Gaussian Processes

Andrea Casalino

June 3, 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 2 formulations are not in contradiction and the decision to discuss them into 2 separate Sections is only 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):

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

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:

$$\left[\begin{array}{c} y = G(X) \\ \hline y_1 \\ \vdots \\ y_N \end{array}\right] \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \hline 0 \end{bmatrix}, \left[\begin{array}{c|c} k(X, X, \Theta) & K_x(X, X_{1,\dots,N}, \Theta)^T \\ \hline K_x(X, X_{1,\dots,N}, \Theta) & K \end{array}\right] \right) \tag{8}$$

where K_x is a vector assembled in this way:

$$K_x(X, X_{1,\dots,N}, \Theta) = \begin{bmatrix} k(X, X_1, \Theta) & \dots & k(X, X_N, \Theta) \end{bmatrix}^T \tag{9}$$

Since eq. 9 describes a Gaussian distibution, the conditioned distribution involving only X can be obtained as follows:

$$y(X|X_{1,\dots,N}) \sim \mathcal{N}\left(K_x K^{-1} \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \sigma_K(X) \right) \tag{10}$$

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

$$\sigma_K(X) = k(X, X) - K_x K^{-1} K_x^T \tag{11}$$

The distribution described by eq. 10, actually represents the prediction made by the **GP**. Eq. 10 can also be rewritten as follows:

$$y(X|X_{1,\dots,N}) \sim \mathcal{N}\left(\begin{bmatrix} y_1 & \dots & y_N \end{bmatrix} K^{-1} K_x^T, \sigma_K(X) \right) \tag{12}$$

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:

$$L = \frac{1}{\sqrt{(2\pi)^N \, |K|}} exp\Big(-\frac{1}{2} \begin{bmatrix} y_1 & \dots & y_N \end{bmatrix} K^{-1} \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \Big) \tag{13}$$

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

$$\begin{aligned} L &= \frac{1}{\sqrt{(2\pi)^N \, |K|}} exp\Big(-\frac{1}{2} Tr\Big[K^{-1} \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \begin{bmatrix} y_1 & \dots & y_N \end{bmatrix} \Big] \Big) & (14) \\ &= \frac{1}{\sqrt{(2\pi)^N \, |K|}} exp\Big(-\frac{1}{2} Tr\Big[K^{-1} M_{YY} \Big] \Big) & (15) \end{aligned}$$

Passing to the logarithm we obtain:

$$\mathcal{L} = log(L) = -\frac{N}{2}(2\pi) - \frac{1}{2} log\big(\, |K| \, \big) - \frac{1}{2} Tr\Big[K^{-1} M_{YY} \Big] \tag{16}$$

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

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

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

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta_t} &= -\frac{1}{2} Tr\Big[K^{-1} \frac{\partial K}{\partial \theta_t} \Big] - \frac{1}{2} Tr\Big[\frac{\partial}{\partial \theta_t} \big(K^{-1} M_{YY} \big) \Big] & (18) \\ &= -\frac{1}{2} Tr\Big[K^{-1} \frac{\partial K}{\partial \theta_t} \Big] - \frac{1}{2} Tr\Big[\frac{\partial (K^{-1})}{\partial \theta_t} M_{YY} \Big] & (19) \\ &= -\frac{1}{2} Tr\Big[K^{-1} \frac{\partial K}{\partial \theta_t} \Big] + \frac{1}{2} Tr\Big[K^{-1} \frac{\partial K}{\partial \theta_t} K^{-1} M_{YY} \Big] & (20) \\ &= \frac{1}{2} Tr\Big[K^{-1} \frac{\partial K}{\partial \theta_t} \Big(K^{-1} M_{YY} - I_{N \times N} \Big) \Big] & (21) \end{aligned}$$

