(\mathbf{x}')} \tag{27}$$

where:

$$\mathbf{\Phi} = \begin{bmatrix} \varphi_1(\mathbf{x_1}) & \dots & \varphi_m(\mathbf{x_1}) \\ \vdots & & \vdots \\ \varphi_1(\mathbf{x_N}) & \dots & \varphi_m(\mathbf{x_N}) \end{bmatrix} \quad \Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{bmatrix}, \tag{28}$$

with N the number of datapoints, m the number of the considered eigenvalues. In this implementation, φ is defined in Equation 5. Substituting Equation 27 in Equation 13 and rewriting the terms using the *binomial inverse theorem*, the GP regression problem can be approximated as:

$$\begin{aligned} \bar{\mathbf{f}}_* &\approx \mathbf{W}\mathbf{y} \\ \mathrm{cov}(\mathbf{f}_*) &\approx \mathbf{\Phi}_{(\mathbf{x}_*)}\Lambda\mathbf{\Phi}^\top_{(\mathbf{x}_*)} - \mathbf{W}\mathbf{\Phi}_{(\mathbf{x})}\Lambda\mathbf{\Phi}^\top_{(\mathbf{x}_*)} \end{aligned} \tag{29}$$

in which

$$\begin{aligned} \mathbf{W} &= \mathbf{\Phi}_{(\mathbf{x}_*)}\Lambda\mathbf{\Phi}^\top_{(\mathbf{x})}\left(\Sigma_N^{-1} - \Sigma_N^{-1}\mathbf{\Phi}_{(\mathbf{x})}\bar{\mathbf{\Lambda}}^{-1}\mathbf{\Phi}^\top_{(\mathbf{x})}\Sigma_N^{-1}\right) \\ \bar{\mathbf{\Lambda}} &= \Lambda^{-1} + \mathbf{\Phi}^\top_{(\mathbf{x})}\Sigma_N^{-1}\mathbf{\Phi}_{(\mathbf{x})} \end{aligned}$$

Note that the regression problem needs the inversion of $\bar{\mathbf{\Lambda}}$, which is a $m \times m$ matrix. Since $m << N$, the matrix inversion is computationally less demanding with respect to the classical GP problem.

5 Logarithmic GPIS

Logarithmic GPIS [6] allows accurate description of the distance from objects. In this implementation, also the gradient information is considered in the LogGPIS problem. Contrarily to what happens for the standard GPIS problem presented in section 2, in LogGPIS only the information of the obstacle border and its gradient are needed. A train dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, with $\mathbf{x}_i \in \mathbf{X}$ and $y_i \in \mathbf{y}$ and a test dataset $\{(\mathbf{x}_{*i}, y_{*i})\}_{i=1}^{M}$, with $\mathbf{x}_{*i} \in \mathbf{X}_*$ are given. For the 2D case, the observation vector at each $\mathbf{x}_i = [x_1 \;\; x_2]$ is defined as:

$$\mathbf{y} = \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^{\top}_{N\times 1} \tag{30}$$

and the augmented observation vector is:

$$\mathbf{y}^{+} = \begin{bmatrix} \mathbf{y} \\ \nabla\mathbf{y} \end{bmatrix}_{3N\times 1} \qquad \nabla\mathbf{y} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} \\ \vdots \\ \frac{\partial y_N}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} \\ \vdots \\ \frac{\partial y_N}{\partial x_2} \end{bmatrix}_{2N\times 1}$$

Matérn 5/2 kernel (Equation 24) is used, as it shows higher distance accuracy and it is two-times differentiable. The augmented kernel $K^{+}_{(\mathbf{x},\mathbf{x}')}$ has the same structure as Equation 19.
To obtain the distance for every test points in the test dataset $\mathbf{X}_*$, the following transfomations are needed:

$$\bar{\mathbf{d}}_* = -\frac{\ln(\bar{\mathbf{f}}_*)}{\sqrt{5}/l}, \quad \nabla\bar{\mathbf{d}}_* = -\frac{1}{\sqrt{5}/l}\frac{\nabla\bar{\mathbf{f}}_*}{\bar{\mathbf{f}}_*} \tag{31}$$

where $\bar{\mathbf{f}}_*$ and $\nabla\bar{\mathbf{f}}_*$ are defined in Equation 18. The covariance turns out to be:

$$\mathrm{cov}(\mathbf{d}_*) = \frac{l}{\sqrt{5}\,\bar{\mathbf{f}}_*}\mathrm{cov}(\mathbf{f}_*)\frac{l}{\sqrt{5}\,\bar{\mathbf{f}}_*}^{\top} \tag{32}$$

References

[1] G. E. Fasshauer and M. J. McCourt, "Stable evaluation of gaussian radial basis function interpolants," *SIAM Journal on Scientific Computing*, vol. 34, no. 2, pp. A737–A762, 2012.

[2] W. Martens, Y. Poffet, P. R. Soria, R. Fitch, and S. Sukkarieh, "Geometric Priors for Gaussian Process Implicit Surfaces," *IEEE Robotics and Automation Letters*, vol. 2, pp. 373–380, Apr. 2017.

[3] O. Williams and A. Fitzgibbon, "Gaussian process implicit surfaces," in *Gaussian Processes in Practice*, 2006.

[4] A. Wu, M. C. Aoi, and J. W. Pillow, "Exploiting gradients and Hessians in Bayesian optimization and Bayesian quadrature," *arXiv:1704.00060 [stat]*, Mar. 2018. arXiv: 1704.00060.

[5] V. Joukov and D. Kulić, "Fast Approximate Multi-output Gaussian Processes," *arXiv:2008.09848 [cs, stat]*, Aug. 2020. arXiv: 2008.09848.

[6] L. Wu, K. M. B. Lee, L. Liu, and T. Vidal-Calleja, "Faithful Euclidean Distance Field from Log-Gaussian Process Implicit Surfaces," *arXiv:2010.11487 [cs]*, Jan. 2021. arXiv: 2010.11487.