



UNIVERSIDAD DE GRANADA

Facultad de Ciencias y Escuela Técnica Superior de Ingenierías
Informática y de Telecomunicación

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y
MATEMÁTICAS

TRABAJO DE FIN DE GRADO

Procesos Gaussianos para Aprendizaje Automático Supervisado

Presentado por:
Álvaro Luna Ramírez

Curso académico 2023-2024

Procesos Gaussianos para Aprendizaje Automático Supervisado

Álvaro Luna Ramírez

Álvaro Luna Ramírez *Procesos Gaussianos para Aprendizaje Automático Supervisado.*
Trabajo de fin de Grado. Curso académico 2023-2024.

**Responsable de
tutorización**

Pablo Morales Álvarez
*Departamento de Estadística e Investigación
Operativa*

José Luis Romero Béjar
*Departamento de Estadística e Investigación
Operativa*

Doble Grado en Ingeniería
Informática y Matemáticas

Facultad de Ciencias y
Escuela Técnica Superior
de Ingenierías Informática
y de Telecomunicación

Universidad de Granada

DECLARACIÓN DE ORIGINALIDAD

D./Dña. Álvaro Luna Ramírez

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2023-2024, es original, entendido esto en el sentido de que no he utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 15 de julio de 2024

Fdo: Álvaro Luna Ramírez

Agradecimientos

Quiero aprovechar este espacio para mostrar mi más sincero agradecimiento a todos aquellos que han sido fundamentales en este camino.

A la Universidad de Granada, por brindarme la oportunidad de formarme en un entorno académico de excelencia. A la Facultad de Ciencias, la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación y todos los profesores que durante estos cinco años han compartido su sabiduría y experiencia, contribuyendo de manera significativa a mi crecimiento académico y personal.

A mis tutores, Pablo y José Luis, cuya orientación y conocimiento han sido imprescindibles durante la realización de este trabajo. En especial a Pablo, por su paciencia y apoyo en tantas tutorías. Gracias por compartir sus conocimientos y por ayudarme a desarrollar mi capacidad de investigación.

A mis compañeros, Germán, Isa, Nerea y Pedro, por su amistad, por los momentos compartidos y por ser una fuente constante de motivación. Gracias por estar siempre ahí, en las buenas y en las malas.

A mi familia, por su amor y apoyo inquebrantable. A mis padres, por creer en mí y darme las herramientas necesarias para seguir adelante. A mis hermanos, por su constante ánimo y comprensión. Sin su respaldo, esto no habría sido posible.

Finalmente, quiero agradecer a todos aquellos que, de una manera u otra, han contribuido a que este Trabajo de Fin de Grado sea una realidad. A todos ustedes, gracias.

Procesos Gaussianos para Aprendizaje Automático Supervisado

Álvaro Luna Ramírez

Palabras clave:

Aprendizaje Automático Supervisado, Inferencia Bayesiana, Procesos Gaussianos, Problemas de Regresión, Problemas de Clasificación, Aproximación de Laplace, Función de Covarianza, Hiperparámetros

Resumen

Este proyecto estudia en profundidad la formulación teórica de los Procesos Gaussianos e investiga su aplicación en tareas de aprendizaje supervisado, resaltando tanto su eficacia como sus limitaciones. Los Procesos Gaussianos son herramientas que adoptan un enfoque probabilístico, permitiendo el trato de la incertidumbre en las predicciones mediante la inferencia Bayesiana facilitada por el Teorema de Bayes. Sin embargo, un inconveniente significativo es su complejidad computacional, que escala de forma cúbica con el número de muestras de entrenamiento.

En este trabajo, estudiamos modelos avanzados basados en Procesos Gaussianos, enfocándonos en problemas de regresión y clasificación. La tarea de clasificación presenta desafíos debido a la naturaleza no gaussiana de la función de verosimilitud, lo que requiere la adopción de la aproximación de Laplace, que aproxima la posteriori como una distribución normal. Además, investigamos funciones de covarianza esenciales como kernels estacionarios, isotrópicos y basados en el producto escalar, junto con la optimización de hiperparámetros.

La investigación incluye una comparación de Procesos Gaussianos con otros modelos populares en varios conjuntos de datos. Los resultados muestran que, aunque los Procesos Gaussianos logran una alta precisión y permiten una estimación de la incertidumbre, su costo computacional es mayor para aplicaciones con grandes conjuntos de datos, por lo que son menos prácticos en tales escenarios.

Este trabajo amplía conceptos estudiados durante la carrera. Se estudia en detalle un modelo probabilístico que no se ha abordado en la asignatura de 'Aprendizaje Automático' y se da una nueva perspectiva sobre el manejo de la incertidumbre en esta área. Se profundiza en la 'Inferencia Estadística' con un enfoque bayesiano, explorando cómo se pueden actualizar las probabilidades utilizando el Teorema de Bayes a medida que se dispone de nueva información. Además, se amplían los conocimientos de 'Estadística Multivariante' mediante el uso de Procesos Gaussianos, que generalizan los conceptos de la distribución normal a un número infinito de variables aleatorias.

Gaussian Processes for Supervised Machine Learning

Álvaro Luna Ramírez

Keywords:

Supervised Learning, Bayesian Inference, Gaussian Processes, Regression Problems, Classification Problems, Laplace Approximation, Covariance Function, Hyperparameters

Abstract

This project studies the theoretical formulation of Gaussian Processes and investigates their application in supervised learning tasks, highlighting their effectiveness and limitations. Gaussian Processes are tools that adopt a probabilistic approach, allowing for the quantification of uncertainty in predictions through Bayesian inference facilitated by Bayes' Theorem. However, a significant inconvenient is their computational complexity, which scales cubically with the number of training samples.

In this work, we study advanced models based on Gaussian Processes, focusing on regression and classification problems. The classification task presents challenges due to the non-Gaussian nature of the likelihood function, requiring the adoption of the Laplace approximation, which approximates the posterior distribution as Gaussian. Additionally, we investigate essential covariance functions such as stationary, isotropic, and dot product kernels, along with the optimization of hyperparameters.

The research includes a comparison of Gaussian Processes with other popular models across several datasets. Results demonstrate that while Gaussian Processes achieve high accuracy and allow an estimate of uncertainty, their computational cost is higher for large-scale applications, making them less practical for such scenarios.

This work expands concepts studied during the degree. A detailed study is conducted on a probabilistic model that was not covered in the 'Machine Learning' course, providing a new perspective on handling uncertainty in this area. The work delves into 'Statistical Inference' with a Bayesian approach, exploring how probabilities can be updated using Bayes' Theorem as new information becomes available. Additionally, knowledge of 'Multivariate Statistics' is expanded through the use of Gaussian Processes, which generalize the concepts of the normal distribution to an infinite number of random variables.

Índice general

1. Introducción	1
2. Fundamentos Teóricos	5
2.1. Distribución Normal	5
2.1.1. Distribución Normal Multivariante	6
2.2. Probabilidad Bayesiana	11
2.3. Aprendizaje Automático Supervisado	13
2.4. Procesos Gaussianos	18
2.4.1. Usos y contexto histórico	20
3. Procesos Gaussianos en Problemas de Regresión	23
3.1. Espacio de Parámetros: Modelo Lineal Bayesiano	23
3.1.1. Proyección al Espacio de Características	26
3.2. Espacio de funciones	28
3.2.1. Observaciones sin Ruido	29
3.2.2. Observaciones con Ruido	30
3.3. Hiperparámetros	33
3.4. Teoría de Decisión	34
4. Procesos Gaussianos en Problemas de Clasificación	37
4.1. Clasificadores Probabilísticos	37
4.1.1. Enfoque Generativo	39
4.1.2. Enfoque Discriminativo	40
4.2. Modelos Lineales de Clasificación	41
4.3. Uso de Procesos Gaussianos	45
4.4. Aproximación de Laplace	46
5. Función de Covarianza y Estimación de Hiperparámetros	51
5.1. Función de Covarianza	51
5.1.1. Funciones de Covarianza Estacionarias	52
5.1.2. Funciones de Covarianza No Estacionarias	57
5.1.3. Generación de Nuevos Kernels a partir de Otros Existentes	58
5.2. Estimación de Hiperparámetros	59
6. Experimentos	63
6.1. Modelado Determinístico vs Modelado Probabilístico	63
6.2. Clasificación	66
6.2.1. <i>Optical Recognition of Handwritten Digits</i>	66
6.2.2. Iris	72
6.3. Regresión	81
6.3.1. Mediciones de CO ₂ en el Volcán Mauna Loa	82

Índice general

7. Conclusiones y Trabajos Futuros	91
7.1. Objetivos Alcanzados	91
7.2. Futuros Trabajos	92
A. Apéndices	93
A.1. Factorización de Cholesky	93
A.2. Método de Newton-Raphson	93
A.3. Planificación Temporal	94
Bibliografía	97

1. Introducción

En las últimas décadas, el Aprendizaje Automático [AMMIL12] [BNo6] ha experimentado un gran auge en numerosos ámbitos, impulsado por varios factores clave como la alta disponibilidad de grandes cantidades de datos y los avances en tecnología y hardware que han facilitado un procesamiento más rápido de la información. Gracias a esto, los algoritmos han adquirido la capacidad de aprender patrones a partir de datos de una manera que hace años era inimaginable, y numerosos campos de conocimiento, desde la visión por computador hasta la predicción en ciencias ambientales, han presenciado una gran revolución. Hoy en día, en numerosas situaciones cotidianas hacemos uso de Aprendizaje Automático. Por ejemplo, plataformas como *Youtube* o *Netflix* lo usan para comprender el comportamiento de los usuarios y dar prioridad a los vídeos o series que los usuarios tienen más probabilidades de ver y con los que es más probable que interactúen.

