# Gaussian Processes

H. Rahnama
hosein.rahnama@outlook.com

**Abstract**

This is an introduction to the basics of Gaussian Processes (GP). Particulary, we are interested in their applications for regression and machine learning. The tutorial is mainly divided in two parts. In the first part, we present a step-by-step explanation of the necessary mathematics to understand GPs. For this purpose, we assume that the reader is familiar with linear algebra, probability, and multivariable calculus. In this part, we touch upon the Cholesky decomposition, the matrix inversion lemma for partitioned matrices, Multivariate Normal (MVN) Probability Density Functions (PDF) for continuous random vectors, and three main operations assosiated with MVNs, namely sampling, conditioning, and marginalization. In the seond part, we define what a GP is and learn how to do regression with GPs in the cases of noise-free and noisy data observations. Afterwards, we present a simple algorithm to do regression with GPs, investigate its time complexity and implementation. Then, we study the effect of hyperparameters of a special kernel function, well known as the Radial Basis Function (RBF), on the predictive distribution. This sets the motivation for optimizing the hyperparameters, which we shall address briefly via introducing Cross Validation (CV) and Marginal Likelihood (ML). We also talk about the methods of combining available kernels to construct new ones. Afterwards, we address a real world application of GPs for designing interactive interfaces. The paper concludes with some remarks about the connection of GPs with other models and their capability for scaling.

**keywords**. Gaussian Process, Multivariate Normal Distribution, Regression, Kernels, Hyperparameters.

## Contents

# 1 Mathematical Preliminaries

In the first part of this tutorial, we express the mathematics that we will employ in the second part. Let us first pay attention to a very useful matrix decomposition.

## 1.1 Cholesky Decomposition

This decomposition for symmetric positive definite (PD) matrices is in perfect analogy with the square root of a positive real number.

**Theorem 1.1** (Cholesky Decomposition)**.** *Let $\mathsf{A} \in \mathbb{R}^{n\times n}$ be a real square matrix. The matrix $\mathsf{A}$ is symmetric positive definite if and only if there exists a non-singular lower-triangular matrix $\mathsf{L} \in \mathbb{R}^{n\times n}$ such that $\mathsf{A} = \mathsf{L}\mathsf{L}^\top$. This is called the **Cholesky decomposition** of $\mathsf{A}$ and $\mathsf{L}$ is called the **Cholesky factor** of $\mathsf{A}$.*

From Theorem 1.1 and the fact that the determinant of a triangular matrix equals the product of its diagonal elements, it follows quickly that

**Corollary 1.2.** *If $\mathsf{A} \in \mathbb{R}^{n\times n}$ is sysmetric positive definite then its determinant is given by $\det \mathsf{A} = (\det \mathsf{L})^2 = \prod_{i=1}^n l_{ii}^2$, where $l_{ii}$ are the diagonal elements of the cholesky factor of $\mathsf{A}$.*

The Cholesky decomposition is numerically highly stable for solving symmetric positive definite linear systems of equations and is the main method of choice for computation if it is available. Below is an algorithm that remedies the computation of Cholesky decomposition. The algorithm 1 simply assumes that matrix $\mathsf{A}$ has a Cholesky decomposition and exploits the fact that

$$a_{jk} = \sum_{i=1}^{n} l_{ji}l_{ki} = \sum_{i=1}^{\min\{j,k\}} l_{ji}l_{ki} = \sum_{i=1}^{k} l_{ji}l_{ki}, \qquad k = 1,\dots,n, \qquad j = k,\dots,n \tag{1.1}$$

to calculate the columns of $L$ inductively. When $k = 1$, we observe that

$$a_{j1} = \sum_{i=1}^{1} l_{ji}l_{1i} = l_{j1}l_{11}, \qquad j = 1,\dots,n \tag{1.2}$$

Now, considering the cases $j = 1$ and $j > 1$, one concludes that

$$\begin{aligned} l_{11} &= \sqrt{a_{11}} \\ l_{j1} &= a_{j1}/l_{11}, \qquad j = 2,\dots,n, \end{aligned} \tag{1.3}$$

meaning that the first column of $L$ is computed. Suppose that we have computed the first $k-1$ columns of $L$ and we want to compute the $k$th column. Again, by considering the cases $j = k$ and $j > k$ in eq. (1.1), we obtain the following inductive equations

$$\begin{aligned} l_{kk} &= \Big(a_{kk} - \sum_{i=1}^{k-1} l_{ki}^2\Big)^{\frac{1}{2}}, \qquad & k = 2,\dots,n, \\ l_{jk} &= \Big(a_{jk} - \sum_{i=1}^{k-1} l_{ki}l_{ji}\Big)/l_{kk}, \qquad & j = k+1,\dots,n, \end{aligned} \tag{1.4}$$

where all of the terms on the right hand sides (RHS) are known according to the hypothesis of induction. Using algorithm 1, it can be easily verified that

**Corollary 1.3.** *Time complexity of computing the Cholesky decomposition of a symmetic PD matrix $A$ is of order $\mathcal{O}(\frac{1}{6}n^3)$.*

and we also have the following result

**Corollary 1.4.** *Given the Cholesky decomposition, solving the symmetric PD linear system $\mathsf{A}\mathbf{x} = \mathbf{b}$ reduces to solving the lower-triangular $\mathsf{L}\mathbf{y} = \mathbf{b}$ and upper-triangular $\mathsf{L}^\top\mathbf{x} = \mathbf{y}$ linear systems. Time complexity of solving each of these is of order $\mathcal{O}(\frac{1}{2}n^2)$.*

**Algorithm 1** Cholesky Decomposition

```
INPUT: A
OUTPUT: L
▷ Initialize the first column of $L$ with zeros.
for $j = 1$ to $n$ do
    $l_{j1} = 0$;
end for
▷ Calculate $l_{kk}$.
for $k = 1$ to $n$ do
    sum = 0;
    for $i = 1$ to $k - 1$ do
        sum = sum + $l_{ki}l_{ki}$;
    end for
    $l_{kk} = a_{kk} -$ sum;
    ▷ Check for positive definitness.
    if $l_{kk} \leq 0$ then
        $A$ is not positive definite;
        exit;
    else
        $l_{kk} = \sqrt{l_{kk}}$;
    end if
end for
▷ Calculate $l_{jk}$ for $j > k$.
for $j = k + 1$ to $n$ do
    sum = 0;
    for $i = 1$ to $k - 1$ do
        sum = sum + $l_{ki}l_{ji}$;
    end for
    $l_{jk} = (a_{jk} -$ sum$)/l_{kk}$;
end for
```

## 1.2 Matrix Inversion Lemma

Here, we shall discuss an important lemma for inverting a partioned matrix and its imediate consequences. It will be useful in dealing with MVN PDFs. We have the following theorem.

**Theorem 1.5** (Matrix Inversion Lemma). *Consider a general invertible partioned matrix* $\mathsf{M} \in \mathbb{R}^{n+m,n+m}$ *as*

$$\mathsf{M} = \begin{bmatrix} \mathsf{A} & \mathsf{U} \\ \mathsf{V} & \mathsf{B}, \end{bmatrix} \tag{1.5}$$

*where* $\mathsf{A} \in \mathbb{R}^{n,n}$ *and* $\mathsf{B} \in \mathbb{R}^{m,m}$ *are invertible,* $\mathsf{U} \in \mathbb{R}^{n,m}$ *and* $\mathsf{V} \in \mathbb{R}^{m,n}$*. Then,* $\mathsf{M}^{-1}$ *is given by*

$$\mathsf{M}^{-1} = \begin{bmatrix} (\mathsf{M}/\mathsf{B})^{-1} & -(\mathsf{M}/\mathsf{B})^{-1}\mathsf{U}\mathsf{B}^{-1} \\ -\mathsf{B}^{-1}\mathsf{V}(\mathsf{M}/\mathsf{B})^{-1} & \mathsf{B}^{-1} + \mathsf{B}^{-1}\mathsf{V}(\mathsf{M}/\mathsf{B})^{-1}\mathsf{U}\mathsf{B}^{-1} \end{bmatrix}, \tag{1.6}$$

*where* $\mathsf{M}/\mathsf{B} := \mathsf{A} - \mathsf{U}\mathsf{B}^{-1}\mathsf{V}$ *is the* ***schur complement*** *of* $\mathsf{M}$ *with respect to* $\mathsf{B}$*. We also have*

$$\mathsf{M}^{-1} = \begin{bmatrix} \mathsf{A}^{-1} + \mathsf{A}^{-1}\mathsf{U}(\mathsf{M}/\mathsf{A})^{-1}\mathsf{V}\mathsf{A}^{-1} & -\mathsf{A}^{-1}\mathsf{U}(\mathsf{M}/\mathsf{A})^{-1} \\ -(\mathsf{M}/\mathsf{A})^{-1}\mathsf{V}\mathsf{A}^{-1} & (\mathsf{M}/\mathsf{A})^{-1} \end{bmatrix}, \tag{1.7}$$

*where* $\mathsf{M}/\mathsf{A} := \mathsf{B} - \mathsf{V}\mathsf{A}^{-1}\mathsf{U}$ *is the* ***schur complement*** *of* $\mathsf{M}$ *with respect to* $\mathsf{A}$*. Furthermore, we have that*

$$\det \mathsf{M} = \det \mathsf{B} \det(\mathsf{M}/\mathsf{B}) = \det \mathsf{A} \det(\mathsf{M}/\mathsf{A}) \tag{1.8}$$

**Proof.** The idea is pretty simple. We will carry out a block diagonal Gauss-Jordan elimination on $\mathsf{M}$ and everything follows so naturally. Let us make the bottom left block zero by a block row operation as below

$$\mathsf{R}\mathsf{M} = \begin{bmatrix} \mathsf{I}_\mathsf{n} & 0 \\ -\mathsf{V}\mathsf{A}^{-1} & \mathsf{I}_\mathsf{m} \end{bmatrix} \begin{bmatrix} \mathsf{A} & \mathsf{U} \\ \mathsf{V} & \mathsf{B} \end{bmatrix} = \begin{bmatrix} \mathsf{A} & \mathsf{U} \\ 0 & \mathsf{B} - \mathsf{V}\mathsf{A}^{-1}\mathsf{U} \end{bmatrix} = \begin{bmatrix} \mathsf{A} & \mathsf{U} \\ 0 & \mathsf{M}/\mathsf{A} \end{bmatrix}. \tag{1.9}$$

