Gaussian Processes

Andrea Casalino

June 17, 2022

1 What is a Gaussian Process?

Gaussian Processes [3], [1], a.k.a. **GP**s, are data driven probabiistic models able to approximate generic multivariate and possibly vectorial real functions G defined like that:

$$G : \mathcal{I} \subseteq \mathbb{R}^i \rightarrow \mathcal{O} \subseteq \mathbb{R}^o \tag{1}$$

In essence, **GP**s are defined by their own **Training Set** and **Kernel Function**. The **Training Set** is a collection of points pertaining to $\mathcal{I}$, for which the corresponding output, or at least thst value summed with noise, inside $\mathcal{O}$ is known.
The **Kernel Function** is something that has to be chosen and typical of the kind of function to approximate. Definitions and meanings of possible **Kernel Function** are extensively detailed in Section 4.
The aim of **GP**s is to be able to predict the value of G for a point inside $\mathcal{I}$ that is not inside of the trainin set. In particular, this is done in probabilistic terms, as the result of the prediction is not just a value, but a conditioned Gaussian distribution.
Section 2 discusses the formulation of scalar **GP**, i.e. cases for which $\mathcal{O} \subseteq \mathbb{R}$. Insted, Section 3 focuses on the more general cases. The reader will notice that the two formulations are not in contradiction and the decision to discuss them into separate Sections is only for the purpose of a better readability.

2 Scalar case

The training set of a scalar **GP** is a collection of points inside $\mathcal{I}$ for which the corresponding value inside $\mathcal{O}$ is known. More formally:

$$S = \left\{ \begin{bmatrix} X_1 \\ \downarrow \\ y_1 \end{bmatrix}, \ldots, \begin{bmatrix} X_N \\ \downarrow \\ y_N \end{bmatrix} \right\} \tag{2}$$

$$\{X_1, \ldots, X_N\} \subset \mathbb{R}^i \tag{3}$$

$$\{y_1, \ldots, y_N\} \subset \mathbb{R} \tag{4}$$

GPs consider values inside the training set to be somehow correlated, as they were generated from the same underlying function. In particular, the joint probability distribution describing such correlation is assumed to be the following 0 mean **Gaussian Distribution**:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \sim \mathcal{N}\Big(0, K\big(X_{1,\ldots,N}\big)\Big) \tag{5}$$

K is a covariance matrix induced by the choice of a certain kernel function

k (refer also to Section 4):

$$K = \begin{bmatrix} k(X_1, X_1, \Theta) & \dots & k(X_1, X_N, \Theta) \\ \vdots & \ddots & \vdots \\ k(X_N, X_1, \Theta) & \dots & k(X_N, X_N, \Theta) \end{bmatrix} \tag{6}$$

where Θ is a vector of hyperparameters that are typical of the chosen kernel function and whose values can be tuned also by training (see Sections 2.2 and 3.2):

$$\Theta = \begin{bmatrix} \theta_1 & \dots & \theta_m \end{bmatrix}^T \tag{7}$$

It is generally a good practice to add to the covariance K an additional term modelling noise in the traning set. This is done by simply summing an isotropic standard deviation σ_{noise}:

$$K^{'} = K + \sigma_{noise} I_{N \times N} \tag{8}$$

2.1 Predictions

The aim of a **GP** is to be able to make predictions about the output value $y = G(X)$ pertaining o an input X that is outside of the training set. This is done assuming again a joint Gaussian correlation between such an additional point and all the ones in the training set:

$$\left[\begin{array}{c} y = G(X) \\ \hline \begin{matrix} y_1 \\ \vdots \\ y_N \end{matrix} \end{array}\right] \sim \mathcal{N}\left(\left[\begin{array}{c} 0 \\ \hline 0 \end{array}\right], \left[\begin{array}{c|c} k(X, X, \Theta) & K_x(X, X_{1,\dots,N}, \Theta) \\ \hline K_x(X, X_{1,\dots,N}, \Theta)^T & K \end{array}\right] \right) \tag{9}$$

where K_x is a vector assembled in this way:

$$K_x(X, X_{1,\dots,N}, \Theta) = \begin{bmatrix} k(X, X_1, \Theta) & \dots & k(X, X_N, \Theta) \end{bmatrix}^T \tag{10}$$