En concreto, este trabajo se centra en el paradigma del Aprendizaje Automático Supervisado [CMo7], que consiste en entrenar modelos utilizando datos etiquetados. Muchos de estos modelos, como pueden ser las redes neuronales, a pesar de obtener muy buen rendimiento en problemas de alta complejidad, presentan desafíos como la interpretabilidad de los resultados y la necesidad de grandes volúmenes de datos para evitar fenómenos no deseados como el sobreajuste.

En este contexto, el presente trabajo pretende estudiar el uso de Procesos Gaussianos como alternativa por su capacidad para gestionar la incertidumbre de manera eficiente y proporcionar predicciones precisas, incluso con cantidades limitadas de datos. Para ello, el trabajo presenta la siguiente estructura:

Tras este primer capítulo introductorio, en el Capítulo 2 se tratarán los fundamentos teóricos para el desarrollo del mismo. En esta segunda sección también se presentarán los Procesos Gaussianos, que se definen como un conjunto de variables aleatorias que cumple que cualquiera de sus subconjuntos finitos tiene una Distribución Normal conjunta [PDPF01]. En otras palabras, un Proceso Gaussiano es un tipo específico de proceso estocástico [Rin12] que se puede entender como una generalización de la Distribución Normal Multivariante [TT90]. Este tipo de proceso permite un número infinito de variables aleatorias, ofreciendo una manera no paramétrica y probabilística de aprender a partir de datos. Tienen un enfoque bayesiano que les permite una interpretación natural de la incertidumbre en las predicciones, algo crucial en aplicaciones donde las estimaciones confiables son esenciales, como podría ser en medicina.

Este enfoque bayesiano [Ber18] es una interpretación de la probabilidad que incorpora por un lado información previa y por otro lado una nueva evidencia para actualizar la probabilidad de un evento. Debe su nombre al Teorema de Bayes, que combina la probabilidad a priori de una hipótesis con la verosimilitud de los datos observados, para calcular la probabilidad

1. Introducción

a posteriori, como se puede ver en la siguiente expresión:

$$\text{posteriori} = \frac{\text{verosimilitud} \times \text{priori}}{\text{verosimilitud marginal}}.$$

En el marco del Aprendizaje Automático Supervisado estudiaremos el uso de Procesos Gaussianos en dos tipos de problemas:

Por una parte, durante el Capítulo 3 se presentarán los problemas de regresión, donde la tarea es predecir valores continuos y se busca encontrar una función que relacione las variables de entrada con la variable de salida continua, minimizando el error de predicción. Véase como ejemplo la Figura 1.1.

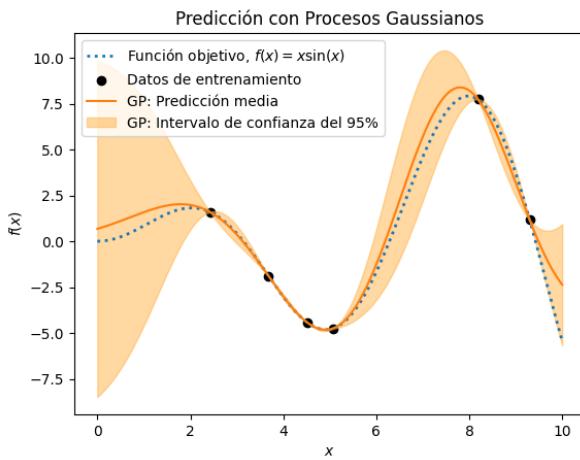


Figura 1.1.: Ejemplo de problema de regresión, que se verá en el apartado 6.1. A partir de los 6 datos de entrenamiento se busca aproximar la función objetivo $f(x) = xsin(x)$. Se da con una línea naranja la predicción del modelo de Procesos Gaussianos, mientras que se muestra un intervalo de incertidumbre del 95 % con una sombra anaranjada. Código de programación propio, disponible en este [enlace](#).

Por otro lado, se estudiarán los problemas de clasificación en el Capítulo 4, donde el objetivo es asignar una etiqueta discreta a una observación basándose en sus atributos de entrada. A diferencia de la regresión, donde se predicen valores continuos, en la clasificación se predicen categorías discretas, y el modelo busca encontrar una frontera de decisión que separe las diferentes clases. Para poder clasificar correctamente nuevas observaciones que no fueron vistas durante el entrenamiento el modelo se basará en estas fronteras. Véase el ejemplo de la Figura 1.2.

A lo largo de esta sección nos encontraremos algunas dificultades que cabe destacar, ya que en los problemas de clasificación, a diferencia de la regresión, no se podrán realizar todos los cálculos de forma cerrada y directa. Nos encontraremos inconvenientes en algunas integrales, debido a que la verosimilitud no será gaussiana, y por lo tanto en clasificación no podremos aplicar los Procesos Gaussianos directamente y se tendrán que realizar algunas aproximaciones. En este trabajo se utilizará el método de la Aproximación de Laplace, facili-

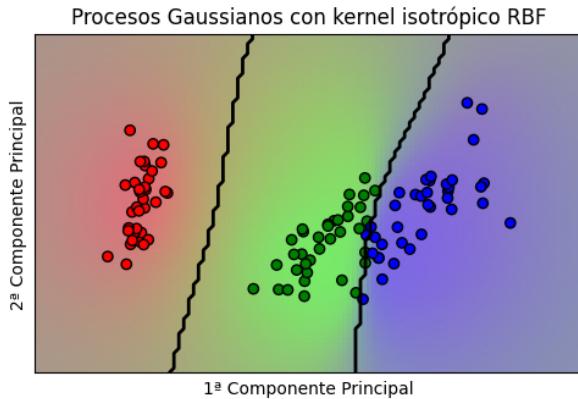


Figura 1.2.: Ejemplo de problema de clasificación que se abordará en el apartado 6.2.2. Se muestran los datos de entrenamiento de tres clases diferentes, correspondientes a tres especies de la flor iris, marcadas en rojo, verde y azul. Las líneas negras muestran las fronteras de decisión de cada clase. Código de programación propio, disponible en este [enlace](#).

tando así la inferencia y el cálculo de las predicciones.

Adicionalmente, se estudiará cómo los Procesos Gaussianos pueden adaptarse a diversas complejidades de datos mediante la selección adecuada de funciones de covarianza, ofreciendo una flexibilidad considerable sin perder interpretabilidad. En el Capítulo 5 se estudiarán las principales funciones de covarianza y sus características, así como los métodos más comunes para determinar sus hiperparámetros.

En el ámbito de los Procesos Gaussianos en Aprendizaje Automático Supervisado la fuente de información más importante es el libro *Gaussian Processes for Machine Learning* de Edward Rasmussen y Christopher Williams [RW+06]. Para no citar el libro constantemente, a menos que se indique explícitamente lo contrario, se parte del entendimiento de que la teoría de este TFG se obtiene de dicho libro.

En los experimentos realizados durante el Capítulo 6 se emplearán las habilidades de programación desarrolladas a lo largo de los cinco años de Ingeniería Informática. En esta ocasión en el lenguaje Python, que a pesar de no verse con excesiva profundidad en la carrera se ha convertido en una herramienta esencial tanto en el mundo empresarial como de la investigación. Se pondrá en práctica la teoría desarrollada durante el trabajo en 2 ejemplos de problemas de regresión y 2 ejemplos de problemas de clasificación, en los que se utilizarán las clases '*GaussianProcessRegressor*' y '*GaussianProcessClassifier*' de la biblioteca *Scikit-Learn*. Todo el código de programación de estos experimentos, al igual que el de todas las imágenes insertadas a lo largo del trabajo, se encuentran en el siguiente [repositorio de Github](#). Para el desarrollo de estas pruebas también se usarán los siguientes Datasets:

- *Optical Recognition of Handwritten Digits*, del *UC Irvine Machine Learning Repository*.
- Conjunto de datos de flores Iris
- Mediciones de CO₂ en el Observatorio del volcán Mauna Loa (Hawái), de la Oficina

1. Introducción

Nacional de Administración Oceánica y Atmosférica de los Estados Unidos.

Finalmente, en el Capítulo 7, se extraerán conclusiones sobre los conocimientos adquiridos y los resultados obtenidos en las pruebas realizadas. Además, se discutirán posibles trabajos futuros.

En este trabajo se demuestra la estrecha relación entre dos disciplinas interconectadas: la Ingeniería Informática y las Matemáticas. Al combinar estas áreas del conocimiento, se pueden obtener resultados notablemente interesantes y valiosos, permitiendo desarrollar soluciones innovadoras y eficientes para una amplia gama de problemas. Se pueden aprovechar las fortalezas de ambas disciplinas para abordar desafíos complejos de manera más efectiva.

2. Fundamentos Teóricos

En este primer capítulo del trabajo se establecen los fundamentos necesarios para el desarrollo del mismo. Se comenzará estudiando la Distribución Normal en la Sección 2.1, tanto en su formulación unidimensional como en sus extensiones multivariante. En la Sección 2.2, nos adentraremos en la Probabilidad Bayesiana, una teoría que será fundamental y sustenta la aplicación de los Procesos Gaussianos en el Aprendizaje Automático. Luego, en la Sección 2.3, se introducirá el Aprendizaje Automático Supervisado, descubriendo sus principios y usos. Finalmente, en la Sección 2.4, nos adentraremos en los Procesos Gaussianos, destacando algunas de sus aplicaciones relevantes.

2.1. Distribución Normal

En este apartado se introducirá la Distribución Normal y se tratarán algunas de sus principales características. Comúnmente es también denominada como 'Gaussiana', aunque la distribución fue reconocida por primera vez por el matemático francés Abraham **de Moivre** (1667-1754) durante el estudio de la aproximación de la distribución binomial para grandes valores de n . Años más tarde, Carl Friedrich **Gauss** (1777-1855) elaboró desarrollos más profundos y formuló la ecuación de la curva, que por esta razón es comúnmente conocida como la '**campana de Gauss**' [PDPFo1].