Next, we shall make the top right block of $\mathsf{RM}$ zero by a column operation

$$\mathsf{RMC} = \begin{bmatrix} \mathsf{A} & \mathsf{U} \\ 0 & \mathsf{M/A} \end{bmatrix} \begin{bmatrix} \mathsf{I_n} & -\mathsf{A}^{-1}\mathsf{U} \\ 0 & \mathsf{I_m} \end{bmatrix} = \begin{bmatrix} \mathsf{A} & 0 \\ 0 & \mathsf{M/A} \end{bmatrix} := \mathsf{D} \tag{1.10}$$

Now, we note that $\mathsf{M} = \mathsf{R}^{-1}\mathsf{D}\mathsf{C}^{-1}$, so the inverse can be calculated as $\mathsf{M}^{-1} = \mathsf{C}\mathsf{D}^{-1}\mathsf{R}$. This conclusion is valid since $\mathsf{R}$, $\mathsf{C}$ and $\mathsf{D}$ are invertible. Invertibility of $\mathsf{R}$ and $\mathsf{C}$ is followed from their definition and the fact that

$$\det \mathsf{R} = \det \mathsf{C} = \det \mathsf{I_n} \det \mathsf{I}_m = 1 \tag{1.11}$$

Matrix $\mathsf{D}$ is invertible since $\mathsf{A}$ and $\mathsf{M/A}$ are invertible. Indeed, $\mathsf{A}$ is invertible by the assumptions of the theorem and invertiblity of $\mathsf{M/A}$ is followed from that of $\mathsf{A}$. Otherwise, according to eq. (1.9), the last $m$ rows of $\mathsf{RM}$ will be linearly independent, so $\mathsf{RM}$ is not invertible, which implies that $\mathsf{M}$ is not invertible, contradicting the assumptions of the theorem. Finally, we have

$$\begin{aligned}
\mathsf{M}^{-1} &= \mathsf{C}\mathsf{D}^{-1}\mathsf{R} \\
&= \begin{bmatrix} \mathsf{I_n} & -\mathsf{A}^{-1}\mathsf{U} \\ 0 & \mathsf{I_m} \end{bmatrix} \begin{bmatrix} \mathsf{A} & 0 \\ 0 & \mathsf{M/A} \end{bmatrix}^{-1} \begin{bmatrix} \mathsf{I_n} & 0 \\ -\mathsf{V}\mathsf{A}^{-1} & \mathsf{I_m} \end{bmatrix} \\
&= \begin{bmatrix} \mathsf{I_n} & -\mathsf{A}^{-1}\mathsf{U} \\ 0 & \mathsf{I_m} \end{bmatrix} \begin{bmatrix} \mathsf{A}^{-1} & 0 \\ 0 & (\mathsf{M/A})^{-1} \end{bmatrix} \begin{bmatrix} \mathsf{I_n} & 0 \\ -\mathsf{V}\mathsf{A}^{-1} & \mathsf{I_m} \end{bmatrix} \\
&= \begin{bmatrix} \mathsf{A}^{-1} & -\mathsf{A}^{-1}\mathsf{U}(\mathsf{M/A})^{-1} \\ 0 & (\mathsf{M/A})^{-1} \end{bmatrix} \begin{bmatrix} \mathsf{I_n} & 0 \\ -\mathsf{V}\mathsf{A}^{-1} & \mathsf{I_m} \end{bmatrix} \\
&= \begin{bmatrix} \mathsf{A}^{-1} + \mathsf{A}^{-1}\mathsf{U}(\mathsf{M/A})^{-1}\mathsf{V}\mathsf{A}^{-1} & -\mathsf{A}^{-1}\mathsf{U}(\mathsf{M/A})^{-1} \\ -(\mathsf{M/A})^{-1}\mathsf{V}\mathsf{A}^{-1} & (\mathsf{M/A})^{-1} \end{bmatrix},
\end{aligned}$$

which proves eq. (1.7). Furthermore, combining eq. (1.10) and eq. (1.11) we obtain

$$\det \mathsf{M} = \det \mathsf{R}^{-1}\mathsf{D}\mathsf{C}^{-1} = \det \mathsf{R}^{-1} \det \mathsf{D} \det \mathsf{C}^{-1} = \det \mathsf{D} = \det \mathsf{A} \det(\mathsf{M/A}),$$

which proves the seond equality in eq. (1.8). The proof for the remaining part of the theorem is very similar and we leave it as an exercise. Indeed, it is proved by first making the bottom left block of $\mathsf{M}$ zero by a column operation $\bar{\mathsf{C}}$ and then making the top right block of $\mathsf{M}\bar{\mathsf{C}}$ zero by a row operation $\bar{\mathsf{R}}$. ■

**Example 1.1.** An application of the above thoerem is to reduce the computational cost in certain circumtances. Consider the equality of the upper left terms in eqs. (1.7) and (1.6)

$$(\mathsf{M/B})^{-1} = \mathsf{A}^{-1} + \mathsf{A}^{-1}\mathsf{U}(\mathsf{M/A})^{-1}\mathsf{V}\mathsf{A}^{-1}, \tag{1.12}$$

which is equivalent to

$$(\mathsf{A} - \mathsf{U}\mathsf{B}^{-1}\mathsf{V})^{-1} = \mathsf{A}^{-1} + \mathsf{A}^{-1}\mathsf{U}(\mathsf{B} - \mathsf{V}\mathsf{A}^{-1}\mathsf{U})^{-1}\mathsf{V}\mathsf{A}^{-1}. \tag{1.13}$$

This is well known as the **Sherman-Morrison-Woodbury** formula. Next, let $\mathsf{A} = \Sigma$ be a diagonal matrix, $\mathsf{U} = \mathsf{V}^\top = \mathsf{X}$ and $\mathsf{B} = -\mathsf{I_m}$ to obtain

$$(\Sigma + \mathsf{X}\mathsf{X}^\top)^{-1} = \Sigma^{-1} - \Sigma^{-1}\mathsf{X}(\mathsf{I}_m + \mathsf{X}^\top\Sigma^{-1}\mathsf{X})^{-1}\mathsf{X}^\top\Sigma^{-1}, \tag{1.14}$$

where the left hand side (LHS) requires the inversion of an $n \times n$ matrix, while the RHS requires the inversion of an $m \times m$ matrix. When $n \gg m$, it is desirable to use the RHS formulation for calculations instead of the LHS.

## 1.3 Univariate Gaussian Distribution

The most widely used distribution in statistics and machine learning is the Gaussian or normal distribution. In this subsection, we address that how one cam sample from a univariate Gaussian distribution.

**Definition 1.1.** A continuous random variable $X$ is said to be a **normal** random variable, if its PDF is

$$p_X(x) = \mathcal{N}(x; \mu, \sigma) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Big(-\frac{(x-\mu)^2}{2\sigma^2}\Big). \tag{1.15}$$

The distribution is determined by $\mu$ and $\sigma$. It is easily verified that the mean is $\mathbb{E}(X) = \mu$ and the variance is $\mathbb{V}(X) = \sigma$. Furthermore, the cumulative probability function (CDF) can be shown to be

$$F_X(x) = \mathbb{P}(X \le x) = \Phi\Big(\frac{x-\mu}{\sigma}\Big), \tag{1.16}$$

with $\Phi(x)$ defined as

$$\Phi(x) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\Big(-\frac{1}{2}t^2\Big)\, \mathrm{d}t. \tag{1.17}$$

We write $X \sim N(x;\mu,\sigma)$ as an alternative notation to emphasize that $X$ is a normal random variable. Indeed, this is used to emphasize that the values $x$ taken by $X$ are **generated**, **simultaed**, or **sampled** from a Gaussian PDF denoted by $N(x;\mu,\sigma)$. For the special case $\mu = 0$ and $\sigma = 1$, the associated random variable is called **standard normal** and denoted by $Z$. Its PDF and CDF simplify to

$$\begin{aligned} p_Z(z) &= \mathcal{N}(z;0,1) = \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{z^2}{2}\Big) \\ F_Z(z) &= \mathbb{P}(Z \le z) = \int_{-\infty}^{z} p_Z(t)\,\mathrm{d}t = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} \exp\Big(-\frac{1}{2}t^2\Big)\,\mathrm{d}t = \Phi(z) \end{aligned} \tag{1.18}$$

The following lemma makes a connection between normal and standard normal random variables.

**Lemma 1.6.** *Let $X \sim \mathcal{N}(x;\mu,\sigma)$ be a normal random variable. There exists a standard normal random variable $Z \sim \mathcal{N}(z;0,1)$ such that $X = \sigma Z + \mu$.*

**Proof.** By the definition of CDF for $Z$ we have

$$F_Z(z) = \mathbb{P}(Z \le z) = \mathbb{P}\Big(\frac{X-\mu}{\sigma} \le z\Big) = \mathbb{P}(X \le \sigma z + \mu) = \Phi\Big(\frac{(\sigma z+\mu)-\mu}{\sigma}\Big) = \Phi(z).$$

Taking the derivative we obtain that $F_Z'(z) = \Phi'(z) = p_Z(z) = \mathcal{N}(z;0,1)$. ■

**Definition 1.2.** A random variable $U$ is said to be **uniform** on the interval $(a,b)$ if its PDF is given by

$$p_U(x) = \mathcal{U}(x;a,b) := \begin{cases} \frac{1}{b-a} & x \in [a,b] \\ 0 & x \notin [a,b] \end{cases}. \tag{1.19}$$

Futhermore, it can easily be shown that its CDF is

$$F_U(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & x \in [a,b] \\ 1 & x > b \end{cases}. \tag{1.20}$$

For the special case, where $a = 0$ and $b = 1$, the PDF is simplified as

$$p_U(x) = \mathcal{U}(x;0,1) := \begin{cases} 1 & x \in [0,1] \\ 0 & x \notin [0,1] \end{cases}, \tag{1.21}$$

and consequenlty, the CDF becomes

$$F_U(x) = \begin{cases} 0 & x < 0 \\ x & x \in [0,1] \\ 1 & x > 1 \end{cases}. \tag{1.22}$$

Computers have a pseudo random number generator which is capable of simulating the sampling from $\mathcal{U}(x;0,1)$. To sample from other PDFs, they employ the following important result, a proof of which is given in [1].

