Gaussian Processes

Bastien Chopard

May 2020

1 Introduction

Gaussian processes (GP) are a common tool in statistics and machine learning. They can be used for classification and regression. Regresssion, that is our main topic of interest here, means to *predict* the unknown values of a phenomena, or of a given quantity of interest, with a *quantification of the uncertainty* on these predicted values. More precisely, it gives a probability distribution from which a expectation value and standard deviation can be computed.

The prediction is based on some observations of the phenomena. Regression will interpolate these observations to points for which no observation is known. For instance one measures the temperature at several meteorological stations and one would like to have an estimate of the temperature everywhere. An example is given in Fig. 1 for a situation in 1D where a quantity y has been measured at some points x_1, x_2, x_3 and x_4. One has then four observed values y_1, y_2, y_3 and y_4, where

$$y_i = f(x_i)$$

for some unknown function f. Our goal is to give an estimate of y^*_j at location x^*_j, requiring that the function f is smooth. This means that $E(y^*)$ is a continuous, differentiable function of x^* for all x^* in the domain. It is also expected that for a x^* close to a point x at which y is known, one has $y^*(x^*) \approx y(x)$. GP offers such a smooth regression, compatible with observation points (often referred to as *training data*, in the language of machine learning). GP are also known as *Kriging* is geoscience.

GP do not provide a data interpolation as a mean-square fit does. In the latter case, one assumes an expression of $f(x) = y$ based on unknown *parameters*, that need to be computed to best fit the data. For instance one may assume that $f(x) = ax + b$. The parameters a and b are then optimally tuned to minimized the distance between the observations and the function. However, if the approximation is not good enough, another function should be considered, for instance $f(x) = ax^2 + bx + c$. Then, there are 3 parameters to determine.

GP are termed *non-parametric* regression as one does not impose an algebraic expression for f. As we shall see, there are always enough degrees of freedom in a GP to provide a smooth fit, compatible with the known data.

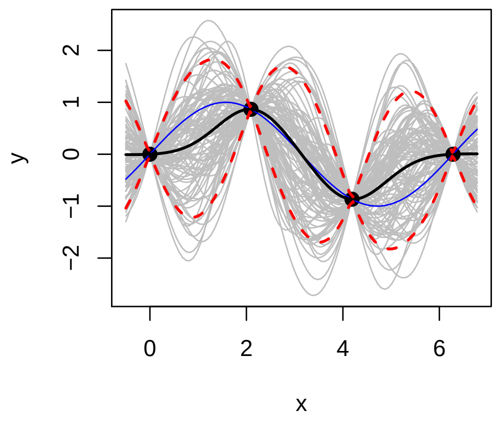

Figure 1: Example of a regression with predicted values specified by an expected value and a standard deviation. The prediction (or interpolation) y^*j at some location x^*j is based on known values (y_i, x_i).

The situation described in Fig. 1 can be easily extended in d-dimensions. The only change is that now x_i are vectors with d components. On the other hand, y_i is still a scalar, the value of some function $f(x_i)$. As an example, consider the situation illustrated in Fig. 2. We assume that we have a satellite image and that we have analyzed a few pixels only. Some are recognized as land, whereas the other are identified as representing water. The question is then to determine the shoreline based on these limited observations and the topography. The interpolation of the line separating the two regions is also a classification task as it predicts if a given pixel is part of the land or the water. Following the same idea, GP can also be used to smooth under-resolved imaged by adding additional pixels that interpolate smoothly the known ones.

There are plenty of websites explaining GP, with more or less details and background. Several codes in python, R, matlab, etc, are available to perform regression with GP. Several can be used as black-boxes and they provide predictions for unknown data, with uncertainty quantification. Here, we would like to explain some of the mathematics behind GP, which actually involve many steps that are often unexplained in many documents. In what follow, we will discuss and explain the following concepts: stochastic processes, multivariate normal distributions, sampling, kernels, as well as the prior and posterior distributions.

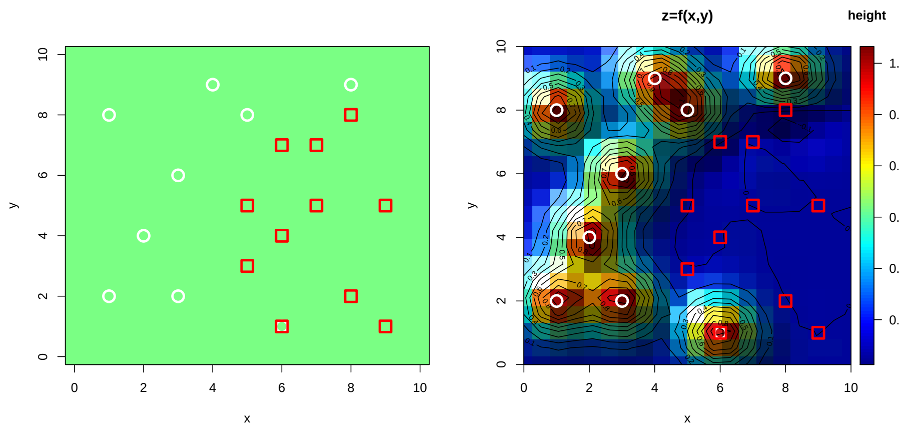

Figure 2: Situation of an image (for instance from a satellite) for which only a few pixels have been analyzed (left panel). Some pixels are identifies as corresponding to land regions (white dots, with elevation 1), while others are found to correspond to water (red square, with elevation 0). Using GP (right panel) one can estimate the corresponding topography and the shape of the lines separating the land from the sea.