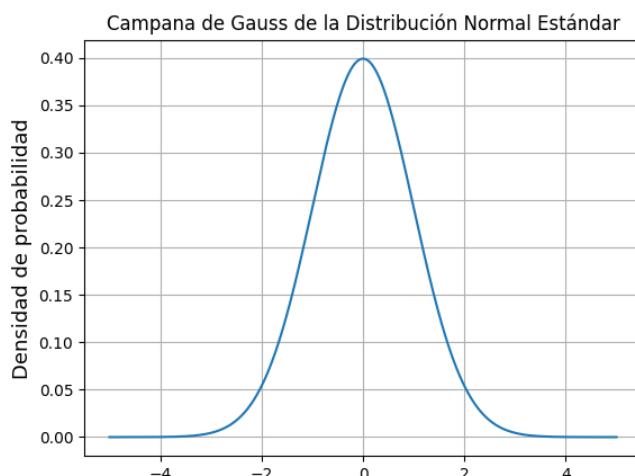


Figura 2.1.: Función de densidad de la Distribución Normal de media 0 y desviación típica 1. Código de programación propio, disponible en este [enlace](#).

Esta distribución, que es uno de los conceptos más importantes de la estadística, aparentemente recibió el nombre de 'normal' en referencia a las ecuaciones normales que aparecían en algunas de las aplicaciones que le daba Gauss, donde en realidad por normal se refería

2. Fundamentos Teóricos

a 'ortogonal', no a 'habitual u ordinario'. Karl **Pearson**, a finales del siglo XIX, terminó de popularizar el nombre de Distribución Normal:

'Many years ago I called the Laplace–Gaussian curve the normal curve, which name, while it avoids an international question of priority, has the disadvantage of leading people to believe that all other distributions of frequency are in one sense or another 'abnormal'. ' Pearson, 1920 [Pea20]

Definición 2.1. Sean $\mu \in \mathbb{R}$, $\sigma^2 \in \mathbb{R}_+$. Se dice que una variable aleatoria continua X sigue una **Distribución Normal** de parámetros μ y σ^2 si su función de densidad f_X viene dada por la siguiente expresión:

$$f_X(x) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \text{para } -\infty < x < \infty.$$

Se notará como $X \sim \mathcal{N}(\mu, \sigma^2)$.

A continuación, se presentan algunas de las principales características de la Distribución Normal:

- Es simétrica respecto a su media, lo que implica que la probabilidad de encontrar un valor por encima de la media es igual a la probabilidad de encontrar un valor por debajo de la media. Véase la Figura 2.1.
- La media coincide con la mediana y la moda.
- Es cerrada bajo transformaciones lineales.
- La media y la desviación típica determinan completamente la Distribución Normal.
- El Teorema Central del Límite establece que cuando se dispone de una muestra aleatoria de un tamaño lo suficientemente grande, aunque esta siga una distribución no normal (incluso distribuciones típicas de variables aleatorias discretas), la distribución de las medias muestrales seguirá una distribución normal [MGMB10].

2.1.1. Distribución Normal Multivariante

La **Distribución Normal Multivariante (DNM)** constituye la extensión natural, en el contexto multivariante, de la Distribución Normal Univariante. En esta sección, se introducirán los conceptos fundamentales de la Distribución Normal Multivariante [TT90].

Definición 2.2. Un **vector aleatorio** es un vector $X = (X_1, \dots, X_p)'$ cuyas componentes, X_i para $i = 1, \dots, p$, son variables aleatorias definidas sobre el mismo espacio de probabilidad (Ω, A, P) .

Proposición 2.1. Dado un vector aleatorio $X \sim (\mu, \Sigma)$ de dimensión p con $\Sigma > 0$, para cualquier matriz C de dimensión $p \times p$ tal que $\Sigma = CC'$, se dice que el vector $Z = C^{-1}(X - \mu)$ es una **normalización** de X y se verifica que $Z \sim (0_p, I_{p \times p})$.

Demostración. Probaremos que $\mathbb{E}[Z] = 0_p$ y $\text{Cov}(Z) = I_{p \times p}$. Por un lado se tiene que:

$$\mathbb{E}[Z] = \mathbb{E}[C^{-1}(X - \mu)] = C^{-1}\mathbb{E}[X - \mu] = C^{-1}(\mathbb{E}[X] - \mathbb{E}[\mu]) = C^{-1}(\mu - \mu) = 0.$$

Por otro lado, sabiendo que $\mathbb{E}[Z] = 0$ y que $Z' = [C^{-1}(X - \mu)]' = (X - \mu)'(C^{-1})'$:

$$\begin{aligned} \text{Cov}[Z] &= \mathbb{E}[ZZ'] = \mathbb{E}[C^{-1}(X - \mu)(X - \mu)'C^{-1}'] = C^{-1}\mathbb{E}[(X - \mu)(X - \mu)'](C^{-1})' = \\ &= C^{-1}\Sigma(C^{-1})' = C^{-1}CC'(C^{-1})' = C^{-1}C(CC^{-1})' = I_{p \times p}. \end{aligned}$$

□

Definición 2.3. Sea un vector aleatorio $X = (X_1, \dots, X_p)'$, se define su **función característica**, ϕ_X , como:

$$\phi_X(t) = \mathbb{E}[e^{it'X}] = \int_{\Omega} e^{it'X(w)} P(dw) = \int_{\mathbb{R}^p} e^{it'x} P_X(dx) \quad \forall t = (t_1, \dots, t_p) \in \mathbb{R}^p.$$

Teorema 2.1. La función característica de un vector aleatorio X de dimensión p siempre existe para cualquier $t = (t_1, \dots, t_p) \in \mathbb{R}^p$. Además, la función característica determina de forma única la distribución de X .

Definición 2.4. Sea $X \sim (\mu, \Sigma)$ con $\Sigma > 0$. Se define la **Distancia de Mahalanobis** de X respecto a la media μ como:

$$\Delta(X, \mu) := \{(X - \mu)' \Sigma^{-1} (X - \mu)\}^{\frac{1}{2}}.$$

Observación 2.1. De la definición de la Distancia de Mahalanobis se obtiene:

- Se cumple que $\Delta(X, \mu) = \|Z\|_p$, siendo Z una normalización cualquiera de X y $\|\cdot\|_p$ la norma Euclídea en \mathbb{R}^p .
- $\Delta(X, \mu)$ es una variable aleatoria que verifica que $E[\Delta^2(X, \mu)] = p$
- Sea $x \in \mathbb{R}^p$ y $k > 0$ constante. La ecuación $\Delta(x, \mu) = k$ define un hiperelipsoide en \mathbb{R}^p , de tal forma que los puntos transformados por normalización corresponden a la esfera euclídea de dimensión p de radio k y centro en el origen.

Definición 2.5. Sea el vector aleatorio $X = (X_1, \dots, X_p)'$. Se dice que X sigue una **Distribución Normal p-variante** si su función de densidad f_X viene dada por la siguiente expresión:

$$f_X(x) = \frac{1}{(2\pi)^{\frac{p}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(x - \mu)' \Sigma^{-1} (x - \mu)\right\} \quad \forall x \in \mathbb{R}^p, \quad (2.1)$$

con $\mu = (\mu_1, \dots, \mu_p)' \in \mathbb{R}^p$ y siendo Σ una matriz simétrica de escalares, definida positiva y de dimensión $(p \times p)$. Se nota como $X \sim \mathcal{N}_p(\mu, \Sigma)$, siendo μ y Σ los parámetros de la distribución. En concreto, son el vector de medias y la matriz de covarianzas de X .

La DNM está completamente determinada por sus momentos de primer y segundo orden, es decir, su **media** (μ) y **covarianza** (Σ). Estos parámetros proporcionan información esencial sobre la distribución, permitiendo caracterizar tanto su ubicación central como su dispersión y relación entre variables.

$$\mu_X := \mathbb{E}[X] := \begin{pmatrix} \mathbb{E}[X_1] \\ \vdots \\ \mathbb{E}[X_2] \end{pmatrix} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_2 \end{pmatrix}$$

2. Fundamentos Teóricos

$$\Sigma_X = Cov(X) := \mathbb{E}[(X - \mu_X)(X - \mu_X)'] = \begin{pmatrix} \sigma_{11} & \dots & \sigma_{1p} \\ \vdots & \ddots & \vdots \\ \sigma_{p1} & \dots & \sigma_{pp} \end{pmatrix},$$

con $\sigma_{ij} = \mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)] = \sigma_{ji}$. En particular, $\sigma_{ii} = \mathbb{E}[(X_i - \mu_i)^2] = Var(X_i) = \sigma_i^2$.

Se verifica que Σ_X es simétrica, los elementos de la diagonal son $\sigma_{ii} \geq 0 \ \forall i \in \{1, \dots, p\}$ y la clase de las matrices de covarianza de dimensión $p \times p$ coincide con la clase de matrices semidefinidas positivas de dimensión $p \times p$.

Otra característica relevante es que la DNM es cerrada bajo **transformaciones lineales**. Esto implica que si aplicamos una transformación lineal a un vector aleatorio con DNM, el resultado seguirá siendo un vector aleatorio con DNM.

Proposición 2.2. Sean $X \sim \mathcal{N}_p(\mu_X, \Sigma_X)$ con $\Sigma > 0$ y $Y = BX + b$ con B matriz constante y regular de dimensión $(p \times p)$ y b vector de constantes de dimensión $(p \times 1)$. Entonces, se tiene que $Y \sim \mathcal{N}_p(\mu_Y, \Sigma_Y) = \mathcal{N}_p(B\mu_X + b, B\Sigma_X B')$.

