Gaussian Processes

Andrea Casalino

February 1, 2022

1 What is a Gaussian Process?

Gaussian Processes [3], [1], a.k.a. **GP**s, are predictive models able to approximate a multivariate scalar function:

$$g : \mathcal{I} \rightarrow \mathcal{O} \tag{1}$$
$$\mathcal{I} \subseteq \mathbb{R}^i \quad \mathcal{O} \subseteq \mathbb{R} \tag{2}$$

or a multivariate vectorial one:

$$G : \mathcal{I} \rightarrow \mathcal{O} \tag{3}$$
$$\mathcal{I} \subseteq \mathbb{R}^i \quad \mathcal{O} \subseteq \mathbb{R}^o \tag{4}$$

GPs are essentially defined by a **Training Set** and a **Kernel Function**. The **Training Set** is a collection of points pertaining to $\mathcal{I}$, for which the corresponding output inside $\mathcal{O}$ is known [1].
The concept of **Kernel Function** is extensively detailed at Section 4.

2 Scalar case

The training set of a scalar **GP** is formally defined in this way:

$$\begin{aligned} S &= \left\langle \begin{bmatrix} X^1 \\ y^1 \end{bmatrix} \dots \begin{bmatrix} X^N \\ y^N \end{bmatrix} \right\rangle && (5) \\ \mathcal{X}^S &= \left\{ X^1 \dots X^N \right\} \subseteq \mathcal{I} && (6) \\ \mathcal{Y}^S &= \left\{ y^1 \dots y^N \right\} \subseteq \mathbb{R} && (7) \end{aligned}$$

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 a **Gaussian Distribution**:

$$\begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix} \sim \mathcal{N}\Big(0, K(\mathcal{X}^S)\Big) \tag{8}$$

K is the kernel covariance, whose values depend on the definition of a kernel function k (refer to Section 4):

$$K = K(\mathcal{X}^S, \Theta) = \begin{bmatrix} k(X^1, X^1, \Theta) & \dots & k(X^1, X^N, \Theta) \\ \vdots & \ddots & \vdots \\ k(X^N, X^1, \Theta) & \dots & k(X^N, X^N, \Theta) \end{bmatrix} \tag{9}$$

where Θ is a vector of hyperparamters that can be tuned by training (see Sections 2.2 and 3.2) the model over a specific training set:

$$\Theta = \begin{bmatrix} \theta_1 & \dots & \theta_m \end{bmatrix}^T \tag{10}$$

[1]To be precise, the exact value might be unknown due to noise, but a also a close one is fine.

2.1 Predictions

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

$$\left[\begin{array}{c} y(X) \\ \hline y^1 \\ \vdots \\ y^N \end{array}\right] \sim \mathcal{N}\left(\left[\begin{array}{c} 0 \\ \hline 0 \end{array}\right], \left[\begin{array}{c|c} k(X,X,\Theta) & K_x^T \\ \hline K_x(X,\mathcal{X}^S,\Theta) & K(\mathcal{X}^S,\Theta) \end{array}\right] \right) \tag{11}$$

where K_x is a vector obtained in the following way:

$$K_x(X,\mathcal{X}^S,\Theta) = \begin{bmatrix} k(X,X^1,\Theta) & \dots & k(X,X^N,\Theta) \end{bmatrix}^T \tag{12}$$

As the joint distribution is **Gaussian**, the conditioned distribution can be obtained as follows:

$$y(X|\mathcal{X}^S) \sim \mathcal{N}\left(K_x^T K^{-1} \begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix}, k(X,X) - K_x^T K^{-1} K_x \right) \tag{13}$$

2.2 Training

Training is done maximizing the likelihood L of the training set w.r.t. Θ. Since eq. (8) describes a **Gaussian** distribution, the likelihood can be computed as follows:

$$L(\mathcal{Y}^S) = \frac{1}{\sqrt{(2\pi)^N |K(\mathcal{X}^S)|}} exp\left(-\frac{1}{2} \begin{bmatrix} y^1 & \dots & y^N \end{bmatrix} K^{-1} \begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix} \right) \tag{14}$$

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

$$\begin{aligned} L(\mathcal{Y}^S) &= \frac{1}{\sqrt{(2\pi)^N |K|}} exp\left(-\frac{1}{2} Tr\left[K^{-1} \begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix} \begin{bmatrix} y^1 & \dots & y^N \end{bmatrix} \right] \right) &(15) \\ &= \frac{1}{\sqrt{(2\pi)^N |K|}} exp\left(-\frac{1}{2} Tr\left[K^{-1} M_{YY} \right] \right) &(16) \end{aligned}$$

Passing to the logarithm we obtain:

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

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]):

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

Choosing your favourite gradient-based approach you can then tune the model.