2 Stochastic processes

In this section, we will give an intuitive description of stochastic processes. A stochastic process is a set of random variables identified by an index. For instance, a Markov process is a stochastic process in time, meaning that with each value of the time t, there is an associated random variable, denoted for instance as $y(t)$. Often one considers only discrete indices, t_0, t_1, t_2,.... The quantity $y_i = y(t_i)$ is a random variable and its value is taken from a given probability distribution. In general, this probability distribution can depend upon all the values of all $y(t_j)$, with $t_j < t_i$, as the whole history of the process may affect its current value. So, the process is specified by its joint probability distribution

$$\mathrm{Prob}(y(t_i), y(t_{i-1}), \ldots, y(t_0)) = \mathrm{Prob}(y(t_i)\mathrm{Prob}(y(t_i)|y(t_{i-1}), \ldots, y(t_0))$$

which can always be expressed with a conditional probability, as specified by the Bayes theorem. For a Markov process, we make the hypothesis that the present depends only on the immediate past, namely that

$$\mathrm{Prob}(y(t_i)|y(t_{i-1}) \ldots, y(t_0)) = \mathrm{Prob}(y(t_i)|y(t_{i-1}))$$

Within this approach, Gaussian Processes (GP) can be seen as a stochastic process in "space" rather than in time. The indices of the random variable y are space coordinate x. Although GP can be though of as a continuous process in x, it is enough to consider only discrete values of x, denoted as x_0, x_1,…, x_{n-1}. Then, we will consider n random variable y_i, corresponding to a process taking place at position x_i. Note that in general x_i can be d-dimensional, $x_i \in \mathbb{R}^d$, as the process can occur anywhere in space. In what follows, x will refer to all the positions at which the process is considered

$$x = (x_0, x_1, \ldots, x_{n-1})$$

It is a vector of n points, but possibly each point can have its own components. In this case, we will denote these components with a Greek symbol, as for instance

$$x_{i\alpha}, \qquad \alpha \in \{1, 2, \ldots, d\}$$

Likewise, the quantity y refers to all the y_i and it is a vector of size n.

$$y = (y_0, y_1, \ldots, y_{n-1})$$

but the values y_i are always scalars.

As suggested by their name, GP processes are characterized by a joint probability distribution

$$\text{Prob}(y) = \text{Prob}(y_0, y_1, \ldots, y_{n.1})$$

which is a Gaussian. We will discuss this aspect in more detail in the next section. But for now, let us just indicate that this Gaussian distribution will be tailored so that two random variables y_i and y_j will be strongly correlated if x_i and x_j are close to each other. In other words, the covariance $\text{Cov}(y_i, y_j)$ will be a function of x_i and x_j.

Considering that y is a function of $f(x)$, drawing y from the joint probability distribution $\text{Prob}(y)$ produces a random candidate for f. In this context, $\text{Prob}(y)$ is called the *prior* distribution, as no further knowledge on the values of y are known. Generating a random y from $\text{Prob}(y)$ is said to sample the prior distribution. It is not very interesting but can be done once we understand how to sample a multivariate distribution, see section 7. The interesting aspect when sampling the prior is to observe that the relation $y = f(x)$ is indeed smooth if the covariance is properly defined.

What makes GP interesting is when some values of y_i are known for some given x_i and one would like to predict the value y^* of the function at some other locations x^*, for which no observation is available. This amounts to sampling the conditional probability

$$\text{Prob}(y^*|y) = \frac{\text{Prob}(y, y^*)}{\text{Prob}(y)}$$

This conditional probability is also known as the *posterior* distribution. It can then be sampled to have candidates values for y^*, from which an expected value and a standard deviation can be computed.

Computing the posterior distribution and sampling it is not an easy task in general. But if the process is described by a Gaussian probability distribution, a lot of results can be computed analytically, as we can see in the next sections.