Demostración. Como $Y = BX + b \Rightarrow X = B^{-1}(Y - b)$.

$$\begin{aligned} f_Y(y) &= f_X(B^{-1}(Y - b)) \text{abs}(|B^{-1}|) = \\ &= \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(B^{-1}(y - b) - \mu_X)' \Sigma_X^{-1} (B^{-1}(y - b) - \mu_X)\right\} \text{abs}(|B^{-1}|) = \\ &= \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}[B^{-1}B(B^{-1}(y - b) - \mu_X)]' \Sigma_X^{-1} [B^{-1}B(B^{-1}(y - b) - \mu_X)]\right\} \text{abs}(|B^{-1}|) = \\ &= \frac{1}{(2\pi)^{p/2} \text{abs}(|B|)^{1/2} |\Sigma|^{1/2} \text{abs}(|B'|)^{1/2}} \exp\left\{-\frac{1}{2}(Y - B\mu_X - b)' \Sigma_X^{-1} (Y - B\mu_X - b)\right\} = \\ &= \frac{1}{(2\pi)^{p/2} |B\Sigma_X B'|^{1/2}} \exp\left\{-\frac{1}{2}(Y - B\mu_X - b)' \Sigma_X^{-1} (Y - B\mu_X - b)\right\} \\ &\Rightarrow Y \sim \mathcal{N}_p(B\mu_X + b, B\Sigma_X B') = \mathcal{N}_p(\mu_Y, \Sigma_Y) \quad \square \end{aligned}$$

Observación 2.2. Este resultado se generaliza para los casos $\Sigma \geq 0$ y B matriz constante de dimensión $(q \times p)$. En esta segunda situación, se mantiene $Y \sim \mathcal{N}_p(\mu_Y, \Sigma_Y) = \mathcal{N}_p(B\mu_X + b, B\Sigma_X B')$, con la circunstancia de que $\text{rango}(B\Sigma_X B') \leq \min\{\text{rango}(B), \text{rango}(\Sigma)\}$.

La DNM también es cerrada bajo **combinaciones lineales de vectores independientes**. Esta propiedad es útil en la modelización y análisis de sistemas multivariantes, ya que permite generar nuevas variables aleatorias con distribución normal multivariante a partir de combinaciones lineales de variables independientes.

Proposición 2.3. Sean respectivamente $X_k, k = 1, \dots, m$ vectores aleatorios p -dimensionales con MND, $X_k \sim \mathcal{N}_p(\mu_k, \Sigma_k)$, siendo independientes. Entonces, para cada conjunto de matrices constantes $A_k, k = 1, \dots, m$ de dimensión $q \times p$, se tiene que

$$Y := \sum_{k=1}^m A_k X_k \sim \mathcal{N}_q\left(\sum_{k=1}^m A_k \mu_k, \sum_{k=1}^m A_k \Sigma_k A_k'\right).$$

Demostración.

$$\begin{aligned}
 \phi_Y(t) &= \mathbb{E}[\exp\{it'Y\}] = \mathbb{E}[\exp\{it'\sum_{k=1}^m A_k X_k\}] = \mathbb{E}[\exp\{\sum_{k=1}^m i(A'_k t)'X_k\}] = \\
 &= \prod_{k=1}^m \mathbb{E}[\exp\{i(A'_k t)'X_k\}] = \prod_{k=1}^m \phi_{X_k}(A'_k t) = \prod_{k=1}^m \exp\{i(A'_k t)'\mu_k - \frac{1}{2}(A'_k t)'\Sigma_k(A'_k t)\} = \\
 &= \exp\{it'\sum_{k=1}^m A_k \mu_k - \frac{1}{2}t'\sum_{k=1}^m (A_k \Sigma_k A'_k)t\} \\
 \Rightarrow Y &\sim \mathcal{N}_q(\sum_{k=1}^m A_k \mu_k, \sum_{k=1}^m A_k \Sigma_k A'_k).
 \end{aligned}$$

□

Una de las propiedades más importantes de la DNM es que las **distribuciones marginales** sean también normales. Esto significa que si tomamos subconjuntos de variables de una Distribución Normal Multivariante, las nuevas distribuciones siguen siendo normales.

Proposición 2.4. *Sea el vector aleatorio $X = (X_1, \dots, X_p)'$ con la siguiente DNM:*

$$X \sim \mathcal{N}_p(\mu_X, \Sigma_X).$$

Asumiendo que las componentes de X están ordenadas de la siguiente forma:

$$\mathbf{X} = \begin{pmatrix} X_{(1)} \\ X_{(2)} \end{pmatrix}, \text{ con } X_{(1)} = (X_1, \dots, X_q)', (X_{(2)} = (X_{q+1}, \dots, X_p)',$$

y considerando $\mu = \begin{pmatrix} \mu_{(1)} \\ \mu_{(2)} \end{pmatrix}$ y $\Sigma = \begin{pmatrix} \Sigma_{(11)} & \Sigma_{(12)} \\ \Sigma_{(21)} & \Sigma_{(22)} \end{pmatrix}$, entonces $\forall q < p$ se cumple que las distribuciones marginales de $X_{(1)}$ y $X_{(2)}$ son: $X_{(1)} \sim \mathcal{N}_q(\mu_{(1)}, \Sigma_{(11)})$ y $X_{(2)} \sim \mathcal{N}_{n-q}(\mu_{(2)}, \Sigma_{(22)})$.

En la siguiente proposición se demuestra que si dos subvectores de un vector aleatorio con DNM tienen **correlación cruzada nula**, entonces son **independientes**. Esto significa que la ausencia de correlación implica independencia en el contexto de la DNM.

Proposición 2.5. *Sea el vector aleatorio $X = (X_1, \dots, X_p)'$ con la siguiente DNM:*

$$X \sim \mathcal{N}_p(\mu_X, \Sigma_X), \Sigma > 0.$$

Asumiendo que las componentes de X están ordenadas de la siguiente forma:

$$\mathbf{X} = \begin{pmatrix} X_{(1)} \\ X_{(2)} \end{pmatrix}, \text{ con } X_{(1)} = (X_1, \dots, X_q)', (X_{(2)} = (X_{q+1}, \dots, X_p)',$$

y considerando $\mu = \begin{pmatrix} \mu_{(1)} \\ \mu_{(2)} \end{pmatrix}$ y $\Sigma = \begin{pmatrix} \Sigma_{(11)} & \Sigma_{(12)} \\ \Sigma_{(21)} & \Sigma_{(22)} \end{pmatrix} = \begin{pmatrix} \Sigma_{(11)} & 0 \\ 0 & \Sigma_{(22)} \end{pmatrix}$. Entonces, los vectores aleatorios $X_{(1)}$ y $X_{(2)}$ son independientes y tienen las siguientes DNM:

$$X_{(1)} \sim \mathcal{N}_q(\mu_{(1)}, \Sigma_{(11)})$$

$$X_{(2)} \sim \mathcal{N}_{p-q}(\mu_{(2)}, \Sigma_{(22)})$$

2. Fundamentos Teóricos

Demostración. Como

$$\begin{aligned} & \begin{pmatrix} X_{(1)} - \mu_{(1)} \\ X_{(2)} - \mu_{(2)} \end{pmatrix}^t \begin{pmatrix} \Sigma_{(11)} & 0 \\ 0 & \Sigma_{(22)} \end{pmatrix}^{-1} \begin{pmatrix} X_{(1)} - \mu_{(1)} \\ X_{(2)} - \mu_{(2)} \end{pmatrix} = \\ &= \begin{pmatrix} X_{(1)} - \mu_{(1)} \\ X_{(2)} - \mu_{(2)} \end{pmatrix}^t \begin{pmatrix} \Sigma_{(11)}^{-1} (X_{(1)} - \mu_{(1)}) \\ \Sigma_{(22)}^{-1} (X_{(2)} - \mu_{(2)}) \end{pmatrix}^t = \\ &= (X_{(1)} - \mu_{(1)})^t \Sigma_{(11)}^{-1} (X_{(1)} - \mu_{(1)}) + (X_{(2)} - \mu_{(2)})^t \Sigma_{(22)}^{-1} (X_{(2)} - \mu_{(2)}) \end{aligned}$$

Partiendo de la función de densidad de X ,

$$\begin{aligned} f_X(x) &= \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x - \mu)' \Sigma_X^{-1} (x - \mu)\right\} = \\ &= \frac{1}{(2\pi)^{q/2} |\Sigma_{(11)}|^{1/2}} \exp\left\{-\frac{1}{2}(x_{(1)} - \mu_{(1)})' \Sigma_{(11)}^{-1} (x_{(1)} - \mu_{(1)})\right\} \times \\ &\quad \times \frac{1}{(2\pi)^{(p-q)/2} |\Sigma_{(22)}|^{1/2}} \exp\left\{-\frac{1}{2}(x_{(2)} - \mu_{(2)})' \Sigma_{(22)}^{-1} (x_{(2)} - \mu_{(2)})\right\} \end{aligned}$$

$\Rightarrow X_{(1)}$ y $X_{(2)}$ son independientes y siguen Distribuciones Normales Multivariantes:

$$X_{(1)} \sim \mathcal{N}_q(\mu_{(1)}, \Sigma_{(11)})$$

$$X_{(2)} \sim \mathcal{N}_{p-q}(\mu_{(2)}, \Sigma_{(22)})$$

□

También puede demostrarse que las **distribuciones condicionadas** de cualquier dimensión son normales:

Proposición 2.6. *Sea el vector aleatorio $X = (X_1, \dots, X_p)'$ con la siguiente DNM:*

$$X \sim \mathcal{N}_p(\mu_X, \Sigma_X), \Sigma > 0.$$

Asumiendo que las componentes de X están ordenadas de la siguiente forma:

$$\mathbf{X} = \begin{pmatrix} X_{(1)} \\ X_{(2)} \end{pmatrix}, \text{ con } X_{(1)} = (X_1, \dots, X_q)', (X_{(2)} = (X_{q+1}, \dots, X_p)',$$

y considerando $\mu = \begin{pmatrix} \mu_{(1)} \\ \mu_{(2)} \end{pmatrix}$ y $\Sigma = \begin{pmatrix} \Sigma_{(11)} & \Sigma_{(12)} \\ \Sigma_{(21)} & \Sigma_{(22)} \end{pmatrix}$. Entonces, la distribución condicionada de $X_{(2)}$ dado $X_{(1)} = x_{(1)}$ es una DNM de la forma

$$\mathcal{N}_{p-q}(\mu_{(2)} + \Sigma_{(21)} \Sigma_{(11)}^{-1} (x_{(1)} - \mu_{(1)}), \Sigma_{(22)} - \Sigma_{(21)} \Sigma_{(11)}^{-1} \Sigma_{(12)}).$$

Observación 2.3. En particular, las distribuciones condicionadas de dimensión uno son normales unidimensionales que tienen por media el valor esperado por la regresión lineal, y por varianza la varianza residual de esa regresión.