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

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:

$$S = \left\{ \begin{bmatrix} X_1 \\ \downarrow \\ Y_1 \end{bmatrix}, \ldots, \begin{bmatrix} X_N \\ \downarrow \\ Y_N \end{bmatrix} \right\} \tag{22}$$

$$\left\{X_1, \ldots, X_N\right\} \subset \mathbb{R}^i \tag{23}$$

$$\left\{Y_1, \ldots, Y_N\right\} \subset \mathbb{R}^o \tag{24}$$

The generic Y_k is a vector made of o components:

$$Y_k = \begin{bmatrix} y_k^1 & \ldots & y_k^o \end{bmatrix}^T \tag{25}$$

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 10. Therefore, the complete prediction can be obtained in this way:

$$Y(X|X_{1,\ldots,N}) = \begin{bmatrix} \mathcal{N}\Big(\begin{bmatrix} y_1^1 & \ldots & y_N^1 \end{bmatrix} K^{-1} K_x^T, \sigma_K(X) \Big) \\ \vdots \\ \mathcal{N}\Big(\begin{bmatrix} y_1^o & \ldots & y_N^o \end{bmatrix} K^{-1} K_x^T, \sigma_K(X) \Big) \end{bmatrix} \tag{26}$$

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

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

$$\sim \mathcal{N}\Big(\begin{bmatrix} Y_1| & \ldots & |Y_N \end{bmatrix} K^{-1} K_x^T, \sigma_K(X) I_{o \times o} \Big) \tag{28}$$

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:

$$\mathcal{L} = \sum_{k=1}^{o}\Big(-\frac{1}{2}log\big(|K|\big)-\frac{1}{2}Tr\Big[K^{-1}\begin{bmatrix}y_1^i\\ \vdots\\ y_N^i\end{bmatrix}\begin{bmatrix}y_1^i & \dots & y_N^i\end{bmatrix}\Big]\Big) \quad (29)$$

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

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

$$= -\frac{o}{2}log\big(|K|\big)-\frac{1}{2}Tr\Big[K^{-1}M_{YY}^{o}\Big] \quad (32)$$

with:

$$M_{YY}^{o} = \sum_{k=1}^{o}\Big(\begin{bmatrix}y_1^i\\ \vdots\\ y_N^i\end{bmatrix}\begin{bmatrix}y_1^i & \dots & y_N^i\end{bmatrix}\Big) \quad (33)$$

$$= \sum_{k=1}^{o}\begin{bmatrix}y_1^iy_1^i & \dots & y_1^iy_N^i\\ \vdots & \ddots & \vdots\\ y_N^iy_1^i & \dots & y_N^iy_N^i\end{bmatrix} \quad (34)$$

$$= \begin{bmatrix}\sum_{k=1}^{o}y_1^iy_1^i & \dots & \sum_{k=1}^{o}y_1^iy_N^i\\ \vdots & \ddots & \vdots\\ \sum_{k=1}^{o}y_N^iy_1^i & \dots & \sum_{k=1}^{o}y_N^iy_N^i\end{bmatrix} \quad (35)$$

$$= \begin{bmatrix}< Y_1,Y_1 > & \dots & < Y_1,Y_N >\\ \vdots & \ddots & \vdots\\ < Y_N,Y_1 > & \dots & < Y_N,Y_N >\end{bmatrix} \quad (36)$$

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

$$\mathcal{L}(\Theta) = -\frac{o}{2}log\big(\,|K(\Theta)|\,\big)-\frac{1}{2}Tr\Big[K^{-1}(\Theta)M_{YY}^{o}\Big] \quad (37)$$

The gradient can be computed with the same steps that led to eq. (21), leading to:

$$\frac{1}{2}Tr\Big[K^{-1}\frac{\partial K}{\partial\theta_t}\Big(K^{-1}M_{YY}^{o}-oI_{N\times N}\Big)\Big] \quad (38)$$