3 Multivariate Normal distributions

A multivariate normal (MVN) distribution is a Gaussian distribution for several variables y_0, y_1,…, y_{n-1}. It is denoted as

$$\mathcal{N}(y, \mu, \Sigma)$$

where

$$y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} \qquad \text{and} \qquad \mu = \begin{pmatrix} \mu_0 \\ \mu_1 \\ \vdots \\ \mu_{n-1} \end{pmatrix}$$

are $n \times 1$ matrices. The values μ_i are given constants which actually correspond to desired the expected value

$$\mu_i = E(y_i) = \int_{y \in \mathbb{R}^n} dy_i y_i \mathcal{N}(y, \mu, \Sigma)$$

. The quantity Σ is a $n \times n$ matrix which actually contains the desired covariance value between y_i and y_j

$$\Sigma_{ij} = \mathrm{Cov}(y_i, y_j) = E[(y_i - \mu_i)(y_j - \mu_j)]$$

The mathematical expression of a MVN is

$$\mathcal{N}(y, \mu, \Sigma) = (2\pi)^{-n/2} |\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(y - \mu)^T \Sigma^{-1} (y - mu) \right) \tag{1}$$

where the upper-script T denotes the transpose of the matrix, and Σ^{-1} is the inverse of Σ. Note that Σ is a symmetric, positive-definite matrix.

One usually write

$$y \sim \mathcal{N}(\mu, \Sigma)$$

to indicate that y is randomly generated according to a MVN of mean μ and covariance Σ. In that case the variable y is omitted from $\mathcal{N}(y, \mu, \Sigma)$.

4 Bloc partitioned matrices

Any matrix can be partitioned into blocs of submatrices. For instance a 4×4 matrix A can be partitioned in four submatrices A_{11}, A_{12}, A_{21} and A_{22} as

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \tag{2}$$

where

$$A_{11} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad A_{12} = \begin{pmatrix} a_{13} & a_{14} \\ a_{23} & a_{24} \end{pmatrix} \quad A_{21} = \begin{pmatrix} a_{31} & a_{32} \\ a_{41} & a_{42} \end{pmatrix} \quad A_{22} = \begin{pmatrix} a_{33} & a_{34} \\ a_{43} & a_{44} \end{pmatrix}$$

Let us now consider a matrix vector multiplication of partitioned matrices. It is easy to show that

$$A \begin{pmatrix} y_0 \\ y_1 \\ y_0^* \\ y_1^* \end{pmatrix} = \begin{pmatrix} A_{11} \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} + A_{12} \begin{pmatrix} y_0^* \\ y_1^* \end{pmatrix} \\ \\ A_{21} \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} + A_{22} \begin{pmatrix} y_0^* \\ y_1^* \end{pmatrix} \end{pmatrix}$$

By denoting

$$y = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \qquad y^* = \begin{pmatrix} y_0^* \\ y_1^* \end{pmatrix}$$

one further gets that

$$(y^T, (y^*)^T) A \begin{pmatrix} y \\ y^* \end{pmatrix} = y^T A_{11}\, y + y^T A_{12}\, y^* + (y^*)^T A_{21}\, y + (y^*)^T A_{22}\, y^* \quad (3)$$

The properties we have illustrated here with a 4×4 matrix are also valid for any size $(n+m) \times (n+m)$ matrix. The blocs A_{11}, A_{12}, A_{21}, A_{22}, are then $n \times n$, $n \times m$, $m \times n$, and $m \times m$, respectively. In that case, y would be n and y^* would be $m \times 1$.

5 Inverse and determinant of partitioned matrices

The inverse and determinant of bloc partitioned matrix can be expressed with algebraic expression. Before enunciating them it is necessary to derive the following property

Theorem 1 *Let A,B, C and D be square matrices, invertible when needed. Then*

$$(A + CBD)^{-1} = A^{-1} - A^{-1}C\left(B^{-1} + DA^{-1}C\right)^{-1} DA^{-1}$$

This relation can be proven by direct calculation

$$\begin{aligned} (A + CBD)[A^{-1} - A^{-1}C\left(B^{-1} + DA^{-1}C\right)^{-1} DA^{-1}] &= \ldots \\ &= I + CBDA^{-1} - CBDA^{-1} = I \end{aligned}$$

Let us now consider the inverse of a partitioned matrix.

Theorem 2 *Let A be a $(n+m) \times (n+m)$ symmetric matrix ($A = A^T$), partitioned in blocs A_{11}, A_{12}, $A_{21} = A_{12}^T$,A_{22} of sizes $n \times n$,$n \times m$, $m \times n$ and $m \times m$*

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix}$$

Then, the inverse of A is a matrix B with the same partitioning

$$B = \begin{pmatrix} B_{11} & B_{12} \\ B_{12}^T & B_{22} \end{pmatrix}$$

such that

$$\begin{aligned} B_{11} &= \left(A_{11} - A_{12}A_{22}^{-1}A_{12}^T\right)^{-1} && (4) \\ B_{22} &= \left(A_{22} - A_{12}^T A_{11}^{-1}A_{12}\right)^{-1} && (5) \\ B_{12} &= -A_{11}^{-1}A_{12}\left(A_{22} - A_{12}^T A_{11}^{-1}A_{12}\right)^{-1} && (6) \\ B_{12}^T &= -A_{22}^{-1}A_{12}^T\left(A_{11} - A_{12}A_{22}^{-1}A_{12}^T\right)^{-1} && (7) \end{aligned}$$