Por último, también se tiene que el producto de dos MND es también normal [RW^{+o6}].

Proposición 2.7. Sean $\mathcal{N}(x | a, A)$ y $\mathcal{N}(x | b, B)$ dos distribuciones normales p -variantes, con medias a , b y matrices de covarianza A y B . Entonces, el producto será la siguiente distribución normal:

$$\mathcal{N}(x | a, A)\mathcal{N}(x | b, B) = Z^{-1}\mathcal{N}(x | c, C),$$

donde

$$\begin{aligned} C &= (A^{-1} + B^{-1})^{-1} \\ c &= C(A^{-1}a + B^{-1}b) \\ Z^{-1} &= \frac{1}{(2\pi)^{-p/2}} |A + B|^{-1/2} \exp\left(\frac{-1}{2}(a - b)^T (A + B)^{-1}(a - b)\right). \end{aligned}$$

2.2. Probabilidad Bayesiana

La **Probabilidad Bayesiana**, llamada así en honor al matemático **Thomas Bayes**, es una de las interpretaciones fundamentales del concepto de probabilidad. Se basa en un razonamiento lógico que actualiza la probabilidad de las **hipótesis** a medida que se obtienen nuevas **evidencias**. Esta metodología nos permite asignar probabilidades a las hipótesis y abordar preguntas que el enfoque **frecuentista** no puede resolver. Por ejemplo, el concepto de asignar una probabilidad a una hipótesis no está contemplado en el paradigma frecuentista. [Ber18]

El **Teorema de Bayes**, que es un resultado de gran importancia en la estadística inferencial y numerosos modelos avanzados de aprendizaje automático, ha sido visto en una gran cantidad de ocasiones durante el grado. Nos da la probabilidad condicionada de suceso A dado otro evento B:

$$P(A | B) = \frac{P(A \cap B)}{P(B)}. \quad (2.2)$$

De donde se obtiene fácilmente que

$$P(A | B) = \frac{P(B | A)P(A)}{P(B)}.$$

A continuación se definen algunos conceptos claves en la probabilidad bayesiana: La **función de verosimilitud**, también conocida simplemente como verosimilitud, representa la función de densidad de los datos observados en función de los parámetros de un modelo estadístico. En otras palabras, representa cuan plausibles son los datos observados bajo un modelo particular. Por ejemplo, $p(x | y)$ indica la probabilidad de observar los datos x bajo la suposición de que y es el valor real del parámetro.

La **distribución a priori** representa nuestro grado de creencia inicial sobre un parámetro o una hipótesis antes de observar datos. Es una distribución que refleja nuestro conocimiento previo o nuestras suposiciones antes de realizar cualquier experimento o análisis, es decir, cuantifica la plausibilidad a priori de la hipótesis.

La **distribución a posteriori** se actualiza con la nueva evidencia (datos observados) y combina la información de la distribución a priori con la verosimilitud de los datos. Representa nuestras creencias actualizadas después de haber observado los datos. Utiliza el Teorema de Bayes para calcular la probabilidad condicional del parámetro dados los datos observados.

2. Fundamentos Teóricos

La **verosimilitud marginal**, también conocida como evidencia, es la probabilidad de los datos observados bajo un modelo dado, integrando sobre todas las posibles configuraciones de los parámetros del modelo.

Con el Teorema de Bayes se relacionan los 4 conceptos definidos anteriormente:

$$\text{posteriori} = \frac{\text{verosimilitud} \times \text{priori}}{\text{verosimilitud marginal}}. \quad (2.3)$$

O dicho de otra forma:

$$P(\text{hipótesis} | \text{datos}) = \frac{P(\text{datos} | \text{hipótesis}) \times P(\text{hipótesis})}{P(\text{datos})} [\text{Ber18}],$$

donde

- $P(\text{datos} | \text{hipótesis})$ es la verosimilitud de los datos dados la hipótesis: 'Si la hipótesis es verdadera, ¿cuál es la probabilidad de observar estos datos?'.
- $P(\text{hipótesis})$ es la probabilidad a priori de la hipótesis: 'A priori, ¿cuál es la probabilidad de que se cumpla la hipótesis?'.
- $P(\text{datos})$ es la probabilidad de observar los datos, independientemente de la hipótesis especificada.
- $P(\text{hipótesis} | \text{datos})$ es la probabilidad a posteriori de una hipótesis dados unos datos observados.

Veamos un claro ejemplo gráfico sobre como cambia una distribución a priori tras observar unos datos. Este ejemplo está basado en la *Figura 1.1* del libro *Gaussian processes for machine learning* de Rasmussen y Williams [RW+06]:

Ejemplo 2.1. Se muestran varias funciones de muestra obtenidas al azar de la distribución a priori sobre funciones especificadas por un proceso gaussiano particular que favorece funciones suaves, con media 0 y kernel RBF. Esta distribución a priori representa nuestras creencias previas sobre el tipo de funciones que esperamos observar, antes de ver cualquier dato. En ausencia de conocimiento contrario, y por simplicidad, se asume que el valor promedio de las funciones de muestra en cada x es cero. Esta elección no supone pérdida de generalidad puesto que los datos observados siempre pueden normalizarse para tener media 0. Aunque las funciones aleatorias específicas representadas en la primera imagen no tienen una media de cero, la media de los valores de $f(x)$ para cualquier x fijo tenderá a cero, independientemente de x , a medida que se generen más y más funciones. En cualquier valor de x , también podemos caracterizar la variabilidad de las funciones de muestra calculando la varianza en ese punto. La región grisácea denota el doble de la desviación estándar puntual; en este caso, utilizamos un proceso gaussiano que especifica que la varianza previa no depende de x . También se muestra la distribución a posteriori suponiendo la observación de 2 puntos.

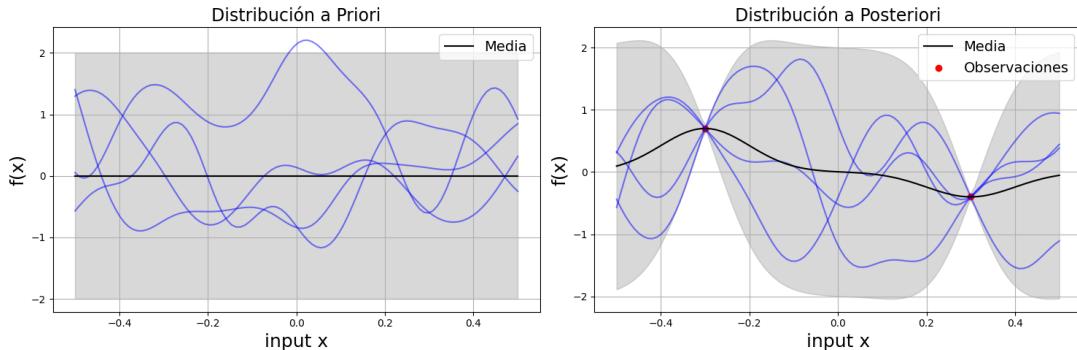


Figura 2.2.: La primera imagen muestra en azul cuatro muestras tomadas de la distribución a priori. La segunda imagen muestra la situación después de observar dos puntos. En ambos gráficos la predicción media se visualiza como una línea negra y cuatro muestras de la posterior se exponen como líneas azules. La región sombreada denota el doble de la desviación estándar en cada valor de entrada x . Código de programación propio, disponible en este [enlace](#).

2.3. Aprendizaje Automático Supervisado

El **Aprendizaje Automático**, más conocido por su término en inglés, **Machine Learning**, es uno de los campos más importantes dentro de la **Inteligencia Artificial**. Busca dotar a las máquinas de la capacidad de aprender automáticamente a partir de **datos**, utilizando un conjunto de observaciones para descubrir un proceso subyacente mediante **algoritmos de aprendizaje** [AMMIL12].

Esto es especialmente útil en situaciones donde existe un patrón que no puede describirse fácilmente de manera matemática, pero que una máquina puede discernir a partir de un conjunto de datos. Por ejemplo, un recomendador de películas: Nuestros gustos no son arbitrarios ya que siguen una serie de patrones, pero no podemos definirlos matemáticamente. Sin embargo, podemos proporcionar ejemplos de películas que nos han gustado, y a la hora de hacer recomendaciones más precisas el algoritmo de aprendizaje automático puede identificar y aprovechar esos patrones.