**Theorem 1.7** (Method of Transformations)**.** *Let $X$ be a continuous random variable, $p_X$ the PDF of $X$, and $A$ the set of possible values of $X$. For the invertible function $h : A \to \mathbb{R}$, let $Y = h(X)$ be a random variable with the set of possible values $B = h(A) = \{h(a) : a \in A\}$. Suppose that the inverse of $y = h(x)$ is the function $x = h^{-1}(y)$, which is differentiable for all values of $y \in B$. Then $p_Y$, the PDF of $Y$ is given by*

$$p_Y(y) = \begin{cases} p_X(h^{-1}(y))\,\big|(h^{-1})'(y)\big| & \forall y \in B \\ 0 & \forall y \notin B \end{cases} \tag{1.23}$$

An imediate and useful result of the above theorem is

**Corollary 1.8** (Sampling)**.** *Let $U$ be a continuous random variable with PDF $\mathcal{U}(u;0,1)$ and let $Y = F_Z^{-1}(U)$ be another random variable $Y$, where $F_Z : \mathbb{R} \to [0,1]$ is the CDF of an arbitrary random variable $Z$ that we want to sample. Assuming that $F_Z^{-1}$ is differentable on $(0,1)$ and strictly increasing, $Y$ and $Z$ have the same PDF.*

**Proof.** According to Theorem 1.7, set $X = U$, where $U \sim \mathcal{U}(x;0,1)$, and $h = F_Z^{-1}$ with $F_Z : \mathbb{R} \to [0,1]$ being the CDF of a random variable $Z$ that we want to sample. Noting that $A = [0,1]$ and $B = \mathbb{R}$, $Y = F_Z^{-1}(U)$ is a random variable with the following CDF

$$p_Y(y) = p_U(F_Z(y))\,|F_Z'(y)| = 1 \cdot |F_Z'(y)| = F_Z'(y) = p_Z(y), \qquad \forall y \in \mathbb{R}, \tag{1.24}$$

proving that $Y$ and $Z$ have the same PDF. ∎

This result is so crucial and illustrates that if we are capable of sampling from $\mathcal{U}(x;0,1)$, then we can sample from any other distribution with a differentiable strictly increasing CDF. In particular, if you set $F_Z(y) = \Phi((y-\mu)/\sigma)$, which satisfies the aforementioned requirements, then you can sample from $\mathcal{N}(x;\mu,\sigma^2)$. For this purpose, draw a sample $x$ from $\mathcal{U}(x;0,1)$ and then calculate $y = F_Z^{-1}(x)$.

## 1.4 Multivariate Gaussian Distribution

The multivariate Gaussian or Normal (MVN) distribution is the most widely used joint PDF for continuous random variables. In this subsection, we address three main operations of MVNs, including sampling, conditioning and marginalization.

**Definition 1.3.** The joint PDF of the MVN distribution in $m$ dimensions is defined as

$$p_{\mathbf{X}}(\mathbf{x}) = \mathcal{N}(\mathbf{x};\boldsymbol{\mu},\Sigma) := \frac{1}{\sqrt{\det 2\pi\Sigma}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big), \tag{1.25}$$

where $\mathbf{X}, \mathbf{x}, \boldsymbol{\mu} \in \mathbb{R}^m$ and $\Sigma \in \mathbb{R}^{m,m}$. The distribution is determined by the vector $\boldsymbol{\mu}$ and matrix $\Sigma$. It can be verified that the **mean vector** is $\boldsymbol{\mu} = \mathbb{E}(\mathbf{X})$. The matrix $\Sigma$ is called the **covariance matrix** and is required to be PD by definition. It can be shown that $\operatorname{cov}(X_i, X_j) = \sigma_{ij}$ with $\Sigma = [\sigma_{ij}]$ so $\Sigma$ is also symmetric due to the fact that $\operatorname{cov}(X_i, X_j) = \operatorname{cov}(X_j, X_i)$. This can be written in a more compact form $\operatorname{cov}(\mathbf{X}) = \Sigma$. The inverse of the covariance matrix $\Lambda = \Sigma^{-1}$ is called the **precision** matrix. When $\boldsymbol{\mu} = \mathbf{0}$ and $\Sigma = \mathsf{I}$, the associated random vector is denoted by $\mathbf{Z}$ and is called the standard normal random vector. In this case, the PDF simplifies to

$$\begin{aligned} p_{\mathbf{Z}}(\mathbf{z}) = \mathcal{N}(\mathbf{z};\mathbf{0},\mathsf{I}) &:= \frac{1}{\sqrt{2\pi}^m}\exp\Big(-\frac{1}{2}\mathbf{z}^\top\mathbf{z}\Big) = \prod_{i=1}^m \frac{1}{\sqrt{2\pi}}\exp\Big(-\frac{1}{2}z_i^2\Big) \\ &= \prod_{i=1}^m \mathcal{N}(z_i;0,1) = \prod_{i=1}^m p_{Z_i}(z_i), \end{aligned} \tag{1.26}$$

showing that in this case the PDF becomes the products of marginal PDFs. The equality is a result of marginalizatoin property of MVNs proved in Theorem 1.11.

The following is the multivariate version of the method transformations, that you a can read a proof for the bivariate case in [1].

**Theorem 1.9** (Method of Transformations)**.** *Let $\mathbf{X} \in \mathbb{R}^m$ be a continuous random vector with $p_{\mathbf{X}}(\mathbf{x})$ as its joint PDF and that its set of values lie in $A \subseteq \mathbb{R}^m$. Let $\mathbf{Y} = H(\mathbf{X})$ such that $H : A \to \mathbb{R}^m$ is a one-to-one transformation from $A$ onto $B = h(A) = \{h(a), a \in A\}$. That is for $\mathbf{x} \in A$ the system $\mathbf{y} = H(\mathbf{x})$ has a unique solution $\mathbf{y} = H^{-1}(\mathbf{x}) := G(\mathbf{x})$. Assume that the Jacobian of the transformation $\mathbf{y} = G(\mathbf{x})$ is nonzero at all points $\mathbf{y} \in B$; that is*

$$\det \mathsf{J}(\mathbf{y}) \neq 0, \qquad \forall \mathbf{y} \in B, \qquad \mathsf{J}_{ij} := \frac{\partial G_i}{\partial y_j}. \tag{1.27}$$

*The the joint PDF of random the vector $\mathbf{Y}$ is given by*

$$p_{\mathbf{Y}}(\mathbf{y}) = \begin{cases} p_{\mathbf{X}}(G(\mathbf{y}))|\det \mathsf{J}(\mathbf{y})| & \mathbf{y} \in B \\ 0 & \mathbf{y} \notin B \end{cases} \tag{1.28}$$

and an imediate result of the multivariate version of method of transformations will be the following lemma, which determines that how we can sample a random vector with MVN distribution.

**Lemma 1.10.** *Let* $\mathbf{X} \sim \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma)$ *be a normal random vector. There is a standard normal random vector* $\mathbf{Z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathsf{I})$ *such that* $\mathbf{X} = \mathsf{L}\mathbf{Z} + \boldsymbol{\mu}$, *where* $\mathsf{L}$ *is the cholesky factor of* $\Sigma$, *that is* $\Sigma = \mathsf{L}\mathsf{L}^\top$.

**Proof.** If $\mathbf{Z} = \mathsf{L}^{-1}(\mathbf{X} - \boldsymbol{\mu})$ the algebraic relation is satisfied. It remains to show that $\mathbf{Z}$ is standard normal. According to Theorem 1.9, $A = B = \mathbb{R}^m$, $\mathbf{z} = H(\mathbf{x}) = \mathsf{L}^{-1}(\mathbf{x} - \boldsymbol{\mu})$. So the inverse map and its Jacobian are $\mathbf{x} = G(\mathbf{z}) = \mathsf{L}\mathbf{z} + \boldsymbol{\mu}$ and $\mathsf{J}(\mathbf{z}) = \mathsf{L}$. Consequently, for every $\mathbf{x} \in \mathbb{R}^m$ we have

$$\begin{aligned}
p_{\mathbf{Z}}(\mathbf{z}) &= |\det \mathsf{J}(\mathbf{z})|\, p_{\mathbf{X}}(G(\mathbf{z})) = |\det \mathsf{L}|\, p_{\mathbf{X}}(\mathsf{L}\mathbf{z} + \boldsymbol{\mu}) \\
&= \frac{|\det \mathsf{L}|}{(2\pi)^{\frac{m}{2}}\sqrt{\det \Sigma}} \exp\Big(-\frac{1}{2}((\mathsf{L}\mathbf{z} + \boldsymbol{\mu}) - \boldsymbol{\mu}))^\top \Sigma^{-1}((\mathsf{L}\mathbf{z} + \boldsymbol{\mu}) - \boldsymbol{\mu})\Big) \\
&= \frac{|\det \mathsf{L}|}{(2\pi)^{\frac{m}{2}}|\det \mathsf{L}|} \exp\Big(-\frac{1}{2}(\mathbf{z}^\top \mathsf{L}^\top)(\mathsf{L}^{-\top}\mathsf{L}^{-1})(\mathsf{L}\mathbf{z})\Big) \\
&= \frac{1}{(2\pi)^{\frac{m}{2}}} \exp\Big(-\frac{1}{2}\mathbf{z}^\top \mathbf{z}\Big),
\end{aligned}$$

which completes the proof. ■

**Example 1.2.** Now, we are able to sample a random normal vector $\mathbf{X} \sim \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma)$. For this purpose, we first calculate the Cholesky factor $\mathsf{L}$ of $\Sigma$. Then, we note that $\mathbf{X} = \mathsf{L}\mathbf{Z} + \boldsymbol{\mu}$, where $\mathbf{Z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathsf{I})$. Now, we are able to sample the random strandard normal vector $\mathbf{Z} = (Z_1, \ldots, Z_m)^\top$ since by eq. (1.26) we know that $Z_i \sim \mathcal{N}(z_i, 0, 1)$.