To prove this theorem, one imposes that

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix}\begin{pmatrix} B_{11} & B_{12} \\ B_{12}^T & B_{22} \end{pmatrix} = \begin{pmatrix} I_n & 0 \\ 0 & I_m \end{pmatrix}$$

where I_n and I_m are the identity matrices of sizes $n \times n$ and $m \times m$. By using the fact that the multiplication of partitioned matrix obey the same rule as usual matrices, we have the following conditions

$$\begin{aligned} A_{11}B_{11} + A_{12}B_{12}^T = I_n && (8) \\ A_{11}B_{12} + A_{12}B_{22} = 0 && (9) \\ A_{12}^T B_{11} + A_{22}B_{12}^T = 0 && (10) \\ A_{12}^T B_{12} + A_{22}B_{22} = I_m && (11) \end{aligned}$$

Equation (8) can be written as

$$B_{11} = A_{11}^{-1} - A_{11}^{-1}A_{12}B_{12}^T \tag{12}$$

In turns, eq. (10) reads

$$B_{12}^T = -A_{22}^{-1}A_{12}^T B_{11} \tag{13}$$

Now we can replace expression (13) into (12) and we get

$$B_{11} = A_{11}^{-1} + A_{11}^{-1}A_{12}A_{22}^{-1}A_{12}^T B_{11}$$

this is equivalent to

$$\left(I_m - A_{11}^{-1}A_{12}A_{22}^{-1}A_{12}^T\right)B_{11} = A_{11}^{-1}$$

and to

$$\left(A_{11} - A_{12}A_{22}^{-1}A_{12}^T\right)B_{11} = I$$

and, thus,

$$B_{11} = \left(A_{11} - A_{12}A_{22}^{-1}A_{12}^T\right)^{-1}$$

as announced by theorem 2. And we immediately obtained from (13 that

$$B_{12}^T = -A_{22}^{-1} A_{12}^T \left(A_{11} - A_{12} A_{22}^{-1} A_{12}^T\right)^{-1}$$

as also proposed in theorem 2. The expression for B_{22} and B_{12} can be obtained in a similar way.

Our last step is to compute the determinant of a partitioned matrix. We have the following theorem

Theorem 3 *Let us consider the a partitioned matrix A*

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix}$$

Then, the determinant of A is

$$\det \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix} = \begin{cases} \det(A_{11})\det(A_{22} - A_{12}^T A_{11}^{-1} A_{12}) \\ \\ \det(A_{22})\det(A_{11} - A_{12} A_{22}^{-1} A_{12}^T) \end{cases}$$

The curly bracket indicates that there are two equivalent expression of the determinant, depending if we want A_{11} or A_{22} to play the main role.

The proof of theorem 3 is first based on the well know results

$$\det(AB) = \det(A)\det(B)$$

and

$$\det \begin{pmatrix} B & 0 \\ C & D \end{pmatrix} = \det \begin{pmatrix} B & C \\ 0 & D \end{pmatrix} = \det(B)\det(A)$$

Then the proof of the theorem follows from the observation

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix} = \underbrace{\begin{pmatrix} A_{11} & 0 \\ A_{12}^T & I \end{pmatrix}}_{\det=\det(A_{11})} \underbrace{\begin{pmatrix} I & A_{11}^{-1} A_{12} \\ 0 & A_{22} - A_{12}^T A_{11}^{-1} A_{12} \end{pmatrix}}_{\det=\det(A_{22} - A_{12}^T A_{11}^{-1} A_{12})}$$

The second expression for the determinant that is proposed in theorem 3 is obtained by considering a different matrix product to decompose A, where A_{22} plays the symmetric role as that of A_{11} in the present proof.

6 Marginal and conditional probabilities of a multivariate Gaussian

Our goal is now to write the MVN $\mathcal{N}(\mu, \Sigma)$ according to a bloc partitioning defined by splitting the variables into a set $y = (y_0, y_1, \ldots, y_{n-1})^T$ and a set

$y^* = (y_0^*, y_1^*, \ldots, y_{m-1}^*)^T$.

$$\begin{aligned}\mathrm{Prob}(y, y^*) &= (2\pi)^{-(n+m)/2}|\Sigma|^{-1/2} \times \\ &\quad \times \exp\left(-\frac{1}{2}[(y-\mu_y)^T, (y^*-\mu_{y^*})^T]\Sigma^{-1}[(y-\mu_y)^T, (y^*-\mu_{y^*})^T\right) \\ &\equiv (2\pi)^{-(n+m)/2}|\Sigma|^{-1/2}\exp\left(-\frac{1}{2}Q(y,y^*)\right) \qquad (14)\end{aligned}$$

with

$$\Sigma = \begin{pmatrix} \Sigma_{yy} & \Sigma_{yy^*} \\ \Sigma_{yy^*}^T & \Sigma_{y^*y^*} \end{pmatrix}$$

and $Q(y, y^*)$ is the argument of the exponential. From theorem 2 we obtain Σ^{-1} as