En los últimos años el Aprendizaje Automático ha experimentado un gran auge y ha ganado una mayor relevancia debido a varios factores:

- Disponibilidad de grandes cantidades de datos para entrenar modelos.
- Avances en tecnología y hardware que han permitido el procesamiento más rápido de grandes conjuntos de datos.
- Mejoras en algoritmos y técnicas de aprendizaje automático que han permitido obtener mejores resultados en una amplia gama de aplicaciones.
- Mayor concienciación, estudio y comprensión general de las capacidades del Aprendizaje Automático.

2. Fundamentos Teóricos

Sin embargo este término no es algo nuevo, en el sentido de que se ha estado desarrollando y estudiando durante décadas. Ya en el año 1959 el informático americano **Arthur Samuel** [Sam59], pionero en la materia al crear un programa que podía jugar a las damas, lo definió de la siguiente forma:

'El Aprendizaje Automático es el campo del estudio que confiere a los ordenadores la capacidad de aprender sin ser programados explícitamente'.

Dentro del campo del Aprendizaje Automático se pueden distinguir dos enfoques principales: el **aprendizaje basado en instancias** y el **aprendizaje basado en modelos**. En el primero, las predicciones se realizan considerando la similitud entre las nuevas instancias y las ya conocidas. El ejemplo más conocido es el algoritmo de los **K-Vecinos Más Cercanos (KNN)**. En contraste, en el aprendizaje basado en modelos se construye un modelo matemático o estadístico que generaliza los patrones presentes en los datos de entrenamiento para hacer predicciones en nuevas instancias. En este trabajo, nos centraremos en este último enfoque, empleando Procesos Gaussianos para la creación de modelos.

Además, dependiendo de la naturaleza de los datos disponibles y del tipo de tarea que se desea realizar, destacan los siguientes paradigmas del Aprendizaje Automático:

- **Aprendizaje Supervisado:** Se entrena un modelo utilizando ejemplos etiquetados, es decir, que cada ejemplo del entrenamiento consta de una entrada y la salida deseada correspondiente. El objetivo del aprendizaje supervisado es aprender una función que pueda mapear las entradas a las salidas correctas.
- **Aprendizaje por Refuerzo:** Casos en los que los datos de entrenamiento no incluyen la salida correcta para cada entrada, sino que evalúan diferentes acciones tomadas en situaciones específicas, dando una medida de su efectividad. El algoritmo de aprendizaje por refuerzo utiliza esta información para aprender qué acciones son más beneficiosas en situaciones similares en el futuro.
- **Aprendizaje No Supervisado:** Es un enfoque en el cual el conjunto de datos de entrenamiento no contiene ninguna información de salida. Es decir, solo se proporcionan ejemplos de entrada, sin etiquetas asociadas. En este contexto, el objetivo es descubrir patrones y estructuras en los datos de entrada sin la guía explícita de las salidas esperadas.

En este trabajo abordaremos situaciones de aprendizaje supervisado: partiremos de un conjunto de datos en el que dada una entrada x conoceremos su etiqueta y . A las entradas x también se les conocerá como **input**, mientras que las etiquetas se denominarán **output (salida)** o **target**. Durante el entrenamiento, el modelo ajusta sus parámetros para minimizar la discrepancia entre las salidas predichas por el modelo y las salidas reales proporcionadas en los ejemplos etiquetados. Para cuantificar esa discrepancia se utilizarán las conocidas como **funciones de pérdida**, y una vez que el modelo ha sido entrenado debería ser capaz de generalizar para hacer predicciones sobre nuevos datos no vistos. Dependiendo del tipo de salida que se esté prediciendo, distinguiremos dos categorías principales de problemas:

Por un lado están los **problemas de clasificación** [CMo7]. Buscan desarrollar un modelo capaz de predecir una variable categórica o discreta, que pertenece a un conjunto finito de

clases, utilizando una serie de características como entrada. En este trabajo se tratarán los dos principales tipos de problemas de clasificación: los **binarios**, donde se tienen únicamente 2 clases distintas y los **multiclasses**, en los que se tienen más de 2 clases diferentes. Un ejemplo común de problema de clasificación multiclasa es clasificar dígitos escritos a mano del 0 al 9, mientras que un problema típico de clasificación binaria podría ser el clasificar un correo electrónico como '*spam*' o '*no spam*'. Se tratarán este tipo de problemas en el Capítulo 4.

Por otro lado se tienen los **problemas de regresión**, que son aquellos en los que se busca predecir un valor numérico continuo como resultado. En este tipo de problemas, el objetivo es encontrar una relación entre un conjunto de características y una variable dependiente, la cual es continua y puede tomar cualquier valor dentro de un codominio específico. Por ejemplo, la variable a predecir podría ser el precio de una vivienda, donde las características podrían incluir el tamaño del terreno, el número de habitaciones, la ubicación, etc. El objetivo final es encontrar un modelo que pueda hacer predicciones precisas sobre el valor de la variable dependiente en función de las características proporcionadas. Estos problemas se tratarán en profundidad en el Capítulo 3.

El término estadístico de regresión, proveniente de la propia palabra que significa 'acción de volver hacia atrás', fue acuñado por primera vez en 1889 por el polímata británico **Francis Galton** en su obra '*Natural inheritance*' [Gal89]. Galton, primo del reconocido científico Charles Darwin, observó una relación lineal entre la altura de los padres y la de sus hijos, resaltando el hecho de que los hijos de padres altos son, en promedio, más bajos que sus padres y que los hijos de padres pequeños son, en promedio, más altos que sus padres. Este fenómeno, inicialmente denominado 'regresión hacia la mediocridad', se conoce hoy como 'regresión hacia la media' o simplemente 'regresión a la media' [Gon09].

Para verificar la eficacia y robustez de los modelos existen diversos protocolos de **validación experimental** que consisten en evaluar los modelos en nuevos datos no vistos durante el entrenamiento. Las técnicas más utilizadas son las siguientes:

- **Hold-out:** Consiste en dividir el conjunto de datos en dos grupos: uno para entrenamiento y otro para validación. Es una idea fácil e intuitiva, pero se debe tener cuidado ya que los datos separados para validar el conjunto deben ser realmente representativos del conjunto de datos. En caso contrario, el resultado de esta validación puede dar una falsa impresión de la eficacia del modelo, llevando a una sobreestimación o subestimación de su rendimiento real.
- **Cross-validation:** Es una evolución de *hold-out*. En este caso, el conjunto de datos se divide en k subconjuntos o pliegues y el modelo se evalúa k veces, utilizando cada vez un conjunto diferente para validar y los $k - 1$ restante para el entrenamiento. Posteriormente, se promedian los resultados obtenidos para dar una estimación más robusta y confiable del rendimiento del modelo, reduciendo el sesgo asociado a una sola partición de datos y proporcionando una mejor medida de su capacidad de generalización.
- **Leaving-one-out:** Es un caso extremo del *cross-validation*, donde el número de pliegues es igual al número de muestras del conjunto de datos. Es decir, se crean tantos conjuntos de validación como datos tenga el conjunto, cada vez tomando una instancia diferente. Esto, obviamente, es muy costoso computacionalmente cuando el conjunto no sea de un tamaño muy limitado.

2. Fundamentos Teóricos

- **Conjunto de solo validación:** Se trata de un caso concreto de *Hold-out*, que consiste en dividir el conjunto de datos en dos subconjuntos disjuntos: uno para entrenamiento y otro para test. Sin embargo, la clave se encuentra en que dentro del conjunto de entrenamiento se vuelve a realizar validación. Este enfoque es el más común en Aprendizaje Automático.

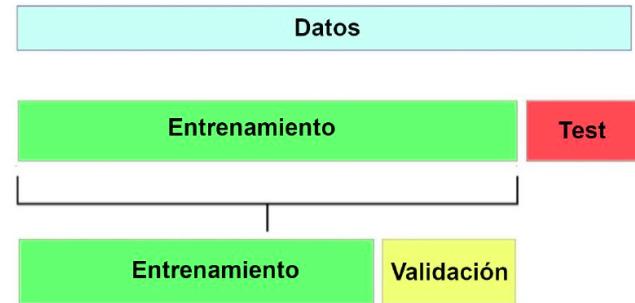


Figura 2.3.: Distribución de los datos en esta técnica.

Para evaluar los modelos se necesitan métricas que cuantifiquen de alguna forma su rendimiento. Es interesante utilizar métricas que evalúen distintos aspectos, para así enriquecer la interpretación de los resultados. Algunas de las métricas más utilizadas para modelos de regresión son:

- **Error Cuadrático Medio (ECM o MSE por sus siglas en inglés):** Cuanto menor sea el valor del MSE, mejor será el rendimiento del modelo, ya que indica que las predicciones están más cerca de los valores reales. Es una medida de dispersión que penaliza de manera cuadrática las diferencias entre las predicciones y los valores reales, lo que significa que los errores más grandes tienen un impacto significativamente mayor en el resultado final.

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}$$

- **Error Absoluto Medio (EAM / MAE):** calcula la diferencia absoluta promedio entre las predicciones del modelo y los valores reales en el conjunto de datos de prueba. El MAE trata todos los errores por igual, no penaliza tanto los errores más grandes, como sí hace el MSE.

$$MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n}$$

- **R² (Coeficiente de determinación):** Representa la proporción de la varianza total de la variable dependiente que es explicada por el modelo. Un valor de 1 indica un ajuste perfecto, mientras que un valor de 0 indica que el modelo no explica ninguna variabilidad. R² no indica la bondad absoluta del modelo, sino su bondad relativa en comparación con un modelo base.