**Theorem 1.11** (Marginalization and Conditioning)**.** *Suppose* $\mathbf{Y} \sim \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}, \Sigma)$ *is jointly Gaussian and is partitioned in two parts, where the first and second partitions have dimensions* $m$ *and* $n$*, respectively. We have*

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \end{bmatrix}, \qquad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \qquad \Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}, \qquad \Lambda := \Sigma^{-1} \begin{bmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{bmatrix}, \tag{1.29}$$

*where* $\Sigma_{21} = \Sigma_{12}^\top$ *and* $\Lambda_{21} = \Lambda_{12}^\top$. *Then, the marginal PDFs are given by*

$$\begin{aligned}
p_{\mathbf{Y}_1}(\mathbf{y}_1) &= \int_{\mathbf{y}_1 \in \mathbb{R}^m} p_{\mathbf{Y}_1,\mathbf{Y}_2}(\mathbf{y}_1, \mathbf{y}_2)\, \mathrm{d}\mathbf{y}_2 = \mathcal{N}(\mathbf{y}_1; \boldsymbol{\mu}_1, \Sigma_{11}) \\
p_{\mathbf{Y}_2}(\mathbf{y}_2) &= \int_{\mathbf{y}_2 \in \mathbb{R}^n} p_{\mathbf{Y}_1,\mathbf{Y}_2}(\mathbf{y}_1, \mathbf{y}_2)\, \mathrm{d}\mathbf{y}_1 = \mathcal{N}(\mathbf{y}_2; \boldsymbol{\mu}_2, \Sigma_{22})
\end{aligned} \tag{1.30}$$

*and the conditional PDF is given by*

$$p_{\mathbf{Y}_1|\mathbf{Y}_2}(\mathbf{y}_1|\mathbf{y}_2) = \frac{p_{\mathbf{Y}_1,\mathbf{Y}_2}(\mathbf{y}_1, \mathbf{y}_2)}{p_{\mathbf{Y}_2}(\mathbf{y}_2)} = \mathcal{N}(\mathbf{y}_1; \boldsymbol{\mu}_{1|2}, \Sigma_{1|2}), \tag{1.31}$$

*with its mean vector and covariance matrix obtained as*

$$\begin{aligned}
\boldsymbol{\mu}_{1|2} &= \boldsymbol{\mu}_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{y}_2 - \boldsymbol{\mu}_2) = \boldsymbol{\mu}_1 - \Lambda_{11}^{-1}\Lambda_{12}(\mathbf{y}_2 - \boldsymbol{\mu}_2), \\
\Sigma_{1|2} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} = \Lambda_{11}^{-1},
\end{aligned} \tag{1.32}$$

*showing that Gaussian PDFs are closed under marginalization and conditioning.*

**Proof.** The proof of this theorem is lengthy. For a detailed treatment see [2]. However, we mention an outline of the proof. To obtain the relation for marginals or conditionals carry out the following steps

1. Write the integral form of the marginal and the algebraic expression of conditional density explicitly.
2. Rewrite the integral and the expression by partitioning the precision matrix $\Lambda = \Sigma^{-1}$.
3. Use a completion-of-squares argument to write the integral and the expression in terms of quadratic forms for $\mathbf{y}_1$ and $\mathbf{y}_2$ with the corresponding sub-blocks of $\Lambda$.
4. Using the matrix inversion lemma in Theorem 1.5 to determine sub-blocks of the precision matrix $\Lambda$ in terms of the sub-blocks of the covariance matrix $\Sigma$ , argue that the resulting density is Gaussian.

This illustrates an outline for the proof. ■

Theorem 1.11 is central in working with MVN distributions. The recap is that if you want the marginals then just take corresponding block in the mean vector and covariance matrix of the joint distribution. To get the conditional, use eq. (1.32). Note that the mean vector in the conditional is linear in $\mathbf{y}_2$ and the covariance matrix is just a constant matrix independent of $\mathbf{y}_2$.

# 2 Gaussian Process

A GP is a special stochastic process (SP). So before defining what a GP is, we start by defining an SP.

**Definition 2.1.** A stochastic process is as collection of random variables. This collection can be countable or uncountable. In methematical notation, we write $\{X_\alpha\}_{\alpha\in\mathcal{X}}$ with $\mathcal{X}$ being an **index set** that labels the random variables as $X_\alpha$. The choice of the index set is arbitrary. Some choices that usually appear in applications are $\mathcal{X} \subseteq \mathbb{N}$ and $\mathcal{X} \subseteq \mathbb{R}^m$ with $m \geq 1$. Stochastic processes are used for modeling states of random evolution of a system with each random variable $X_\alpha$ corresponding to a specific random state labeled by $\alpha$.

**Example 2.1.** A famous example of an SP is the discrete-time Markov chain. It is defined as the set of random variables $\{X_t\}_{t\in\mathbb{N}}$ with the **Markov property**, namely that the probability of moving to the next state depends only on the present state and not on the previous states formulated as below

$$\mathbb{P}(X_{n+1} = x_{n+1}|X_n = x_n, \dots, X_1 = x_1) = \mathbb{P}(X_{n+1} = x_{n+1}|X_n = x_n).$$

The possible values of $X_i$ form a countable set $S$ called the **state space** of the chain. As a concrete example, the number of daily car accidents from a origin in time can be modeled as a Markov chain. Indeed, we don't expect the number of tommorow's car accidents to depend on the number of car accidents on yesterday or one week ago or even today. In other words, the future only depends on the present and not the past.

## 2.1 What is a GP?

We are now ready to introduce a Gaussian Process.

**Definition 2.2.** A Gaussian process is a collection of random variables $\mathscr{C} = \{f(\mathbf{x})\}_{\mathbf{x}\in\mathcal{X}}$, any finite number of which have a joint Gaussian PDF. The process is determined by its **mean function** $\mu : \mathcal{X} \to \mathbb{R}$ and **kernel** or **covariance function** $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ to specify the expected value and covariance of the random variables in the set $\mathscr{C}$ as follows

$$\begin{aligned} \mu(\mathbf{x}) &= \mathbb{E}(f(\mathbf{x})), && \forall \mathbf{x} \in \mathcal{X}, \\ \kappa(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\big((f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}')\big) = \operatorname{cov}(f(\mathbf{x}), f(\mathbf{x}')), && \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}. \end{aligned} \tag{2.1}$$

We use the following notation to emphasize that the random variable $f(\mathbf{x})$ is taken from a GP

$$f(\mathbf{x}) \leftarrow \mathrm{GP}(\mu(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')), \tag{2.2}$$

and by definition its PDF will be

$$f(\mathbf{x}) \sim \mathcal{N}\big(t; \mu(\mathbf{x}), \kappa(f(\mathbf{x}), f(\mathbf{x}))\big). \tag{2.3}$$

**Example 2.2** (Consistency Requirement)**.** Suppose that we select $n$ random variables from a GP $\mathscr{C} = \{f(\mathbf{x})\}_{\mathbf{x}\in\mathcal{X}}$ and for *notational convenience* we label them as $Y_i = f(\mathbf{x}_i)$ with $i = 1, 2, \dots, n$. According to the definition of the GP, we know that $\mathbf{Y} \sim \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}, \Sigma)$ with

$$\begin{aligned} \mu_i &= \mathbb{E}(Y_i) = \mathbb{E}(f(\mathbf{x}_i)) = \mu(\mathbf{x}_i), && i = 1, \dots, n \\ \sigma_{ij} &= \operatorname{cov}(Y_i, Y_j) = \operatorname{cov}(f(\mathbf{x}_i), f(\mathbf{x}_j)) = \kappa(\mathbf{x}_i, \mathbf{x}_j), && i = 1, \dots, n, \qquad j = 1 \dots, n, \end{aligned} \tag{2.4}$$

where $\boldsymbol{\mu} = (\mu_i)$ and $\Sigma = [\sigma_{ij}]$. So the mean vector and the covariance matrix of the joint PDF $\mathcal{N}(\mathbf{y}; \boldsymbol{\mu}, \Sigma)$ are determined by the mean function and covariance function of the GP. Now partition the random vector $\mathbf{Y}$ as $(\mathbf{Y}_1, \mathbf{Y}_2)$ with $\mathbf{Y}_1 = \mathbf{Y}_{1:r}$ and $\mathbf{Y}_2 = \mathbf{Y}_{r+1:n}$. From the marginalization property of MVNs in Theorem 1.9 we know that $\mathbf{Y}_1 \sim \mathcal{N}(\mathbf{y}_1; \boldsymbol{\mu}_1, \Sigma_{11})$ and $\mathbf{Y}_2 \sim \mathcal{N}(\mathbf{y}_2; \boldsymbol{\mu}_2, \Sigma_{22})$, where $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_{1:r}$, $\Sigma_{11} = \Sigma_{1:r,\,1:r}$ and $\boldsymbol{\mu}_2 = \boldsymbol{\mu}_{r+1:n}$, $\Sigma_{22} = \Sigma_{r+1:n,\,r+1:n}$. The natural question to ask is that do we get the same result if we had initially chosen $\mathbf{Y}_1$ or $\mathbf{Y}_2$ from the GP. The answer is YES, since by eq. (2.4) we observe that everything remains the same except for the range of indices $i$ and $j$. In other words, examination of a larger set of variables does not change the distribution of the smaller set. This is well known as the **consistency requirement**. To recap, a GP is well-defined thanks to the marginalization property of MVNs. Note that the consistency requirement would be lost if we had used a precision function to determine the precision matrix of a finite number of selected random variables from the collection $\mathscr{C}$. In that case, the definition is invalid since we could get two different PDFs for the same random vector because by the matrix inversion lemma we know that $\Lambda_{11} = (\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1}$, implying that the examination of a larger set does influence the distribution of the smaller set.