$$\Sigma^{-1} = \begin{pmatrix} (\Sigma_{yy} - \Sigma_{yy^*}\Sigma_{y^*y^*}^{-1}\Sigma_{yy^*}^T)^{-1} & -\Sigma_{yy}^{-1}\Sigma_{yy^*}(\Sigma_{y^*y^*} - \Sigma_{yy^*}^T\Sigma_{yy}^{-1}\Sigma_{yy^*})^{-1} \\ -\Sigma_{y^*y^*}^{-1}\Sigma_{yy^*}^T(\Sigma_{yy} - \Sigma_{yy^*}\Sigma_{y^*y^*}^{-1}\Sigma_{yy^*}^T)^{-1} & (\Sigma_{y^*y^*} - \Sigma_{yy^*}^T\Sigma_{yy}^{-1}\Sigma_{yy^*})^{-1} \end{pmatrix}$$

The matrices that need to be inverted in the blocs of Σ^{-1} are of the form $(A - BCD)^{-1}$, as given in theorem 1. Thus

$$(\Sigma_{yy} - \Sigma_{yy^*}\Sigma_{y^*y^*}^{-1}\Sigma_{yy^*}^T)^{-1} = \Sigma_{yy}^{-1} + \Sigma_{yy}^{-1}\Sigma_{yy^*}(\Sigma_{y^*y^*} - \Sigma_{yy^*}^T\Sigma_{yy}^{-1}\Sigma_{yy^*})^{-1}\Sigma_{yy^*}\Sigma_{y^*y^*}^{-1}$$

Using this expression in 14 together with the form of Σ^{-1} for the other blocs, gives, after many steps of algebra[1]

$$Q(y, y^*) = (y - \mu_y)^T\Sigma_{yy}^{-1}(y - \mu_y) + (y^* - b)^T A^{-1}(y^* - b) \qquad (15)$$

with

$$\boxed{A = \Sigma_{y^*y^*} - \Sigma_{yy^*}^T\Sigma_{yy}^{-1}\Sigma_{y^*y} \qquad \text{and} \qquad b = \mu_{y^*} + \Sigma_{yy^*}^T\Sigma_{yy}^{-1}(y - \mu_y) \quad (16)}$$

Note that b, which is the expectation of y^* depends on y, but A does not.

Before we can compute the marginal and conditional probabilities associated with the MVN given in eq. (14), we need to express the determinant of Σ as a function of our bloc partitioning. Here we introduce the notation

$$\det(\Sigma) = |\Sigma|$$

Using theorem 3 eq. (14) reads

$$\begin{aligned}\mathrm{Prob}(y, y^*) &= (2\pi)^{-(n+m)/2}|\Sigma|^{-1/2}\exp\left(-\frac{1}{2}Q(y,y^*)\right) \\ &= \frac{(2\pi)^{-n/2}(2\pi)^{-m/2}}{|\Sigma_{yy}|^{1/2}|\Sigma_{y^*y^*} - \Sigma_{yy^*}^T\Sigma_{yy}^{-1}\Sigma_{yy^*}|^{1/2}}\exp\left(-\frac{1}{2}Q(y,y^*)\right)\end{aligned}$$

[1]See `http://fourier.eng.hmc.edu/e161/lectures/gaussianprocess/node7.html` for details

$$= \frac{(2\pi)^{-n/2}(2\pi)^{-m/2}}{|\Sigma_{yy}|^{1/2}|A|^{1/2}} \exp\left(-\frac{1}{2}[(y-\mu_y)^T\Sigma_{yy}^{-1}(y-\mu_y) + (y^*-b)^T A^{-1}(y^*-b)]\right)$$

$$= \frac{(2\pi)^{-n/2}}{|\Sigma_{yy}|^{1/2}} \exp\left(-\frac{1}{2}(y-\mu_y)^T\Sigma_{yy}^{-1}(y-\mu)\right) \frac{(2\pi)^{-m/2}}{|A|^{1/2}} \exp\left(-\frac{1}{2}(y^*-b)^T A^{-1}(y^*-b)\right)$$

Therefore we obtain the key result that the joint probability distribution for y and y^* can be written as as the product of two Gaussian distributions

$$\boxed{\text{Prob}(y, y^*) = \mathcal{N}(y, \mu_y, \Sigma_{yy})\mathcal{N}(y^*, b(y), A) \qquad (17)}$$

where the $m \times m$ matrix A and the m-element column vector $b(y)$ are defined in eq. (16).