Since eq. 10 describes a Gaussian distibution, the conditioned distribution involving only X can be obtained as follows:

$$y(X|X_{1,\dots,N}) \sim \mathcal{N}\left(K_x^T K^{-1} \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \sigma_K(X) \right) \tag{11}$$

where $\sigma_K(X)$ is the covariance of the conditioned distribution and can be computed as follows:

$$\sigma_K(X) = k(X, X) - K_x^T K^{-1} K_x \tag{12}$$

The distribution described by eq. 11, actually represents the prediction made by the **GP**. Eq. 11 can also be rewritten as follows:

$$y(X|X_{1,\dots,N}) \sim \mathcal{N}\left(\begin{bmatrix} y_1 & \dots & y_N \end{bmatrix} K^{-1} K_x, \sigma_K(X) \right) \tag{13}$$

2.2 Training

Training is done maximizing the likelihood L of the training set w.r.t. the hyperparameters Θ of the kernel function. Since eq. (5) describes a **Gaussian** distribution, the likelihood can be computed as follows:

$$L = \frac{1}{\sqrt{(2\pi)^N \, |K|}} exp\Big(-\frac{1}{2} \begin{bmatrix} y_1 & \dots & y_N \end{bmatrix} K^{-1} \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \Big) \tag{14}$$

At this point, the property described in appendix A can be exploited to rewrite the above equation as follows:

$$\begin{aligned} L &= \frac{1}{\sqrt{(2\pi)^N \, |K|}} exp\Big(-\frac{1}{2} Tr\Big[K^{-1} \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \begin{bmatrix} y_1 & \dots & y_N \end{bmatrix} \Big] \Big) & (15) \\ &= \frac{1}{\sqrt{(2\pi)^N \, |K|}} exp\Big(-\frac{1}{2} Tr\Big[K^{-1} M_{YY} \Big] \Big) & (16) \end{aligned}$$

Passing to the logarithm we obtain:

$$\mathcal{L} = log(L) = -\frac{N}{2}(2\pi) - \frac{1}{2} log\big(\, |K| \, \big) - \frac{1}{2} Tr\Big[K^{-1} M_{YY} \Big] \tag{17}$$

keeping in mind that $\mathcal{L}$ is a function of the hyperparameters Θ:

$$\mathcal{L}(\Theta) = -\frac{N}{2}(2\pi) - \frac{1}{2} log\big(\, |K(\Theta)| \, \big) - \frac{1}{2} Tr\Big[K(\Theta)^{-1} M_{YY} \Big] \tag{18}$$

The gradient of $\mathcal{L}$ w.r.t. the generic hyperparameter θ_t is computed as follows (refer to the properties detailed at [2]):

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta_t} &= -\frac{1}{2} Tr\Big[K^{-1} \frac{\partial K}{\partial \theta_t} \Big] - \frac{1}{2} Tr\Big[\frac{\partial}{\partial \theta_t} \big(K^{-1} M_{YY} \big) \Big] & (19) \\ &= -\frac{1}{2} Tr\Big[K^{-1} \frac{\partial K}{\partial \theta_t} \Big] - \frac{1}{2} Tr\Big[\frac{\partial (K^{-1})}{\partial \theta_t} M_{YY} \Big] & (20) \\ &= -\frac{1}{2} Tr\Big[K^{-1} \frac{\partial K}{\partial \theta_t} \Big] + \frac{1}{2} Tr\Big[K^{-1} \frac{\partial K}{\partial \theta_t} K^{-1} M_{YY} \Big] & (21) \\ &= \frac{1}{2} Tr\Big[K^{-1} \frac{\partial K}{\partial \theta_t} \Big(K^{-1} M_{YY} - I_{N \times N} \Big) \Big] & (22) \end{aligned}$$

Choosing your favourite gradient-based approach you can tune the model, byt computing the gradient as described by the above equation.