**Example 2.3.** If the index set $\mathcal{X}$ is finite, e.g. $\mathcal{X} = \{1, 2, \dots, n\}$, then the GP is reduced to an MVN distribution. So GP is a generalization of MVN distribution to infinitely many random variables.

## 2.2 Probabilistic View of Regression

A typtical regression problem has the following elements.

**Definition 2.3** (Inputs, Outputs, Dataset)**.** In the regression problem, we have a training set $\mathcal{D}$ of $n$ observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, \dots, n\}$, where $\mathbf{x} \in \mathbb{R}^d$ denotes an input vector and $y \in \mathbb{R}$ denotes the output or **target** (dependent) variable. The column vector inputs $\mathbf{x}_i$ for all $n$ cases are aggregated in the **design matrix** $\mathsf{X} \in \mathbb{R}^{n\times d}$ so its rows are $\mathsf{X}_{i,.} = \mathbf{x}_i^\top$. The targets are collected in the vector $\mathbf{y} \in \mathbb{R}^n$. It is also customary to write $\mathcal{D} = (\mathsf{X}, \mathbf{y})$.

We shall take a probabilistic approach to solve the regression problem. We assume that the **observed** targets $\mathbf{y}_1$ and **unobserved** targets $\mathbf{y}_2$ are samples drawn from a set of random variables belonging to a GP, while the inputs are exactly determined and do not have any source of randomness. Now, the question is that if we look at new input points $\mathbf{x}_2$, what the conditional PDF $p_{\mathbf{Y}_2|\mathbf{Y}_1}(\mathbf{y}_2|\mathbf{y}_1)$ is. More intuitively, we are interested in knowing the odds of seeing the target values $\mathbf{y}_2$ at new input points $\mathbf{x}_1$, given that we have seen target values $\mathbf{y}_1$ at input pionts $\mathbf{x}_1$. Note that the subscripts 1 and 2 refer to the **training** and **test** data.

To take a step futher, we should present our **prior believes** about the GP that generates the target values. Consequently, we should determine the associated mean function and kernel. For simplicity in our calculations, let us assume that the mean function of the GP is zero, so $\mu(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathcal{X}$. Even with this simplifying assumption, we shall see that the model's predictive power is good enough. For a more general mean function formulation consult [3, Section 2.7]. Note that the index set $\mathcal{X} \subseteq \mathbb{R}^d$ is now just the input domain. The only thing that remains to specify is the kernel function.

**Definition 2.4.** The Radial Basis Function (RBF) or the Squared Exponential (SE) kernel is defined as

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sigma_y \exp\Big(-\frac{1}{2\ell^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\Big), \tag{2.5}$$

where $\sigma_y$ is a **hyperparameter** representing the amplitude of the covariance and $\ell$ is also a hyperparameter known as the length-scale. The RBF kernel encodes the intuition that if $\mathbf{x}_i$ and $\mathbf{x}_j$ are close to each other in the input space $\mathcal{X}$, then their correponding target values $y_i$ and $y_j$ are samples drawn from two highly correlated random variables $Y_i = f(\mathbf{x}_i)$ and $Y_j = f(\mathbf{x}_j)$. As the distance between the inputs increases, the correlation between $Y_i$ and $Y_j$ decreases exponentially. Pay attention to the fact that the covariance between *outputs* is written as a function of *inputs.*

There are a wide range of other kernels that can be used. However, it should be noted that the kernel must have specific properties to guarantee producing PD covariance matrices. It can be proved that convariance matrices produced by the RBF kernel are PD. For more details on the kernels refer to [3, Chapter 4].

### 2.2.1 Noise-Free Prediction

Now, based on our assumption that the target values are samples drawn from random variables that come from a GP, we can write the following

$$p_{\mathbf{Y}_1,\mathbf{Y}_2}(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix}; \begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12}\\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right) \sim \begin{bmatrix}\mathbf{Y}_1\\ \mathbf{Y}_2\end{bmatrix} \tag{2.6}$$

An intuitive way to think about GPs is that they are distributions over functions. To see this, think of function values as vectors of infinite length. In detail, we choose an *arbitrary* $m$ number of input points $\mathsf{X}_2 \in \mathbb{R}^{m\times d}$ with $[\mathsf{X}_2]_{i,.} = \mathbf{x}_{2,i}^\top$ and write out the corresponding covariance matrix using eqs. (2.4) and (2.5) elementwise. Next, we generate a random Gaussian vector $\mathbf{y}_2$ with this covariance matrix denoted by $\Sigma_{22}$ as follows

$$p_{\mathbf{Y}_2}(\mathbf{y}_2) = \mathcal{N}(\mathbf{y}_2; \mathbf{0}, \Sigma_{22}) \sim \mathbf{Y}_2. \tag{2.7}$$

Indeed, this is a sample drawn from the **prior distribution** as we have not used our knowledge of the observed targets. The method of sampling from an MVN was described in example 1.2. You can visit [4] for nice visualization of sampling from prior distributions with different kernels and hyperparameters. Returning to our original question, we use Theorem 1.11 to compute $p_{\mathbf{Y}_2|\mathbf{Y}_1}(\mathbf{y}_2|\mathbf{y}_1)$ as below

$$p_{\mathbf{Y}_2|\mathbf{Y}_1}(\mathbf{y}_2|\mathbf{y}_1) = \mathcal{N}(\mathbf{y}_2; \Sigma_{21}\Sigma_{11}^{-1}\mathbf{y}_1, \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}) \tag{2.8}$$

which is called the **noise-free** predictive or **posterior** distribution. Note that we have interchanged the role of subscripts 1 and 2 in using Theorem 1.11. The *prior* and posterior terminology come from the well-known **Baysian Inference** since by the Bayes rule we know that

$$p_{\mathbf{Y}_2|\mathbf{Y}_1}(\mathbf{y}_2|\mathbf{y}_1) = \frac{p_{\mathbf{Y}_2,\mathbf{Y}_1}(\mathbf{y}_2, \mathbf{y}_1)}{p_{\mathbf{Y}_1}(\mathbf{y}_1)} = \frac{p_{\mathbf{Y}_1|\mathbf{Y}_2}(\mathbf{y}_1|\mathbf{y}_2)p_{\mathbf{Y}_2}(\mathbf{y}_2)}{p_{\mathbf{Y}_1}(\mathbf{y}_1)}, \tag{2.9}$$

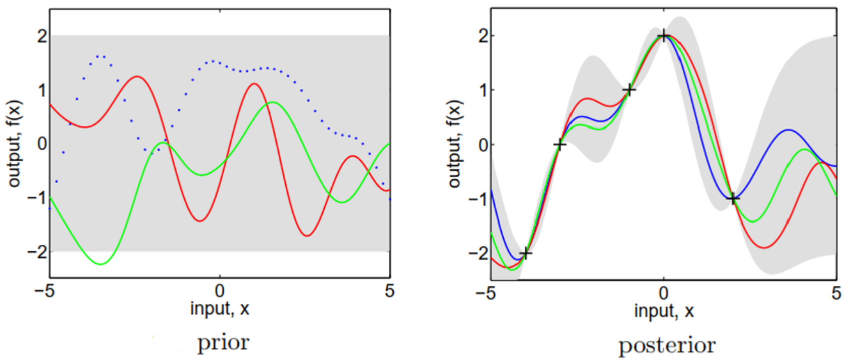

**Figure 1.** The left figure shows three functions drawn at random from a GP prior. The dots indicate values of $y_{2,i}$ actually generated. The right figure depicts functions that have been drawn as lines by joining a large number evaluated pionts. Notice that the functions look smooth. The middle figure depicts three random functions drawn from the posterior. The + signs are five noise free observation indicated. In both plots the shaded area represents the pointwise mean $\mu_{1|2}$ plus and minus $2\sigma_{1|2}$ known as the 95 percent **confidence region**. Pay attention to the fact that the variance vanishes rapidly near the observed data points and increase as we get away from them. [3].

which is usually seen as

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood or evidence}}, \tag{2.10}$$

whose interpretation is that our prior believes combined with our observations (likelihood) construct our posterior predictions. Eq. (2.8) is all we need for prediction. According to this equation, the posterior mean is a linear function of the observed targets $\mathbf{y}_1$, while the posterior covariance matrix is constant. Fig. 1 summarizes out achievements so far.