4 Kernel functions

The kernel function describes correlation between 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:

$$\frac{\partial K}{\partial \theta_t} = \begin{bmatrix} \frac{\partial k(X_1,X_1)}{\partial \theta_t} & \cdots & \frac{\partial k(X_1,X_N)}{\partial \theta_t} \\ \vdots & \ddots & \vdots \\ \frac{\partial k(X_N,X_1)}{\partial \theta_t} & \cdots & \frac{\partial k(X_N,X_N)}{\partial \theta_t} \end{bmatrix} \tag{39}$$

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

4.1 Linear function

Hyperparameters:

$$\Theta = \begin{bmatrix} \theta_0 & \theta_1 & \mu_1 & \dots & \mu_o \end{bmatrix} \tag{40}$$

Fuction evaluation:

$$\begin{aligned} k(a,b,\Theta) &= \theta_0^2 + \theta_1^2 \big(a-\mu\big)^T \big(b-\mu\big) && (41)\\ &= \theta_0^2 + \theta_1^2 \big(< \mu,\mu > + < a,b > - < \mu, a+b > \big) && (42) \end{aligned}$$

Function gradient:

$$\begin{aligned} \frac{\partial k(a,b)}{\partial \theta_0} &= 2\theta_0 && (43)\\ \frac{\partial k(a,b)}{\partial \theta_1} &= 2\theta_1 \big(a-\mu\big)\big(b-\mu\big) && (44)\\ \begin{bmatrix} \frac{\partial k(a,b)}{\partial \mu^1} \\ \vdots \\ \frac{\partial k(a,b)}{\partial \mu^o} \end{bmatrix} &= 2\theta_1^2 \mu - \theta_1^2 \big(a+b\big) && (45) \end{aligned}$$

4.2 Squared exponential

Hyperparameters:

$$\Theta = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix} \tag{46}$$

Fuction evaluation:

$$\begin{aligned} k(a,b,\Theta) &= \theta_0^2 exp\big(-\theta_1^2 \left\| a-b \right\|_2^2 \big) && (47)\\ &= \theta_0^2 exp\big(-\theta_1^2 (a-b)^T (a-b) \big) && (48) \end{aligned}$$

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

Function gradient:

$$\frac{\partial k(a,b)}{\partial \theta_0} = 2\theta_0 exp\big(-\theta_1^2 \left\|a-b\right\|_2^2\big) \tag{49}$$

$$\frac{\partial k(a,b)}{\partial \theta_1} = \theta_0^2 exp\big(-\theta_1^2 \left\|a-b\right\|_2^2\big)\big(-2\theta_1(a-b)^T(a-b)\big) \tag{50}$$

4.3 What to expect from the predictions

TODO spiegare che incertezza aumenta tanto piu sonon lontano da punti in training set

A Trace property

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

$$x^TAx = Tr\Big[Axx^T\Big] \tag{51}$$

$$= Tr\Big[xx^TA^T\Big] \tag{52}$$

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

$$x^TAx = Tr\Big[xx^TA\Big] \tag{53}$$

We will now prove equation (51).
x^TAx can be also expressed as follows:

$$x^TAx = x^T\begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix} x \tag{54}$$

$$= x^T\begin{bmatrix} a_1^Tx \\ \vdots \\ a_n^Tx \end{bmatrix} = x^T\begin{bmatrix} < a_1, x > \\ \vdots \\ < a_n, x > \end{bmatrix} \tag{55}$$

$$= \sum_{i=1}^{n} x_i < a_i, x > \tag{56}$$

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

$$
\begin{aligned}
Tr\Big[Axx^T\Big] &= Tr\Bigg[\begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix}\begin{bmatrix} xx_1 & \dots & xx_n \end{bmatrix}\Bigg] && (57)\\
&= Tr\Bigg[\begin{bmatrix} a_1^T x x_1 & & \\ & \ddots & \\ & & a_n^T x x_n \end{bmatrix}\Bigg] && (58)\\
&= \sum_{i=1}^{n} x_i < a_i, x > && (59)
\end{aligned}
$$

where we recognize that eq. (56) and (59) 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.