From eq. (17), one immediately obtain the marginal distribution Prob(y) and the conditional probability Prob$(y^*|y)$

$$\text{Prob}(y) = \int_{y^* \in \mathbb{R}^m} dy^* \,\text{Prob}(y, y^*) = \mathcal{N}(y, \mu_y, \Sigma_{yy}) \qquad (18)$$

because, from its expression, $\mathcal{N}(y^*, b(y), A)$ is normalized to 1, for any value of y. The conditional probability is then obtained from Bayes formula

$$\boxed{\text{Prob}(y^*|y) = \frac{\text{Prob}(y, y^*)}{\text{Prob}(y)} = \mathcal{N}(y^*, b(y), A) \qquad (19)}$$

Note that the above formula introduce an asymmetry between y and y^*, which may look surprising as, at this point that are on equal footing. Actually other expression could have been derived where y^* would have played the role of y. For instance there are two possibility for the expression of $\det(\Sigma)$, as seen in theorem 3. Also, in the calculation of the inverse of the blocs forming Σ, we decided to use theorem **??** to express $(\Sigma)^{-1}_{yy}$. We could have used it for $(\Sigma)^{-1}_{y^*y^*}$, too.

7 Sampling

As mentioned several time, Prob$(y^*|y)$ given in eq. (19) is called the posterior, and this is the probability distribution that we need to sample to obtain predictions for the non-observed values y^* that are compatible with the known values y^*.

This raises the question of how to sample a probability distribution. With only one variable –let's us call it s– and an uniform random generator `uniform(0,1)`, procedure is the following. To generate a value of $s \in [s_{min}, s_{max}]$, distributed as

$p(s)$, one draws a random number `r=uniform(0,1)` and solves for s the equation

$$r = \int_{s_{min}}^{s} ds' \, p(s)$$

In case of a multivariate distribution, with several variables s_0, s_1,..., s_{m-1}, the situation may be very different as there usually exits correlations between these variables. However, if the variables are independent,

$$\text{Prob}(s_0, s_1, \ldots, s_{m-1}) = P_0(s_0) P_1(s_1) \ldots P_{m-1}(s_{m-1})$$

one just has to generate each component s_i of independently, according to their own probability distribution P_i, as explained for the case of a single variable.

For the general case, when correlations exit between the variable, the techniques called *Monte-Carlo-Markov-Chain* (MCMC) provides a way to sample a multivariate distribution whose expression $\text{Prob}(s_0, s_1, \ldots, s_{m-1})$ is known.

Fortunately, for a Gaussian distribution, there is a way to consider a change of variables that makes the new variables u independent. These new variables can be easily sampled. The standard approach is to use the so-called Cholesky decomposition of the covarance matrix Σ

$$\Sigma = LL^T \qquad \Sigma^{-1} = (L^T)^{-1} L^{-1}$$

where L is a lower triangular matrix. Algorithms exists to compute efficiently this decomposition. The new variables u are defined as

$$y = Lu + \mu$$

and $\mathcal{N}(y, \mu, \Sigma)$ becomes $\mathcal{N}(u, 0, I)$ because

$$\begin{aligned}
\exp\left(-\frac{1}{2}(y-\mu)^T \Sigma^{-1} (y-\mu)\right) &= \exp\left(-\frac{1}{2}(Lu)^T \Sigma Lu)\right) \\
&= \exp\left(-\frac{1}{2}(Lu)^T (L^T)^{-1} L^{-1} Lu)\right) \\
&= \exp\left(-\frac{1}{2}(\sum_{i=0}^{m-1} u_i^2)\right) \\
&= \Pi_{i=0}^{m-1} \exp\left(-\frac{1}{2}u_i^2\right)
\end{aligned}$$

Therefore, each component u_i can be separately sampled from a 1d Gaussian distribution, and y is computed form these values as $y = \mu + Ly$.

Note that, fortunately, the sampling of a MVN distribution is done by predefined function in several programming languages. For instance,

- In R, the library `plgp` provided the following solution:

$$y \leftarrow \text{rmvnorm}(\text{m}, \mu, \Sigma)$$

- In python numpy, there is the following function

$$y = \text{np.random.multivariate\_normal}(\mu, \Sigma, \text{m})$$

8 Kernels and Gaussian processes

The last step to define a Gaussian Process is to specified the value of the covariance matrix Σ. The goal is to enforce that the values of two random variables that represent the same quantity of interest at nearby locations are strongly correlated. The location, in space, of the random variables is specified by its index x. Therefore, we shall impose that

$$\Sigma(i,j) \equiv \mathrm{Cov}(y_i, y_j) = K(x_i, x_j) \tag{20}$$

where $K(x_i, x_j)$ is termed the *kernel* of the GP. The choice of K is arbitrary. The most common expression is the so-called "square exponential" (SE), also known as RBF kernel (Radial Basis Function) in some communities. It is defined as

$$\boxed{K(x_i, x_j) = \sigma^2 \exp\left(-\frac{1}{2}\frac{||x_i - x_j||^2}{\ell^2}\right)} \tag{21}$$

where σ and ℓ are called hyper-parameters, affecting the scale of the correlation between neighboring points x_i and x_j. Often, $\sigma = 1$, $\ell^2 = 2$ are chosen. The argument of the exponential is essentially the square of the Euclidean distance between x_i and x_j. Note that x_i can belong to $\mathbb{R}^d$, and that expression (21) is valid in any dimension. Therefore the generalization of GP to more spatial dimension is straightforward.