### 2.2.2 Noisy Prediction

It is typical for more realistic modelling situations that we do not have access to function values themselves, but only noisy versions of them. Consequenlty, it is reasonabe to write

$$\mathbf{Z}_1 = \mathbf{Y}_1 + \boldsymbol{\mathcal{E}}, \tag{2.11}$$

where $\boldsymbol{\mathcal{E}} \in \mathbb{R}^n$ is a random noise vector. Assume that $\mathbf{Y}_1$ and $\boldsymbol{\mathcal{E}}$ are independent random vectors and that the noise random vector is Gaussian $p_{\boldsymbol{\mathcal{E}}}(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \sigma_\epsilon \mathsf{I})$. It then follows that

$$\begin{aligned} \mathbb{E}(Z_{1,i}) &= \mathbb{E}(Y_{1,i}) + \mathbb{E}(\mathcal{E}_{1,i}) = 0 + 0 = 0, \\ \operatorname{cov}(Z_{1,i}, Z_{1,j}) &= \operatorname{cov}(Y_{1,i}, Y_{1,j}) + \operatorname{cov}(\mathcal{E}_i, \mathcal{E}_j) + 2\operatorname{cov}(Y_{1,i}, \mathcal{E}_j) = \kappa(\mathbf{x}_{1,i}, \mathbf{x}_{1,j}) + \sigma_\epsilon^2 \delta_{ij}, \end{aligned} \tag{2.12}$$

where $\delta_{ij}$ is the Kronecker's delta. The first line follows from the linearity of expectation and the second line follows from bilinearity of the covariance and independence of $\mathbf{Y}_1$ and $\boldsymbol{\mathcal{E}}$. Futhermore, with a little bit of work, it can be shown that the associated joint PDF will be [5, Sectoin 4.3]

$$p_{\mathbf{Z}_1,\mathbf{Y}_2}(\mathbf{z}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix}\mathbf{z}_1 \\ \mathbf{y}_2\end{bmatrix}; \begin{bmatrix}\mathbf{0} \\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\Sigma_{11} + \sigma_\epsilon^2 \mathsf{I} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right) \sim \begin{bmatrix}\mathbf{Z}_1 \\ \mathbf{Y}_2\end{bmatrix} \tag{2.13}$$

and the predictive distribution becomes

$$p_{\mathbf{Y}_2|\mathbf{Z}_1}(\mathbf{y}_2|\mathbf{z}_1) = \mathcal{N}(\mathbf{y}_2; \Sigma_{21}(\Sigma_{11} + \sigma_\epsilon^2 \mathsf{I})^{-1}\mathbf{z}_1, \Sigma_{22} - \Sigma_{21}(\Sigma_{11} + \sigma_\epsilon^2 \mathsf{I})^{-1}\Sigma_{12}). \tag{2.14}$$

In comparison with eq. (2.8), we have replaced $\Sigma_{11}$, $\mathbf{Y}_1$, $\mathbf{y}_1$ with $\Sigma_{11} + \sigma_\epsilon^2 \mathsf{I}$, $\mathbf{Z}_1$, $\mathbf{z}_1$, respectively.

## 2.3 GP Algorithm

Let us simplify our formulation of eq. (2.14) a little bit. In the case that there is only one test point we introduce the following notation for the sub-blocks of the covariance matrix

$$\mathsf{K} := \Sigma_{11}, \qquad \mathbf{k} := \Sigma_{12} = \Sigma_{21}^\top, \qquad k := \Sigma_{22}, \tag{2.15}$$

**Algorithm 2** Gaussian Process

```
    INPUT: $\mathsf{X}_1$, $\mathbf{y}_1$, $\kappa, \sigma_\epsilon$, $\mathbf{x}_{2,1}$
    OUTPUT: $\mu_{1|2}$, $\sigma_{1|2}$
 1: ▷ Calculate $\mathsf{K}$.
 2: for $i = 1$ to $n$ do
 3:     for $j = 1$ to $n$ do
 4:         $\mathsf{K}_{ij} = \kappa(\mathbf{x}_{1,i}, \mathbf{x}_{1,j})$;
 5:     end for
 6: end for
 7: ▷ Calculate $\mathbf{k}$.
 8: for $i = 1$ to $n$ do
 9:     $k_i = \kappa(\mathbf{x}_{1,i}, \mathbf{x}_{2,1})$;
10: end for
11: ▷ Calculate Cholesky decomposition.
12: $\mathsf{L} = \text{Cholesky}(\mathsf{K} + \sigma_\epsilon^2 \mathsf{I})$;
13: ▷ Calculate $\mu_{1|2}$.
14: Solve $\mathsf{L}\mathbf{b} = \mathbf{y}_1$;
15: Solve $\mathsf{L}^\top \mathbf{a} = \mathbf{b}$;
16: $\mu_{1|2} = \mathbf{k}^\top \mathbf{a}$;
17: ▷ Calculate $\sigma_{1|2}$.
18: Solve $\mathsf{L}\mathbf{c} = \mathbf{k}$;
19: $\sigma_{1|2} = k - \mathbf{c}^\top \mathbf{c}$;
```

and the resulting simplifications follow

$$\mathbf{Y}_2 = Y_2, \qquad \mathbf{y}_2 = y_2, \qquad \boldsymbol{\mu}_{1|2} = \mu_{1|2} \qquad \Sigma_{1|2} = \sigma_{1|2}. \tag{2.16}$$

Then, the posterior PDF in eq. (2.14) reduces to a univariate Gaussian

$$p_{Y_2|\mathbf{Y}_1}(y_2|\mathbf{y}_1) = \mathcal{N}(y_2; \mathbf{k}^\top(\mathsf{K} + \sigma_\epsilon^2 \mathsf{I})^{-1}\mathbf{y}_1, k - \mathbf{k}^\top(\mathsf{K} + \sigma_\epsilon^2 \mathsf{I})^{-1}\mathbf{k}), \tag{2.17}$$

and the posterior mean and variance will be

$$\begin{aligned} \mu_{1|2} &= \mathbf{k}^\top(\mathsf{K} + \sigma_\epsilon^2 \mathsf{I})^{-1}\mathbf{y}_1, \\ \sigma_{1|2} &= k - \mathbf{k}^\top(\mathsf{K} + \sigma_\epsilon^2 \mathsf{I})^{-1}\mathbf{k}. \end{aligned} \tag{2.18}$$

Noting that $\mathsf{K} + \sigma_\epsilon^2 \mathsf{I}$ is symmetric PD, using the Cholesky decomposition, we get $\mathsf{K} + \sigma_\epsilon^2 \mathsf{I} = \mathsf{L}\mathsf{L}^\top$. Substituting this into the new formulation we obtain

$$\begin{aligned} \mu_{1|2} &= \mathbf{k}^\top \mathsf{L}^{-\top}\mathsf{L}^{-1}\mathbf{y}_1, \\ \sigma_{1|2} &= k - (\mathsf{L}^{-1}\mathbf{k})^\top(\mathsf{L}^{-1}\mathbf{k}). \end{aligned} \tag{2.19}$$

Using eq. (2.19), Algorithm 2 addresses a simple method for calculating the predictive mean, variance, and hence the PDF in (2.17). It should be noted that we avoid computing the inverse directly as it is considered problematic from a numerical perpective [6, Section 4.7]. The main work is to compute the Cholesky decomposition, which takes $\mathcal{O}(\frac{1}{6}n^3)$ time. Solving the upper and lower-triangular linear systems in lines 14, 15, and 15 take $\mathcal{O}(\frac{1}{2}n^2)$ time. The dot produtcs in lines 16 and 19 needs $\mathcal{O}(n)$ time and the vector sum in line 19 requires $\mathcal{O}(n)$. Assuming that we use an SE kernel (RBF), calculating $\mathsf{K}$ takes a work of $\mathcal{O}(n^2 d)$ and calculating $\mathbf{k}$ requires $\mathcal{O}(nd)$. Summing all these up, the running time $T(n, d)$ of the GP algorithm is of order

$$T(n, d) \in \mathcal{O}\Big(\frac{1}{6}n^3 + n^2 d + \frac{3}{2}n^2 + nd + n\Big), \tag{2.20}$$

where $n$ is the size of our dataset and $d$ is the number of features. In the case of $n \gg d$, this simplifies to

$$T(n, d) \in \mathcal{O}\Big(\frac{1}{6}n^3\Big). \tag{2.21}$$

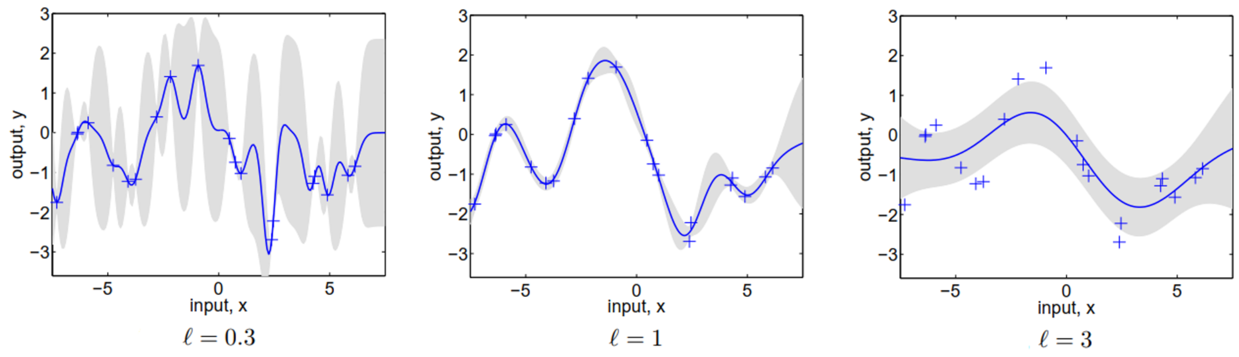

**Figure 2.** In all three figures + signs show the training data and the solid blue line is the posterior mean. The figures show GPs with SE kernels but different hyper-parameters fit to 20 noisy observations. In the left figure, the hyperparameters are $(\ell, \sigma_y, \sigma_\epsilon) = (0.3, 1.08, 0.00005)$ that corresponds to a short length-scale and as we see the wiggly looking mean oscillates so rapidly that it has fit almost all training data, ringing the bell for possible overfitting . In the middle figure, the hyperparameter are $(\ell, \sigma_y, \sigma_\epsilon) = (1, 1, 0.1)$ that corresponds to a medium length-scale. The oscillations of the mean are reduced in comparison with the left figure. This is in agreement with our intuition that the length-scale represents a neighborhood radius that you can move in the input space before the target values change drastically. In the right figure, the hyper parameters are $(\ell, \sigma_y, \sigma_\epsilon) = (3, 1.16, 0.89)$, which corresponds to a long length-scale. As we see, the mean varies so slowly and has not fit many of the training data, showing that the model's bias has increased significantly. In all of the figures, if the $x$-axis were extended one would observe that the width of the confidence region $4\sigma_{1|2}$ remains constant and reaches the value $4\sigma_y$. You can get more sense about this phenomenon by looking at eq. (2.19) [3].

