Gaussian Process Flavours

Davide Carminati

1 Kernel Eigenvalue Decomposition

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

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

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:

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

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:

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

Setting:

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

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:

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

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

where d is the number of dimensions. For the 2D case, $\mathbf{x} \in \mathrm{R}^{N \times d}$, $\varphi_{\mathbf{n}} \in \mathrm{R}^N$.

2 Gaussian Process Implicit Surface

Gaussian Process Implicit Surface (GPIS) 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:

$$\bar{\mathbf{f}}_* = \mathbf{k}_* [\mathbf{K} + \sigma_n^2 \mathrm{I}]^{-1} \mathbf{y} \tag{13}$$

$$\mathrm{cov}(\mathbf{f}_*) = \mathbf{k}_{**} - \mathbf{k}_* [\mathbf{K} + \sigma_n^2 \mathrm{I}]^{-1} \mathbf{k}_*^T \tag{14}$$

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 15) and the *Thin Plate* kernel (Equation 16) are used.

$$K_{(x,x')} = \exp^{\frac{(x-x')^2}{2l^2}} \tag{15}$$

$$K_{(x,x')} = 2\|x - x'\|^3 - 3R\|x - x'\|^2 + R^3 \tag{16}$$

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