Sean la varianza de la variable dependiente $\sigma_y^2 = \frac{\sum_{i=1}^n (y_i - \bar{y})^2}{n}$ y la varianza residual

$\sigma_r^2 = ECM = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}$, se determina el coeficiente de determinación como

$$R^2 = 1 - \frac{\sigma_r^2}{\sigma_y^2}.$$

Por otro lado, en problemas de clasificación destacan las siguientes métricas:

- **Accuracy (exactitud):** Mide la proporción de predicciones correctas sobre el total de predicciones realizadas. Es una métrica simple y muy utilizada cuando las clases están balanceadas, pero puede ser engañosa si hay un desbalance significativo entre las clases.

$$\text{Accuracy} = \frac{\text{Número de predicciones correctas}}{\text{Número total de predicciones}}$$

- **Precisión:** Se enfoca en la exactitud de las predicciones positivas del modelo. Se define como la proporción de verdaderos positivos (TP) sobre la suma de verdaderos positivos y falsos positivos (FP). Es útil cuando el costo de los falsos positivos es alto.

$$\text{Precisión} = \frac{TP}{TP + FP}$$

- **Recall (Sensibilidad):** Mide la capacidad del modelo para identificar correctamente todas las instancias positivas. Se define como la proporción de verdaderos positivos (TP) sobre la suma de verdaderos positivos y falsos negativos (FN).

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-Score:** Es la media armónica de la precisión y la sensibilidad. Tiene especial utilidad en conjuntos de datos desbalanceados.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precisión} \cdot \text{Recall}}{\text{Precisión} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

Otra herramienta ampliamente utilizada y valorada en la evaluación de modelos en problemas de clasificación es la **matriz de confusión**, la cual proporciona una visualización clara del desempeño del modelo al mostrar el número de ejemplos asignados a cada clase en función del valor real de la clase. Permite visualizar dónde se equivoca el modelo.

Cuando se trabaja con Aprendizaje Automático es natural preguntarse acerca de cuál es el mejor modelo. **David Wolpert** y **William MacReady** proporcionaron una respuesta en 1997 con una serie de resultados conocidos como '**No Free Lunch**' (NFL). Demuestran que si no se hace ninguna suposición sobre los datos, no hay ninguna razón para preferir un modelo a otro. Es decir, no hay ningún modelo que asegure de antemano su superioridad, de ahí el nombre de *No Free Lunch*. La única manera de determinar con certeza cuál es el mejor modelo es evaluar todos los disponibles. Sin embargo, dado que esta tarea es imprácticable, en la práctica se realizan suposiciones sobre los datos y se evalúan únicamente algunos modelos considerados razonables [AAPV19].

2. Fundamentos Teóricos

En el apartado anterior se introdujo una rama muy interesante de la Estadística como es la Probabilidad Bayesiana. Hay que resaltar que gran parte de la teoría básica y muchos algoritmos son compartidos entre la Estadística y el Aprendizaje Automático. Las diferencias principales radican en los tipos de problemas abordados y los objetivos del aprendizaje. En Estadística, uno de los principales enfoques es comprender los datos y las relaciones a través de modelos que proporcionan resúmenes aproximados, como relaciones lineales o independencias. Por otro lado, en el Aprendizaje Automático, los objetivos principales son hacer predicciones precisas y comprender el comportamiento de los algoritmos de aprendizaje. Estos objetivos diferentes han dado lugar a desarrollos específicos en cada campo: por ejemplo, las Redes Neuronales se utilizan ampliamente como aproximadores de funciones en el Aprendizaje Automático, pero algunos estadísticos pueden encontrar dificultades para interpretar estos modelos debido a su naturaleza de caja negra.

2.4. Procesos Gaussianos

A continuación, definiremos el elemento clave de este trabajo: los **Procesos Gaussianos**, que son un tipo concreto de procesos estocásticos. Al ser Gaussianos cuentan con unas características particulares que hacen que la tarea de aprendizaje y los cálculos necesarios para realizar inferencia sean relativamente sencillos, de ahí su uso en Aprendizaje Automático Supervisado.

Consideremos un sistema que puede estar en uno de varios estados predefinidos. Supongamos que el sistema evoluciona de un estado a otro a lo largo del tiempo según una cierta ley de movimiento, y sea X_t el estado del sistema en el tiempo t . Si la evolución del sistema no es determinista, sino que es impulsada por algún mecanismo aleatorio, entonces X_t puede ser considerado como una variable aleatoria para cada valor de t . Esta colección de variables aleatorias constituye un proceso estocástico, que sirve como modelo para representar la evolución aleatoria de un sistema a lo largo del tiempo. Generalmente, las variables aleatorias en un proceso no son independientes entre sí, sino que están relacionadas de alguna manera particular. Las diversas formas en que estas dependencias pueden surgir son características distintivas de diferentes tipos de procesos. Más precisamente, la definición de un proceso estocástico se basa en un espacio de probabilidad (Ω, \mathcal{F}, P) y se puede expresar de la siguiente manera.

Definición 2.6. Un **Proceso Estocástico** es una colección de variables aleatorias $X_t : t \in T$ parametrizada por un conjunto T , llamado *espacio paramétrico*. Cada variable aleatoria X_t representa una observación en un instante t y toma valores en un conjunto S , conocido como espacio de estados [Rin12].

Observación 2.4. El conjunto T podrá ser tanto **discreto** como **continuo**.

Uno de los tipos de Procesos Estocásticos más relevantes son los Procesos Gaussianos, también conocidos como Procesos Estocásticos Gaussianos. Este tipo de proceso es ampliamente utilizado en el ámbito del Aprendizaje Automático y la modelización estadística debido a sus propiedades matemáticas y su capacidad para capturar la incertidumbre en los datos de forma efectiva.

Definición 2.7. Un **Proceso Gaussiano** es un conjunto de variables aleatorias tal que cualquier subconjunto finito suyo tiene una distribución gaussiana multivariante [RW⁺06].

Esto significa que, para cualquier conjunto finito de puntos en el dominio del proceso, las correspondientes variables aleatorias siguen una distribución gaussiana conjunta. Esta propiedad fundamental hace que los procesos gaussianos sean especialmente útiles para modelar sistemas complejos y para realizar inferencias sobre los mismos.

Observación 2.5. Si el conjunto de índices de un Proceso Gaussiano es finito, entonces es simplemente una Distribución Normal Multivariante.

Observación 2.6. Como un proceso Gaussiano es un conjunto de variables aleatorias que siguen una distribución normal multivariante, entonces cumple la propiedad de la marginal de las MND.

Los procesos gaussianos se caracterizan por su capacidad para proporcionar una descripción completa de la **incertidumbre** asociada a las predicciones o estimaciones realizadas en el contexto del modelado estadístico. Esto se debe a que, además de estimar la media o valor esperado de una función, los procesos gaussianos también proporcionan información sobre la variabilidad de las predicciones a través de la covarianza asociada.

Las funciones de **media** y **covarianza** caracterizan por completo un Proceso Gaussiano $f(x)$ y se denotan respectivamente como $m(x)$ y $k(x, x')$. Se definen de la siguiente manera:

$$\begin{aligned} m(x) &= \mathbb{E}[f(x)] \\ k(x, x') &= \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \end{aligned} \tag{2.4}$$

El Proceso Gaussiano se escribirá como $f(x) \sim \mathcal{GP}(m(x), k(x, x'))$. Para mayor simplicidad, trabajaremos con media $m(x) = 0$.

Normalmente los procesos estocásticos trabajan con el tiempo, de ahí que se utilice como conjunto de índices T , sin embargo para el uso de Procesos Gaussianos en Aprendizaje Automático, es más común el uso de \mathcal{X} . Nosotros trabajamos con funciones, por lo que es más intuitivo utilizar $f(x)$ en lugar de X_t . El input en nuestro caso puede ser más general que el tiempo, como por ejemplo \mathbb{R}^D .

El enfoque de un proceso gaussiano ofrece ciertas ventajas respecto a los modelos paramétricos más tradicionales. Por ejemplo, al no ser un modelo paramétrico, no nos enfrentamos a la preocupación de si el modelo puede ajustarse adecuadamente a los datos, como podría suceder al intentar ajustar un modelo lineal a datos altamente no lineales. Incluso con un gran número de observaciones, aún conservamos cierta flexibilidad en la forma de las funciones.

Podemos visualizar esta idea imaginando que generamos muchas funciones aleatorias a partir de una distribución de probabilidad inicial (llamada ‘a priori’) y luego descartamos aquellas que no concuerdan con las observaciones. Si bien esta técnica es teóricamente válida para la inferencia, en la práctica resulta poco práctica para la mayoría de los propósitos debido a la complejidad de los cálculos necesarios para cuantificar estas propiedades.

2. Fundamentos Teóricos

En resumen, un Proceso Gaussiano es una generalización de la distribución de probabilidad gaussiana. Mientras que una distribución de probabilidad describe variables aleatorias que pueden ser escalares o vectores (en el caso de distribuciones multivariantes), un proceso estocástico controla las propiedades de las funciones. Simplificando un poco desde el punto de vista matemático, podemos concebir una función como un vector extremadamente largo, donde cada entrada en el vector representa el valor de la función $f(x)$ en un punto específico x . Aunque esta conceptualización puede parecer un poco ingenua, resulta sorprendentemente adecuada para nuestros propósitos.

2.4.1. Usos y contexto histórico

Las predicciones con Procesos Gaussianos no son precisamente un tema nuevo en la investigación. De hecho, su fundamento teórico se remonta a la década de 1940, con contribuciones importantes de destacados matemáticos como **Wiener** en 1949 [Wie49] y **Kolmogorov** en 1941 [Kol41]. Sentaron las bases para la comprensión y el desarrollo posterior de los Procesos Gaussianos, estableciendo los fundamentos que aún son fundamentales en la teoría y aplicación de este enfoque en la actualidad.