3 Vectorial case

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

$$S = \left\langle \begin{bmatrix} X^1 \\ \hline y_1^1 \\ \vdots \\ y_o^1 \end{bmatrix} \dots \begin{bmatrix} X^N \\ \hline y_1^N \\ \vdots \\ y_o^N \end{bmatrix} \right\rangle \tag{22}$$

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 $i \in \{0, \dots, o\}$ holds that:

$$\begin{bmatrix} y_i(X) \\ \hline y_i^1 \\ \vdots \\ y_i^N \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \hline 0 \end{bmatrix}, \left[\begin{array}{c|c} k(X, X, \Theta) & K_x^T \\ \hline K_x(X, \mathcal{X}^S, \Theta) & K(\mathcal{X}^S, \Theta) \end{array}\right] \right) \tag{23}$$

Then, the complete prediction is obtained in this way:

$$\begin{aligned}
Y(X|X^S,\Theta) &= \begin{bmatrix} y_1(X|X^S,\Theta) \\ \vdots \\ y_o(X|X^S,\Theta) \end{bmatrix} && (24) \\
&\sim \begin{bmatrix} \mathcal{N}\Big(K_x^T K^{-1} \begin{bmatrix} y_1^1 \\ \vdots \\ y_1^N \end{bmatrix}, k(X,X) - K_x^T K^{-1} K_x\Big) \\ \vdots \\ \mathcal{N}\Big(K_x^T K^{-1} \begin{bmatrix} y_o^1 \\ \vdots \\ y_o^N \end{bmatrix}, k(X,X) - K_x^T K^{-1} K_x\Big) \end{bmatrix} && (25) \\
&\sim \mathcal{N}\Bigg(\Bigg(K_x^T K^{-1} \begin{bmatrix} y_1^1 & \dots & y_o^1 \\ \vdots & \ddots & \vdots \\ y_1^N & \dots & y_o^N \end{bmatrix}\Bigg)^T, K_x^T K^{-1} K_x I_{o,o}\Bigg) && (26)
\end{aligned}$$

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:

$$\begin{aligned}
\mathcal{L} &= \sum_{i=0}^{o} \Bigg(-\frac{1}{2} log\big(|K|\big) - \frac{1}{2} Tr\Bigg[K^{-1} \begin{bmatrix} y_i^1 \\ \vdots \\ y_i^N \end{bmatrix} \begin{bmatrix} y_i^1 & \dots & y_i^N \end{bmatrix}\Bigg]\Bigg) && (27) \\
&= -\frac{o}{2} log\big(|K|\big) - \frac{1}{2} \sum_{i=0}^{o} \Bigg(Tr\Bigg[K^{-1} \begin{bmatrix} y_i^1 \\ \vdots \\ y_i^N \end{bmatrix} \begin{bmatrix} y_i^1 & \dots & y_i^N \end{bmatrix}\Bigg]\Bigg) && (28) \\
&= -\frac{o}{2} log\big(|K|\big) - \frac{1}{2} Tr\Bigg[K^{-1} \sum_{i=0}^{o} \Bigg(\begin{bmatrix} y_i^1 \\ \vdots \\ y_i^N \end{bmatrix} \begin{bmatrix} y_i^1 & \dots & y_i^N \end{bmatrix}\Bigg)\Bigg] && (29) \\
&= -\frac{o}{2} log\big(|K|\big) - \frac{1}{2} Tr\Big[K^{-1} M_{YY}^o\Big] && (30)
\end{aligned}$$

with:

$$M_{YY}^o = \sum_{i=0}^{o} \Bigg(\begin{bmatrix} y_i^1 \\ \vdots \\ y_i^N \end{bmatrix} \begin{bmatrix} y_i^1 & \dots & y_i^N \end{bmatrix}\Bigg) \qquad (31)$$

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

$$\mathcal{L}(\Theta) = -\frac{o}{2} log\Big(\big| K(\Theta) \big| \Big) - \frac{1}{2} Tr \Big[K^{-1}(\Theta) M^o_{YY} \Big] \tag{32}$$