Check Section 3.2.1 to understand how priors about the hyperparameters Θ can be handled.

3 Vectorial case

Vectorial **GP**s are defined as similarly done for the scalar case detailed in the previous Section. The training set should now account for the multi-dimensionality of the process and is therefore defined in this way:

$$S = \left\{ \begin{bmatrix} X_1 \\ \downarrow \\ Y_1 \end{bmatrix}, \ldots, \begin{bmatrix} X_N \\ \downarrow \\ Y_N \end{bmatrix} \right\} \tag{23}$$

$$\{X_1, \ldots, X_N\} \subset \mathbb{R}^i \tag{24}$$

$$\{Y_1, \ldots, Y_N\} \subset \mathbb{R}^o \tag{25}$$

The generic Y_k is a vector made of o components:

$$Y_k = \begin{bmatrix} y_k^1 & \ldots & y_k^o \end{bmatrix}^T \tag{26}$$

3.1 Predictions

A vectorial **GP** is actually a composition of independent scalar **GP**s. The prediction is done as similarly discussed in Section 2.1, doing o predictions at the same time. Indeed, for each component $y^{k \in \{1,\ldots,o\}}$ of the prediction holds equation 11. Therefore, the complete prediction can be obtained in this way:

$$Y(X|X_{1,\ldots,N}) = \begin{bmatrix} \mathcal{N}\Big(\begin{bmatrix} y_1^1 & \ldots & y_N^1 \end{bmatrix} K^{-1} K_x, \sigma_K(X) \Big) \\ \vdots \\ \mathcal{N}\Big(\begin{bmatrix} y_1^o & \ldots & y_N^o \end{bmatrix} K^{-1} K_x, \sigma_K(X) \Big) \end{bmatrix} \tag{27}$$

the above expression can be further elaborated, leading to the distribution of an isotropic Gaussian:

$$Y(X|X_{1,\ldots,N}) \sim \mathcal{N}\Big(\begin{bmatrix} y_1^1 & \ldots & y_N^1 \\ \vdots & \ddots & \vdots \\ y_1^o & \ldots & y_N^o \end{bmatrix} K^{-1} K_x, \sigma_K(X) I_{o \times o} \Big) \tag{28}$$

$$\sim \mathcal{N}\Big(\begin{bmatrix} Y_1| & \ldots & |Y_N \end{bmatrix} K^{-1} K_x, \sigma_K(X) I_{o \times o} \Big) \tag{29}$$

3.2 Training

The logarithmic likelihood is the summation of the logarithmic likelihood of each process that compose the vectorial **GP**, which leads, omitting constant

terms, to:

$$\mathcal{L} = \sum_{k=1}^{o} \Big(-\frac{1}{2} log\big(|K| \big) - \frac{1}{2} Tr\Big[K^{-1} \begin{bmatrix} y_1^i \\ \vdots \\ y_N^i \end{bmatrix} \begin{bmatrix} y_1^i & \dots & y_N^i \end{bmatrix} \Big] \Big) \tag{30}$$

$$= -\frac{o}{2} log\big(|K| \big) - \frac{1}{2} \sum_{k=1}^{o} \Big(Tr\Big[K^{-1} \begin{bmatrix} y_1^i \\ \vdots \\ y_N^i \end{bmatrix} \begin{bmatrix} y_1^i & \dots & y_N^i \end{bmatrix} \Big] \Big) \tag{31}$$

$$= -\frac{o}{2} log\big(|K| \big) - \frac{1}{2} Tr\Big[K^{-1} \sum_{k=1}^{o} \Big(\begin{bmatrix} y_1^i \\ \vdots \\ y_N^i \end{bmatrix} \begin{bmatrix} y_1^i & \dots & y_N^i \end{bmatrix} \Big) \Big] \tag{32}$$