One observes that the above SE kernel has the properties to decrease as the distance between x_i and x_j increases. The correlation is strong for point close to each other, which will ensure that a sample $y = f(x)$ is a smooth and continuous function.

Other kernels have been proposed in the literature[2]. For instance, a linear kernel has been proposed as

$$K(x_i, x_j) = \sigma_p^2 + \sigma^2(x_i - c)(x_j - c) \tag{22}$$

for $x_i, x_j \in \mathbb{R}$, σ_p, σ^2 and c some hyper-parameters. A periodic kernel can be defined as

$$K(x_i, x_j) = \sigma^2 \exp\left(-\frac{2\sin^2\left(\frac{\pi}{p}||x_i - x_j||\right)}{\ell^2}\right) \tag{23}$$

with again σ, ℓ and p being hyper-parameters.

9 Algorithm

In this section we illustrate, in the language R, the algorithm that will allow us to sample the posterior distribution,

$$P(y^*|y) = \mathcal{N}(y^*, b(y), A)$$

[2] `https://distill.pub/2019/visual-exploration-gaussian-processes/`

. We will consider a simple example that can be easily generalized.

The first step is to load the proper R library

```
library(plgp)
```

Then we define the values of n space locations x. Note that in R, the assignment is indicated as `<-` which, for readability reasons, we shall indicate as $\leftarrow$.

```
n ← 8
x ← matrix(seq(0, 2*pi, length=n), ncol=1)
```

We follow this instruction by the values we choose for y. Here y is also a vector of size n

```
y ← sin(x)
```

Note that we could have input n values manually using the appropriate R construct, for instance as $y = (1, 3, 2, 5, 5, 7, 3, 4)$

```
y ← c(1,3,2,5,5,7,3,4)            # another possibility
```

Now we need to compute the covariance matrix Σ_{yy}, based on the kernel $K(x, x)$. In the library `plgp` we can use the function `distance()` that compute the Euclidean distance between all pairs of x

```
D ← distance(x)
```

D is a $n \times n$ matrix whose element ij is

$$D[i, j] = ||x_i - x_j||^2$$

With this, Σ_{yy} is obtained as

```
Σ_yy ← exp(-D)
```

We can now defined the m points x^*, for which we want to estimate the value $y^* = f(x^*)$, compatible the the observations $y = f(x)$. For instance we defined 100 of them, equally spaced on the interval $[-1/2, 2\pi + 1/2]$

```
m ← 100
x* ← matrix(seq(-0.5, 2*pi + 0.5, length=m), ncol=1)
```

As before, we compute the covariance matrix $\Sigma_{y^*y^*}$

```
$D_{**} \leftarrow \text{distance}(\text{x}^*)$
$\Sigma_{y^*y^*} \leftarrow \exp(-D_{**})$
```

Now we need to build the covariance matrix Σ_{y^*y}. We can use the fact the the function `distance()` can take two arguments

```
$D_* \leftarrow \text{distance}(\text{x}^*, \text{x})$
$\Sigma_{y^*y} \leftarrow \exp(-D_*)$
```

Here D_* is a $m \times n$ matrix and its elements are $D_*[i,j] = ||x_i^* - x_j||$. We now have to compute the matrices $A =$ and $b =$

$$A = \Sigma_{y^*y^*} - \Sigma_{y^*y}(\Sigma_{yy})^{-1}(\Sigma_{y^*y})^T \qquad b = \Sigma_{y^*y}(\Sigma_{yy})^{-1}y$$

where $\Sigma_{y^*y} = (\Sigma_{yy^*})^T$ and we chose to take $\mu = 0$, which is a common choice for GP.

```
$(\Sigma_{yy})^-1 \leftarrow \text{solve}(\Sigma_{\text{yy}})$
$A \leftarrow \Sigma_{y^*y^*} - \Sigma_{y^*y}\% * \%(\Sigma_{yy})^{-1}\% * \%t(\Sigma_{y^*y})$
$b \leftarrow \Sigma_{y^*y}\% * \%(\Sigma_{yy})^{-1}\% * \%y$
```

where $\% * \%$ is the symbol for matrix multiplication in R, and $t()$ is the transpose operation. The final step is to sample the multivariate normal distribution $\mathcal{N}(y^*, b, A)$, using the function `rmvnorm()` from the `plgp` library of R.

```
sample ← 100
$y^*$ ← rmvnorm(sample, b, A) # sampling of 100 functions
```

Here y^* is a matrix of size sample×m. Each of its line is a possible function $y^* = f(x^*)$. These functions can be plotted using R graphical functions.

```
matplot(XX, t(YY), type=”l”, col=”gray”, lty=1, xlab=”x”, ylab=”y”)
```