## 2.4 Kernels and Hyperparameters

Typically, the kernel functions that we use will have some free parameters, known as hyperparameters that are not determined during the learning process. According to eqs. (2.5) and (2.17), the hyperparameters for a noisy GP prediction with the SE kernel are the length-scale $\ell$, the signal variance $\sigma_y$, and the noise variance $\sigma_\epsilon$. The hyperparameter $\sigma_y$ controls the amplitude of the covariance between the random variables $f(\mathbf{x})$ and $f(\mathbf{x}')$. $\sigma_\epsilon$ determines the amplitude of the variance of the noise $\mathcal{E}_i$. The hyperparameter $\ell$ determines a neighborhood radius around an input point $\mathbf{x}$ such that for every other input point $\mathbf{x}'$ lying in that neighborhood the corresponding random variables $f(\mathbf{x})$ and $f(\mathbf{x}')$ have a high correlation. Fig. 2 illustrates the effect of this hyperparameter on the predictive distribution. The overall message of the figure is that the hyperparameters can significantly affect the posterior distribution. This gives rise to the question of how should we choose these hyperparameters. In the following, we address two methods for tunning hyperparameters, but this can be studied in more depth by consulting the Bible of GPs [3, Chapter 5].

### 2.4.1 Cross Validation

In this sub-section, we consider how to use methods of cross-validation (CV) for determining hyperparameters. The basic idea is to split the training set into two disjoint sets, one which is actually used for training, and the other, the **validation set**, which is used to monitor performance, as shown in fig. 3. The performance on the validation set is used as a measure for the **generalization error** and determining hyperparameters is done based on this. In practice a drawback of this method is that only a fraction of the full data set can be used for training, and that if the validation set is small, the performance estimate obtained may have large variance.

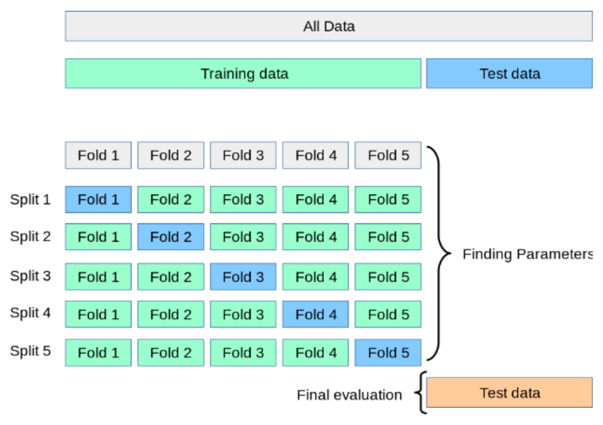

**Figure 3.** The figure illustrates a 5-fold cross validation [7].

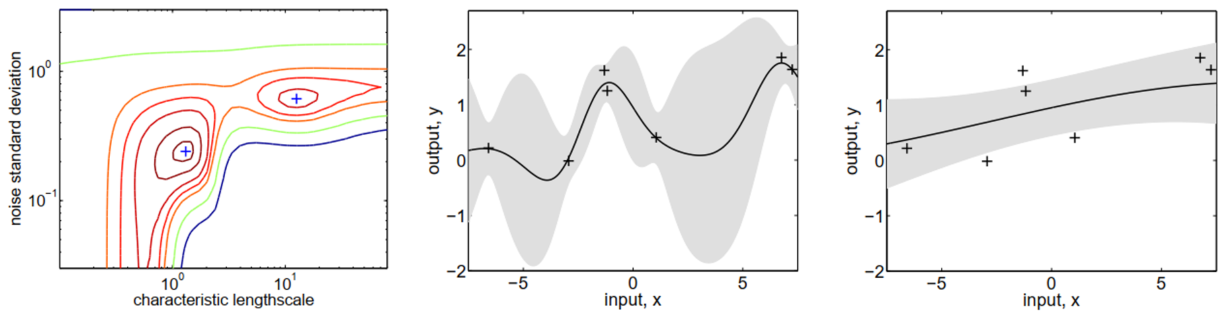

**Figure 4.** The left figure shows the marginal likelihood as a function of the hyperparameters $\ell$ and $\sigma_\epsilon$. Signal's standard deviation is chosen to be $\sigma_y^2 = 1$ for a dataset of 7 observations shown by + signs in the other two figures. There are two local optima, indicated with + signs in the left figure. The middle figure shows the function corresponding to the lower left local maximum that has the hyperparameters $(\ell, \sigma_\epsilon^2) = (1, 0.2)$ and is indeed the global maximum. This is quite *wiggly* and has low noise and short length-scale. The function corresponding to the top right local maximum has the hyperparameters $(\ell, \sigma_\epsilon^2) = (10, 0.8)$. This is quite smooth and has high noise and long length-scale [3, Section 5.4.1]. With only 7 data points, there is not enough evidence to confidently decide which is more reasonable, although the more oscillating model (middle figure) has a marginal likelihood that is about 60 percent higher than the simpler model (right figure)[5, Section 15.2.4].

To minimize these problems, CV is almost always used in the $k$-**fold** setting. In $k$-fold CV data is split into $k$ disjoint, equally sized subsets; validation is done on a single subset and training is done using the union of the remaining $k-1$ subsets, the entire procedure being repeated $k$ times, each time with a different subset for validation. Thus, a large fraction of the data can be used for training, and all cases appear as validation cases. The price is that $k$ models must be trained instead of one. Typical values for $k$ are in the range 3 to 10. An extreme case of $k$-fold cross-validation is obtained for $k = n$, the number of training data points, also known as **Leave-One-Out Cross-Validation** (LOO-CV). Often the computational cost of LOO-CV (training $n$ models) is high. Cross-validation can be used with any loss function. Although the **squared error loss** is by far the most common for regression [3, Chapter 5].

To summarize, you should consider discrete values for each hyperparameter and construct a **search grid** for each split shown in fig. 3. Then, you calculate the squared error loss for each point of the grid in a fixed split. Repeating this for all of the splits, you obtain $k$ different squared error loss for each point of the grid. Finally, you can choose that value of the hyperparameters that results in the minimum averaged squared error loss over all the $k$ values.

### 2.4.2 Marginal Likelihood

In this method, we look for those hyperparameters that maximize the probability of seeing the training data, i.e. maximizing the marginal likelihood (ML) $p_{\mathbf{Z}_1}(\mathbf{z}_1)$. This is indeed based on the **frequentist** view of statistics. From (2.13), we know that

$$p_{\mathbf{Z}_1}(\mathbf{z}_1) = \mathcal{N}(\mathbf{z}_1; \mathbf{0}, \Sigma + \sigma_\epsilon \mathsf{I}) = \mathcal{N}(\mathbf{z}_1; \mathbf{0}, \mathsf{K}_z), \tag{2.22}$$

where we have used the definition $\mathsf{K}_z := \mathsf{K} + \sigma_\epsilon^2 \mathsf{I}$. If we denote the vector of hyperparameters in a GP model by $\boldsymbol{\theta} \in \mathbb{R}^h$, then the ML method looks for the following maximizer

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \mathbb{R}^h}{\arg\max}\, p_{\mathbf{Z}_1}(\mathbf{z}_1; \boldsymbol{\theta}) = \underset{\boldsymbol{\theta} \in \mathbb{R}^h}{\arg\max}\, \log p_{\mathbf{Z}_1}(\mathbf{z}_1; \boldsymbol{\theta}), \tag{2.23}$$

where the second equality follows because the natural logrithm is an strictly increasing function and does not change the optimizer. The term *marginal* refer to the fact that the likelihood $p_{\mathbf{Z}_1}(\mathbf{z}_1; \theta)$ can be obtained by perfoming marginalization on the PDF given in (2.13). This is sometimes refered as **emprical Bayes** since pure Baysians believe that this method is fallacious. They argue that when we solve the optimization problem in (2.23) then the hyperparameters $\boldsymbol{\theta}$ will depend on the seen data $\mathbf{y}_1$; however, all of our calculation was based on the fact that they are independent of the observed data [8, Lecture 9]. For this reason, more complicated methods such as Monte Carlo (MC) methods have been developed to optimize the hyperparametrs [9, Section 45.5]. With this in mind, ML is usually done in practice and the numerical results are satisfactory.

To solve the nonlineaer unconstrained optimization problem in eq. (2.23), one needs to calculate the gradient. Feeding the result into any gradient solver for such problems could find the solution. According to (2.13), natural logarithm of the ML (log-ML) will be

$$\log p_{\mathbf{Z}_1}(\mathbf{z}_1; \theta) = \log \mathcal{N}(\mathbf{z}_1; \mathbf{0}, \mathsf{K}_z) = -\frac{1}{2}\mathbf{z}_1^\top \mathsf{K}_z^{-1} \mathbf{z}_1 - \frac{1}{2}\log\det \mathsf{K}_z - \frac{n}{2}\log 2\pi, \tag{2.24}$$

To calculate the gradient, we need the following lemma

**Lemma 2.1.** *Suppose that $\mathsf{K}(\boldsymbol{\theta}) \in \mathbb{R}^{n,n}$ is an invertible matrix whose elements depend on the vector $\boldsymbol{\theta} \in \mathbb{R}^h$. The the following matrix derivative identities are valid*

$$\partial_{\theta_i} \mathsf{K}^{-1} = -\mathsf{K}^{-1} \partial_{\theta_i} \mathsf{K}\ \mathsf{K}^{-1}, \qquad \partial_{\theta_i} \log \det \mathsf{K} = \operatorname{tr}(\mathsf{K}^{-1} \partial_{\theta_i} \mathsf{K}), \qquad i = 1, \dots, h, \tag{2.25}$$

*where $\partial_{\theta_i} \mathsf{K}$ is an elementwize derivative with respect to $\theta_i$ and* tr *is the trace of a matrix.*