$$= -\frac{o}{2} log\big(|K| \big) - \frac{1}{2} Tr\Big[K^{-1} M_{YY}^o \Big] \tag{33}$$

with:

$$M_{YY}^o = \sum_{k=1}^{o} \Big(\begin{bmatrix} y_1^i \\ \vdots \\ y_N^i \end{bmatrix} \begin{bmatrix} y_1^i & \dots & y_N^i \end{bmatrix} \Big) \tag{34}$$

$$= \sum_{k=1}^{o} \begin{bmatrix} y_1^i y_1^i & \dots & y_1^i y_N^i \\ \vdots & \ddots & \vdots \\ y_N^i y_1^i & \dots & y_N^i y_N^i \end{bmatrix} \tag{35}$$

$$= \begin{bmatrix} \sum_{k=1}^{o} y_1^i y_1^i & \dots & \sum_{k=1}^{o} y_1^i y_N^i \\ \vdots & \ddots & \vdots \\ \sum_{k=1}^{o} y_N^i y_1^i & \dots & \sum_{k=1}^{o} y_N^i y_N^i \end{bmatrix} \tag{36}$$

$$= \begin{bmatrix} < Y_1, Y_1 > & \dots & < Y_1, Y_N > \\ \vdots & \ddots & \vdots \\ < Y_N, Y_1 > & \dots & < Y_N, Y_N > \end{bmatrix} \tag{37}$$

keeping again in mind that $\mathcal{L}$ is a function of the hyperparameters Θ:

$$\mathcal{L}(\Theta) = -\frac{o}{2} log\big(\big|K\big(\Theta\big)\big| \big) - \frac{1}{2} Tr\Big[K^{-1}\big(\Theta\big) M_{YY}^o \Big] \tag{38}$$

The gradient can be computed with the same steps that led to eq. (22), leading to:

$$\frac{\partial \mathcal{L}}{\partial \theta_t} = \frac{1}{2} Tr\Big[K^{-1} \frac{\partial K}{\partial \theta_t} \Big(K^{-1} M_{YY}^o - o I_{N \times N} \Big) \Big] \tag{39}$$

3.2.1 Hyperparameters prior knowledge

It is possible to also take into account an a-priori knowledge of the process, in terms of a prior distribution describing the hyperparameters values. Such priors

modify the likelihood function to optimize in this way:

$$L^{'}(\Theta) = L(\Theta)L(\Theta)_{prior} \tag{40}$$

In principle, any distribution can be used as a prior for the hyperparameters. Without loss of generality, assume the prior is modelled with a multivariate gaussian distribution defined by a certain mean μ_{prior} and covariance Σ_{prior}. In such a case, equation (40) became:

$$L^{'}(\Theta) \propto L(\Theta)exp\bigg(-\frac{1}{2}\big(\Theta - \mu_{prior}\big)^{T}\Sigma_{prior}^{-1}\big(\Theta - \mu_{prior}\big)\bigg) \tag{41}$$

passing to the logarithm and neglecting constant values we get:

$$\mathcal{L}^{'}(\Theta) = \mathcal{L}(\Theta) - \frac{1}{2}\big(\Theta - \mu_{prior}\big)^{T}\Sigma_{prior}^{-1}\big(\Theta - \mu_{prior}\big) \tag{42}$$

Clearly, the gradient of the complete likelihood $\mathcal{L}^{'}(\Theta)$ can be computed as follows:

$$\frac{\partial \mathcal{L}^{'}}{\partial \Theta}^{T} = \frac{\partial \mathcal{L}}{\partial \Theta}^{T} - \Sigma_{prior}^{-1}\big(\Theta - \mu_{prior}\big) \tag{43}$$

4 Kernel functions

The kernel function describes the correlation between the inputs. Ideally, it should assume a low value for inputs that are "far", w.r.t. a certain metrics, from each other and high values for those inputs that are close.
The kernel function k should be designed in order to produce a symemtric positive definite matrix K, as this latter should be representative of a covariance matrix, see eq. (5).
Clearly, the gradient of K can be computed element by element:

$$\frac{\partial K}{\partial \theta_t} = \begin{bmatrix} \frac{\partial k(X_1,X_1)}{\partial \theta_t} & \cdots & \frac{\partial k(X_1,X_N)}{\partial \theta_t} \\ \vdots & \ddots & \vdots \\ \frac{\partial k(X_N,X_1)}{\partial \theta_t} & \cdots & \frac{\partial k(X_N,X_N)}{\partial \theta_t} \end{bmatrix} \tag{44}$$

In the following of this Section, some of the most popular kernel functions [1] will be discussed.