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

$$\frac{\partial \mathcal{L}}{\partial \theta_t} = -\frac{o}{2} Tr \Big[K^{-1} \frac{\partial K}{\partial \theta_t} \Big] + \frac{1}{2} Tr \Big[K^{-1} \frac{\partial K}{\partial \theta_t} K^{-1} M^o_{YY} \Big] \tag{33}$$

4 The Kernel function

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. (8).
Clearly, the gradient of K can be computed element by element:

$$\frac{\partial K}{\partial \theta_t} = \begin{bmatrix} \frac{\partial k(X^1, X^1)}{\partial \theta_t} & \dots & \frac{\partial k(X^1, X^N)}{\partial \theta_t} \\ \vdots & \ddots & \vdots \\ \frac{\partial k(X^N, X^1)}{\partial \theta_t} & \dots & \frac{\partial k(X^N, X^N)}{\partial \theta_t} \end{bmatrix} \tag{34}$$

In the following of this Section, the most popular kernel functions [2] will be discussed.

4.1 Linear function

Hyperparameters:

$$\Theta = \begin{bmatrix} \theta_0 & \theta_1 & \mu_1 & \dots & \mu_o \end{bmatrix} \tag{35}$$

Fuction evaluation:

$$\begin{aligned} k(x, y, \Theta) &= \theta_0^2 + \theta_1^2 \big(x - \mu\big)^T \big(x - \mu\big) && (36) \\ &= \theta_0^2 + \theta_1^2 x^T y + \theta_1^2 \mu^T \mu - \mu^T \theta_1^2 \big(x + y\big) && (37) \end{aligned}$$

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

Function gradient:

$$\begin{aligned}
\frac{\partial k(x,y)}{\partial \theta_0} &= 2\theta_0 && (38)\\
\frac{\partial k(x,y)}{\partial \theta_1} &= 2\theta_1\big(x-\mu\big)\big(y-\mu\big) && (39)\\
\begin{bmatrix} \frac{\partial k(x,y)}{\partial \mu_1} \\ \vdots \\ \frac{\partial k(x,y)}{\partial \mu_o} \end{bmatrix} &= 2\theta_1^2\mu - \theta_1^2\big(x+y\big) && (40)
\end{aligned}$$

4.2 Squared exponential

Hyperparameters:

$$\Theta = \begin{bmatrix}\theta_0 & \theta_1\end{bmatrix} \qquad (41)$$

Fuction evaluation:

$$\begin{aligned}
k(x,y,\Theta) &= \theta_0^2 exp\big(-\theta_1^2 \left\|x-y\right\|_2^2\big) && (42)\\
&= \theta_0^2 exp\big(-\theta_1^2 (x-y)^T(x-y)\big) && (43)
\end{aligned}$$

Function gradient:

$$\begin{aligned}
\frac{\partial k(x,y)}{\partial \theta_0} &= 2\theta_0 exp\big(-\theta_1^2 \left\|x-y\right\|_2^2\big) && (44)\\
\frac{\partial k(x,y)}{\partial \theta_1} &= \theta_0^2 exp\big(-\theta_1^2 \left\|x-y\right\|_2^2\big)\big(-2\theta_1(x-y)^T(x-y)\big) && (45)
\end{aligned}$$

A Trace property

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

$$\begin{aligned}
x^TAx &= Tr\Big[Axx^T\Big] && (46)\\
&= Tr\Big[xx^TA^T\Big] && (47)
\end{aligned}$$

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

$$x^TAx = Tr\Big[xx^TA\Big] \qquad (48)$$

We will now prove equation (46).

$x^T Ax$ can be also expressed as follows:

$$x^T Ax = x^T \begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix} x \tag{49}$$

$$= x^T \begin{bmatrix} a_1^T x \\ \vdots \\ a_n^T x \end{bmatrix} = x^T \begin{bmatrix} < a_1, x > \\ \vdots \\ < a_n, x > \end{bmatrix} \tag{50}$$

$$= \sum_{i=0}^{n} x_i < a_i, x > \tag{51}$$

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

$$Tr\Big[Axx^T\Big] = Tr\Bigg[\begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix} \begin{bmatrix} xx_1 & \dots & xx_n \end{bmatrix}\Bigg] \tag{52}$$

$$= Tr\Bigg[\begin{bmatrix} a_1^T xx_1 & & \\ & \ddots & \\ & & a_n^T xx_n \end{bmatrix}\Bigg] \tag{53}$$

$$= \sum_{i=0}^{n} x_i < a_i, x > \tag{54}$$

where we recognize that eq. (51) and (54) 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.