Using eq. (2.25), computing the gradient of log-ML in eq. (2.24) leads to

$$\begin{aligned}
\partial_{\theta_i} \log p_{\mathbf{Z}_1}(\mathbf{z}_1; \theta) &= \frac{1}{2} \mathbf{z}_1^\top \mathsf{K}_z^{-1} \partial_{\theta_i} \mathsf{K}_z\ \mathsf{K}_z^{-1} \mathbf{z}_1 - \frac{1}{2} \operatorname{tr}(\mathsf{K}_z^{-1} \partial_{\theta_i} \mathsf{K}_z) \\
&= \frac{1}{2} \mathbf{a}^\top \partial_{\theta_i} \mathsf{K}_z\ \mathbf{a} - \frac{1}{2} \operatorname{tr}(\mathsf{K}_z^{-1} \partial_{\theta_i} \mathsf{K}_z) \\
&= \frac{1}{2} \operatorname{tr}(\mathbf{a}^\top \partial_{\theta_i} \mathsf{K}_z\ \mathbf{a}) - \frac{1}{2} \operatorname{tr}(\mathsf{K}_z^{-1} \partial_{\theta_i} \mathsf{K}_z) \\
&= \frac{1}{2} \operatorname{tr}(\mathbf{a}\mathbf{a}^\top \partial_{\theta_i} \mathsf{K}_z) - \frac{1}{2} \operatorname{tr}(\mathsf{K}_z^{-1} \partial_{\theta_i} \mathsf{K}_z) \\
&= \frac{1}{2} \operatorname{tr}((\mathbf{a}\mathbf{a}^\top - \mathsf{K}_z^{-1}) \partial_{\theta_i} \mathsf{K}_z), \\
&= \frac{1}{2} \operatorname{tr}\Big(((\mathsf{K}_z^{-1} \mathbf{z}_1)(\mathsf{K}_z^{-1} \mathbf{z}_1)^\top - \mathsf{K}_z^{-1}) \partial_{\theta_i} \mathsf{K}_z\Big),
\end{aligned} \tag{2.26}$$

where we used the fact that the trace of a scalar equals the scalar itself along with the linearity of the trace and the general identity $\operatorname{tr}(\mathsf{AB}) = \operatorname{tr}(\mathsf{BA})$. The computational cost for the ML in (2.26) is dominated by computing the inverse $\mathsf{K}_z^{-1}$. More specifically, since $\mathsf{K}_z$ is symmetric PD, its inverse can be computed by first calculating the Cholesky decomposition and solving $n$ resulting upper- and lower-triangular linear systems, which takes a work of $\mathcal{O}(n^3)$.

Fig. 4 illustrates an example of finiding the optimal values for hyperparameters $\boldsymbol{\theta} = (\ell, \sigma_\epsilon)^\top$ of a GP model with an SE kernel. We observe that there may exist several local optimums as the marginal likelihood is not a convex function.

#### 2.4.3 Constructing New Kernels

As described earlier, the power of GP lies in the choice of the kernel function. This property allows us to introduce domain knowledge into the process and lends GPs their flexibility to capture trends in the training data. For example, by choosing a suitable length-scale for the RBF kernel, we can control how smooth and oscillating the resulting function will be. A big benefit that kernels provide is that they can be combined together, resulting in a more specialized kernel. The decision of which kernel to use is highly dependent on prior knowledge about the data, e.g. if certain characteristics are expected [4].

The only constraint on our choice of covariance function is that it must generate a PD covariance matrix for any set of input points. In this connection, we have following theorem

**Theorem 2.2.** *The sum and product of two kernels is a kernel.*

For the proof of this theorem, a wide range of kernel functions, and other possibilities of combining kernels see [3, Section 4.2]. For example, the sum property can be used to add RBF kernels with different characteristic length-scales and product rule can be employed to produce trending functions with varying oscillations.

Fig. 5 depicts combinations of a linear and a periodic kernel via summation and multiplication. The two aforementioned kernels are defined as

$$\begin{aligned}
\kappa_{\text{per}}(x, x') &= \sigma_y \exp\Big(-\frac{2}{\ell^2} \sin^2\Big(\frac{\pi(x - x')}{p}\Big)\Big), \\
\kappa_{\text{lin}}(x, x') &= \sigma_b + \sigma_y (x - c)^\top (x' - c),
\end{aligned} \tag{2.27}$$

where $(\ell, \sigma_y, p)$ are the hyperparameters of the periodic kernel and $(\sigma_b, \sigma_y, c)$ are hyperparameters of the linear kernel. For more general forms of these kernel functions in higher dimensional input spaces see [9, Section 45.4].

## 2.5 A Real World Example

Here, we mention an interesting real world example of GPs in action. The computer graphics and animation fields are filled with applications that require the setting of tricky parameters. In many cases, the models are complex and the parameters are unintuitive for non-experts. Brochu, Brochu, and de Freitas [10] present

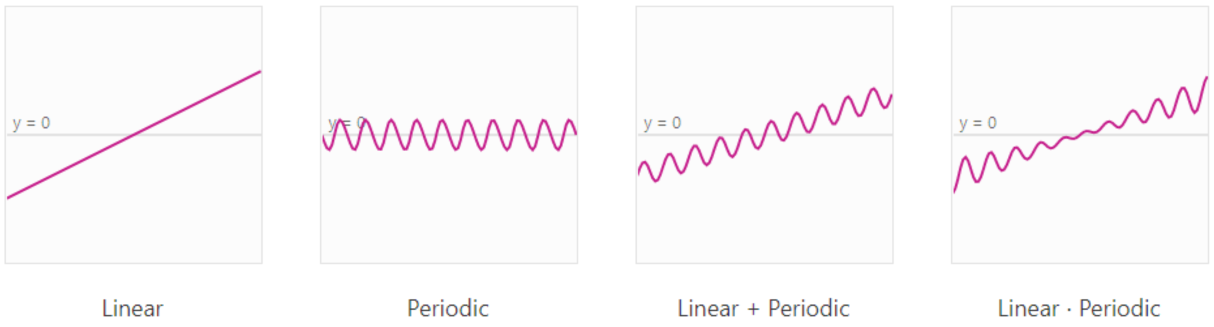

**Figure 5.** The figures show samples drawn from prior distributions with different kernels. Addition results in an oscillating function with a global trend, while multiplication increases the oscillation amplitude outwards [4].

an optimization method for setting parameters of a procedural fluid animation system by showing the user examples of different parametrized animations and asking for feedback. They simulate the fluid flow by taking the curl of a vector valued potential function which has several technical parameters. In this example, the the aforementioned parameters are the inputs $\mathbf{x}$ and the targets $\mathbf{y}$ are the feedbacks of the user and some previous default data. By introducing an **active learning** method, they employ the Bayesian technique of bringing in prior belief based on previous runs of the system and expert knowledge to assist users find good parameter settings in as few steps as possible. At the heart of the machinary they have produced, lies the GP introduced in this tutorial. The main benefit is that the GP helps construct an efficient **explore and exploit** strategy in the input space based on the distribution of uncertainty in that space [8, Lecture 9]. Watch the video of the **interactive interface** they have created to help users create animations without taking the burdon of adjusting parameters manually [11]. Indeed, this idea is notileresticted to computer graphics field and can be employed anywhere else that requires the adjusment of techincal parameters.

## 3 What is Next?

In this tutorial, we took a trip from the basics of GPs to their real world applications. We learned that GPs are **supervised probabilistic** models with high capability for doing regression. Fortunately, they can also be extended for classification tasks. For this purpose, the main idea is to pass the output values through a sigmoid function. You can study more about this in [3, Chapter 3]. More surprisingly, GPs have several models as their special cases. Indeed, some of the well known results are that

1. Using eq. (2.19) and Mercer's theorem [3, Section 4.3], it can be shown that the posterior mean $\mu_{1|2}$ is equivalent to a Regression model with nonlinear basis functions and **inifinite** number of parameters. In this regard, it is said that a GP model is a **non-parametric** model. Despite what the name suggests, it is meant that it can have as many parameters as required.
2. The Ridge regression model can be recoverd by the posterior mean $\mu_{1|2}$, if we choose a dot product kernel $\kappa(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$ and set the noise variance $\sigma_\epsilon$ equal to the regularization parameter $\delta$ [8, Lecture 9].
3. It can also be shown that specific large neural networks are equivalent to a GP [12].

The main drawback of GPs is their $\mathcal{O}(n^3)$ time complexity, which makes it hard to scale. One of the main trends of research in recent years is to make GPs appropriate for large datasets. You can start reading about this in [3, Chapter 8]. A good review about scalable GPs is also presented in [13].

## 4 Support

I am happy that you have read this so far, showing that you are serious about understanding GPs and machine learning. I tried to write an easy-to-follow and at the same time rigorous introduction to GPs and I hope that it appealed to you. If you liked this tutorial, please give the repository of this document a star on GitHub. Futhermore, you are more than welcomed to report any errors by opening an issue or to suggest further improvements by making pull requests on GitHub.

## 5 References

[1] S. Ghahramani, *Fundamentals of Probability with Stochastic Processes*, 4th. CRC Press, 2019.

[2] C. B. Do, "More on multivariate gaussians," *Lecture Notes*, pp. 1–11, 2009.

[3] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[4] J. Gortler, R. Kehlbeck, and O. Deussen. "A visual exploration of gaussian processes." (2019), [Online]. Available: https://distill.pub/2019/visual-exploration-gaussian-processes.

[5] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[6] P. E. Gill, W. Murray, and M. H. Wright, *Numerical Linear Algebra and Optimization*. Addison-Wesley, 1991, vol. 1.

[7] "Cross-validation: Evaluating estimator performance." (2019), [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html.

[8] N. de Freitas. "Machine learning." (2013), [Online]. Available: https://www.youtube.com/playlist?list=PLE6Wd9FR--EdyJ5lbFl8UuGjecvVw66F6.

[9] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 1998.

[10] E. Brochu, T. Brochu, and N. de Freitas, "A bayesian interactive optimization approach to procedural animation design," in *Proceedings of the 2010 ACM SIGGRAPH, Eurographics Symposium on Computer Animation*, Madrid, Spain: Eurographics Association, 2010, pp. 103–112.

[11] E. Brochu, T. Brochu, and N. de Freitas. "A bayesian interactive optimization approach to procedural animation design." (2010), [Online]. Available: https://www.youtube.com/watch?v=AUdW_Kx0u44.

[12] D. J. C. MacKay, "Introduction to gaussian processes," *Lecture Notes*, pp. 1–32, 1998.

[13] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When gaussian process meets big data: A review of scalable gps," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4405–4423, 2020.