4.1 Linear function

Hyperparameters:

$$\Theta = \begin{bmatrix} \theta_0 & \theta_1 & \mu_1 & \dots & \mu_o \end{bmatrix} \tag{45}$$

[1] Which are also the ones default supported by this package.

Fuction evaluation:

$$\begin{aligned} k(a,b,\Theta) &= \theta_0^2 + \theta_1^2 \big(a-\mu\big)^T \big(b-\mu\big) && (46)\\ &= \theta_0^2 + \theta_1^2 \big(<\mu,\mu> + <a,b> - <\mu, a+b> \big) && (47) \end{aligned}$$

Function gradient:

$$\begin{aligned} \frac{\partial k(a,b)}{\partial \theta_0} &= 2\theta_0 && (48)\\ \frac{\partial k(a,b)}{\partial \theta_1} &= 2\theta_1 \big(a-\mu\big)\big(b-\mu\big) && (49)\\ \begin{bmatrix} \frac{\partial k(a,b)}{\partial \mu_1} \\ \vdots \\ \frac{\partial k(a,b)}{\partial \mu_o} \end{bmatrix} &= \theta_1^2 \big(2\mu - a - b\big) && (50) \end{aligned}$$

4.2 Squared exponential

Hyperparameters:

$$\Theta = \begin{bmatrix} \sigma & d \end{bmatrix} \tag{51}$$

Fuction evaluation:

$$k(a,b,\Theta) = \sigma^2 exp\bigg(- \frac{\|a-b\|_2^2}{d^2} \bigg) \tag{52}$$

Function gradient:

$$\begin{aligned} \frac{\partial k(a,b)}{\partial \sigma} &= 2\sigma exp\bigg(- \frac{\|a-b\|_2^2}{d^2} \bigg) && (53)\\ \frac{\partial k(a,b)}{\partial d} &= \sigma^2 exp\bigg(- \frac{\|a-b\|_2^2}{d^2} \bigg) 2\frac{\|a-b\|_2^2}{d^3} && (54) \end{aligned}$$

4.3 Periodic kernel

This function is able to well express the periodicity in the function to approximate. The yyperparameters are:

$$\Theta = \begin{bmatrix} \sigma & d & p \end{bmatrix} \tag{55}$$

p represents the period of the kernel function.

Fuction evaluation:

$$k(a,b,\Theta) = \sigma^2 exp\bigg(- \frac{sin^2\big(\frac{2\pi}{p}\,\|a-b\|_2\big)}{d^2} \bigg) = \sigma^2 exp\bigg(- \frac{sin^2\big(\alpha\big)}{d^2} \bigg) \tag{56}$$

Function gradient:

$$\frac{\partial k(a,b)}{\partial \sigma} = 2\sigma exp\Big(-\frac{sin^2(\alpha)}{d^2}\Big) \tag{57}$$

$$\frac{\partial k(a,b)}{\partial d} = \sigma^2 exp\Big(-\frac{sin^2(\alpha)}{d^2}\Big)2\frac{sin^2(\alpha)}{d^3} \tag{58}$$

$$\begin{aligned}\frac{\partial k(a,b)}{\partial p} &= \sigma^2 exp\Big(-\frac{sin^2(\alpha)}{d^2}\Big)\Big(-\frac{2sin(\alpha)cos(\alpha)}{d^2}\Big)\frac{\partial \alpha}{\partial p} \\ &= \sigma^2 exp\Big(-\frac{sin^2(\alpha)}{d^2}\Big)\Big(-\frac{2sin(\alpha)cos(\alpha)}{d^2}\Big)\Big(-\frac{2\pi \left\|a-b\right\|_2}{p^2}\Big) \\ &= \sigma^2 exp\Big(-\frac{sin^2(\alpha)}{d^2}\Big)\frac{2sin(\alpha)cos(\alpha)\alpha}{pd^2}\end{aligned} \tag{59}$$

4.4 Combining kernel functions

Clearly, you can also combine the kernel functions described in this Section to create more complex one.