Here `type=’’l’’` means “line”, `lty=1` corresponds to the type of the line. The value 1 is a solid line, the value 2 would be a dashed line. The other parameters are self-explanatory. It is also interesting to see the training, or observation, points $x = f(x)$. This is obtained with

```
points(x,y, pch=20,cex=2)
```

where `pch=20` specifies the type of point. It corresponds to black disks (15 would be squares and 20 empty circles); `cex=2` determines the diameter. In order to see the expectation of the y^* values, one has to plot b as a function of x^*. This is obtained as

```
lines($x^*$, $b$,lwd=2)
```

producing a black (default color) of width `lwd=2`. As the observation y where obtained as $y = \sin(x)$ it is also interesting to plot $\sin(x^*)$ to see how well the interpolation corresponds to the actual function. This is done as a blue line with

```
lines($x^*$, sin($x^*$),col=”blue”)
```

Finally, for each x^*, one can view, as red dashed lines, the values of y^* which corresponds to 5% and 95% of the posterior distribution

```
$q1 \leftarrow b + \text{qnorm}(0.05, 0, \text{sqrt}(\text{diag}(\Sigma_{\text{y}^*\text{y}^*})))$
$q2 \leftarrow b + \text{qnorm}(0.95, 0, \text{sqrt}(\text{diag}(\Sigma_{\text{y}^*\text{y}^*})))$
$\text{lines}(x^*, q1, \text{lwd} = 2, \text{lty} = 2, \text{col} = \text{”red”})$
$\text{lines}(x^*, q2, \text{lwd} = 2, \text{lty} = 2, \text{col} = \text{”red”})$
```

where `qnorm()` computes the quantile of the corresponding normal distribution. The above code produced the image shown in Fig. 3.

10 Exercises

1. You need to install R on your computer. Then program in R the above GP algorithm in a file `gaussianProcess.r` (you just need to choose appropriate variable names and type...). Then, from the command line you start R by typing R. From the R interpreter, you can install the `plpg` library by issuing the instruction `install.packages(c("plgp")`. Then you can run your code by typing `source(‘‘gaussianProcess.r’’`.

2. See what happens when you change n, the number of training point y, or m, the number of testing points.

3. Take $n = 8$ and $m = 15$ and define x^* in the same interval as x. With this choice $x^*[3] = x[2]$. Print in the interpreter the values of $y[2]$ and $b[3]$. Are they identical? Print the variance of $y^*[2]$ and that of $y^*[3]$.

4. $b[i]$ is the average value of $y^*[i]$ at location $x^*[i]$. Is this interpolation a linear or non-linear regression? Why?

5. What happens if you take all the x^* equal to the x? Try it in the code. Consider also the value of the matrix A. What do you observe? To avoid the problem, it is usual to add a small diagonal value to the matrices Σ_{yy} and $\Sigma_{y^*y^*}$. For instance

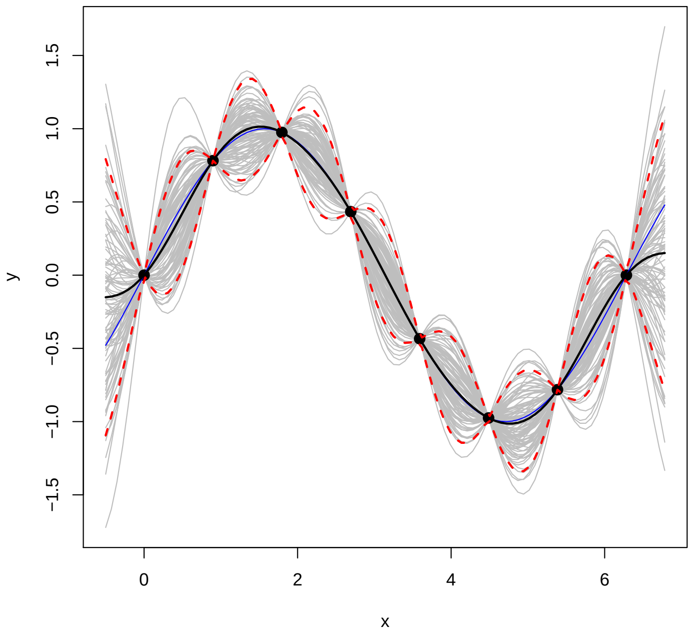

Figure 3: The result of the plots corresponding to the above algorithm.

```
$\Sigma_{yy} \leftarrow \exp(-D) + \mathrm{diag}(\epsilon, \mathrm{ncol}(D))$
```

where $\epsilon$is for instance 1e-8. Modify your code as suggested and see if it works better.

6. Actually, one can add an uncertainty (or an error) to the observed values y. This can be done by taking ϵ above as the desired variance of y. Try to take $\epsilon = 0.001$ in your code and observe what you see.

7. Implement in the code the more general kernel

$$K(x_i, x_j) = \sigma^2 \exp\left(-\frac{||x_i - x_j||^2}{2\ell^2}\right)$$

11 Hint and tips

In R you can save a plot, for instance as a pdf, by first declaring the file name

```
pdf(file=”currentPlot.pdf”,width=6,height=6)
```

Then, after the plot has been completely displayed, issue the command

```
dev.off() # to actually print the plot into the selected file
```