4.4.1 Summation

You can sum more kernel functions $k_{1,\ldots,m}$ together, each having its own group of hyperparameters $\Theta_{1,\ldots,m}$:

$$\Theta = \begin{bmatrix}\Theta_1^T & \ldots & \Theta_m^T\end{bmatrix}^T \tag{60}$$

$$k(a,b,\Theta) = \sum_{i=1}^{m} k_i(a,b,\Theta_i) \tag{61}$$

The gradient is clearly computed as follows:

$$\frac{\partial k(a,b)}{\partial \Theta}^T = \begin{bmatrix}\frac{\partial k_1(a,b)}{\partial \Theta_1}^T \\ \vdots \\ \frac{\partial k_m(a,b)}{\partial \Theta_m}^T\end{bmatrix} \tag{62}$$

4.4.2 Product

You can multiply bunch of kernel functions $k_{1,\ldots,m}$ together, each having its own group of hyperparameters $\Theta_{1,\ldots,m}$:

$$\Theta = \begin{bmatrix}\Theta_1^T & \ldots & \Theta_m^T\end{bmatrix}^T \tag{63}$$

$$k(a,b,\Theta) = \prod_{i=1}^{m} k_i(a,b,\Theta_i) \tag{64}$$

The gradient is clearly computed as follows:

$$\frac{\partial k(a,b)}{\partial \Theta}^T = \left(\prod_{i=1}^{m} k_i(a,b) \right) \begin{bmatrix} \frac{\partial k_1(a,b)}{\partial \Theta_1}^T \frac{1}{k_1(a,b)} \\ \vdots \\ \frac{\partial k_m(a,b)}{\partial \Theta_m}^T \frac{1}{k_m(a,b)} \end{bmatrix} \tag{65}$$

A Trace property

Take an (n, n) matrix A and a vector X, the scalar quantity $x^T Ax$ is equal to:

$$x^T Ax = Tr\left[Axx^T \right] \tag{66}$$

$$= Tr\left[xx^T A^T \right] \tag{67}$$

Clearly, in case of symmetric matrix, the following holds:

$$x^T Ax = Tr\left[xx^T A \right] \tag{68}$$

We will now prove equation (66).
$x^T Ax$ can be also expressed as follows:

$$x^T Ax = x^T \begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix} x \tag{69}$$

$$= x^T \begin{bmatrix} a_1^T x \\ \vdots \\ a_n^T x \end{bmatrix} = x^T \begin{bmatrix} < a_1, x > \\ \vdots \\ < a_n, x > \end{bmatrix} \tag{70}$$

$$= \sum_{i=1}^{n} x_i < a_i, x > \tag{71}$$

where a_i is the i^{th} row of A. At the same time, the following fact is also true:

$$Tr\left[Axx^T \right] = Tr\left[\begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix} \begin{bmatrix} xx_1 & \dots & xx_n \end{bmatrix} \right] \tag{72}$$

$$= Tr\left[\begin{bmatrix} a_1^T x x_1 & & \\ & \ddots & \\ & & a_n^T x x_n \end{bmatrix} \right] \tag{73}$$

$$= \sum_{i=1}^{n} x_i < a_i, x > \tag{74}$$

where we recognize that eq. (71) and (74) are identical.

References

[1] Richard A. Davis. Gaussian process: Theory, 2014.

[2] K. B. Petersen and M. S. Pedersen. The matrix cookbook, 2008.

[3] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.