Los Procesos Gaussianos han demostrado ser herramientas sumamente útiles en diversos campos, entre los que destaca la **geoestadística**. Entre las aplicaciones más destacadas se encuentra el *kriging*, un método utilizado para la interpolación espacial. Aunque el *kriging* fue inicialmente desarrollado para aplicaciones en geoestadística, es un método de **interpolación** estadística general que puede emplearse en cualquier disciplina para datos muestreados de campos aleatorios que cumplan con las suposiciones matemáticas apropiadas. Es útil en situaciones donde se han recolectado datos relacionados espacialmente (en 2-D o 3-D) y se necesitan estimaciones de datos 'de relleno' en las ubicaciones intermedias entre las mediciones reales. Este método es muy útil en la elaboración de mapas de riesgo sísmico, así como en la evaluación de la calidad del aire, entre otras aplicaciones.

El término *kriging* proviene de Daniel G. Krige [Kri51], un ingeniero de minas sudafricano que desarrolló y popularizó esta técnica en la década de 1950. Su trabajo pionero ha facilitado la aplicación efectiva de los procesos gaussianos en la estimación y predicción de fenómenos espaciales en diversos contextos geoespaciales.

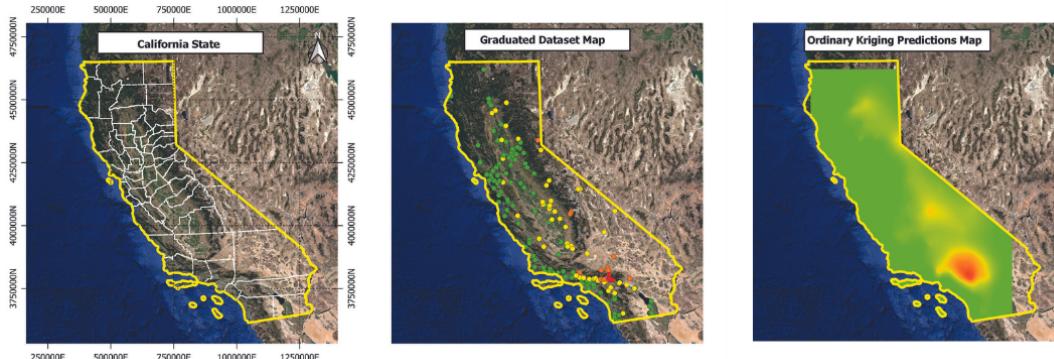


Figura 2.4.: Ejemplo de interpolación mediante kriging [Ced23].

En **meteorología** [Dal93], los procesos gaussianos también se emplean en tareas de interpolación, siguiendo un enfoque similar al mencionado anteriormente en el ámbito de la geoestadística. Por ejemplo, se pueden utilizar para crear mapas del tiempo registrado en toda una región a partir de datos recopilados por estaciones meteorológicas distribuidas de manera dispersa en el área.

Además, los Procesos Gaussianos son útiles para la predicción meteorológica[SWX14] y el análisis de la incertidumbre asociada a estas predicciones.

Con el tiempo, se fue entendiendo que la predicción mediante Procesos Gaussianos podía aplicarse en un contexto de regresión general. En el año 1978 **O'Hagan** [O'H78] expuso la teoría general, tal como se describe en las ecuaciones (3.10) y (3.11), y las aplicó a diversos problemas de regresión unidimensional. En 1989 **Sacks** et al. [SW89] describieron los Procesos Gaussianos en el contexto de experimentos computacionales donde las observaciones y no tienen ruido y discutieron varias direcciones interesantes, como la optimización de los parámetros de la función de covarianza.

Rasmussen y **Williams**, autores del principal trabajo sobre Procesos Gaussianos en Aprendizaje Automático [RW⁺06], han colaborado en varios trabajos conjuntos y también han realizado contribuciones individuales significativas en este campo. Su interés en los modelos de Procesos Gaussianos en el contexto del aprendizaje automático surgió en 1994 mientras eran estudiantes de posgrado en un grupo de investigación sobre **Redes Neuronales** de la **Universidad de Toronto**. En esta época se estaba dando un gran desarrollo en el campo de las redes neuronales, donde se evidenciaron las conexiones con la física estadística, modelos probabilísticos y estadísticas. Además, los primeros algoritmos de aprendizaje basados en kernel empezaban a ganar popularidad.

Muchos investigadores estaban encontrándose con algunas limitaciones prácticas de las redes neuronales, al tener que tomar numerosas decisiones para ser usadas (como elegir arquitectura, funciones de activación y tasas de aprendizaje), sin disponer de un marco sólido para orientar estas decisiones. En este contexto, se examinó el marco probabilístico mediante aproximaciones, como las propuestas por **MacKay** [Mac92], así como mediante **métodos de Montecarlo** basados en **cadenas de Markov**, según lo desarrollado por **Neal** [Nea96]. Neal, que estaba en el mismo grupo de investigación que Williams y Rasmussen, en su tesis trató de demostrar que al utilizar el formalismo bayesiano, no necesariamente se presentan problemas de sobreajuste cuando los modelos se vuelven grandes. En su trabajo señaló que algunas de sus redes se transformaban en procesos gaussianos en el límite de tamaño infinito, y planteó la posibilidad de que existan formas más simples de realizar inferencia en dicho caso.

En 1996, inspirados por esta conexión con redes neuronales infinitas, Rasmussen y Williams describieron el uso de Procesos Gaussianos en problemas de regresión y cómo optimizar los parámetros de la función de covarianza.

3. Procesos Gaussianos en Problemas de Regresión

Como se comentó en el capítulo anterior, los problemas de regresión se refieren a los casos de predicción continuos. En este capítulo veremos el uso de los Procesos Gaussianos en este tipo de problemas. En la Sección 3.2 vamos a considerar los Procesos Gaussianos como distribuciones sobre funciones, realizando inferencia directamente sobre este espacio de funciones. Sin embargo, este concepto puede ser complicado de entender en un principio, por lo que anteriormente en la Sección 3.1 trataremos una visión más sencilla y familiar, como es el Espacio de Parámetros, para así introducir a continuación los Procesos Gaussianos.

3.1. Espacio de Parámetros: Modelo Lineal Bayesiano

En este apartado examinaremos un enfoque bayesiano del problema común de regresión lineal, en el cual se emplea una combinación lineal de las variables de entrada del modelo. Sus principales virtudes se encuentran en su fácil implementación e interpretabilidad. Sin embargo, cuenta con el claro inconveniente de que su flexibilidad es limitada; así que si la relación entre el *input* y el *target* no puede aproximarse razonablemente mediante una función lineal, el modelo producirá predicciones deficientes. En la sección 3.1.1 se solventará este problema a través del estudio de la proyección al espacio de características, de modo que ya se podrían modelar relaciones no-lineales entre y y x .

Partiremos de un **Dataset** $\mathcal{D} = \{(x_i, y_i) \mid i = 1, \dots, n\}$, que será nuestro conjunto de entrenamiento con n observaciones. Cada x_i será un vector de entrada de dimensión m , mientras que la variable dependiente y_i , será un escalar, que representará cada una de las salidas empleadas en el entrenamiento. Uniendo todos los *inputs* del entrenamiento se formará una **matriz de diseño** X de dimensión $m \times n$, $X \in \mathcal{M}^{m \times n}$. De la misma forma, juntando todos los *outputs*, crearemos un vector y . Por lo tanto, también se podrá notar el Dataset como $\mathcal{D} = (X, y)$.

Primero asumiremos el **modelo de regresión lineal estándar con ruido Gaussiano**,

$$f(x) = x^T w, \quad y = f(x) + \epsilon,$$

donde x es el *input*, y el *output*, w un vector de pesos y ϵ será el ruido, representado por una variable aleatoria que sigue una distribución Gaussiana independiente e idénticamente distribuida

$$\epsilon \sim \mathcal{N}(0, \sigma_n^2).$$

A partir del modelo con la suposición del ruido ϵ , se define la **función de verosimilitud** como la función de densidad de las observaciones dados los parámetros. Esta, describe la desviación de la función subyacente fruto de posibles mediciones ruidosas y sigue una

3. Procesos Gaussianos en Problemas de Regresión

Distribución Normal de media $X^T w$ y varianza $\sigma_n^2 I$:

$$\begin{aligned} p(y | X, w) &= \prod_{i=1}^n p(y_i | x_i, w) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{(y_i - x_i^T w)^2}{2\sigma_n^2}\right) = \\ &= \frac{1}{(2\pi\sigma_n^2)^{\frac{n}{2}}} \exp\left(-\frac{1}{2\sigma_n^2} |y - X^T w|^2\right) = \mathcal{N}(y | X^T w, \sigma_n^2 I). \end{aligned} \quad (3.1)$$

En la inferencia bayesiana, como vimos en el apartado 2.2, se necesita una **distribución a priori** donde se muestran los conocimientos o creencias sobre de los parámetros. En este caso, asumiremos que los pesos, w , se distribuyen de acuerdo a una normal de media 0 y varianza Σ_p ,

$$w \sim \mathcal{N}(0, \Sigma_p). \quad (3.2)$$

Para obtener la **distribución a posteriori** se lleva a cabo inferencia bayesiana sobre los pesos del modelo. De la fórmula del Teorema de Bayes, como se vio en (2.3), se tiene

$$p(w | y, X) = \frac{p(y | X, w)p(w)}{p(y | X)},$$

donde el denominador $p(y | X)$, conocido como **verosimilitud marginal**, es una constante normalizadora (al ser una función de distribución, la integral sobre $p(w | y, X)$ deberá ser 1) que es independiente del parámetro de los pesos, w :

$$p(y | X) = \int p(y | X, w)p(w) dw. \quad (3.3)$$

Por lo tanto, podemos decir que la distribución a posteriori, $p(w | y, X)$, es proporcional a la distribución a priori por la función de verosimilitud [BNo6]. Es decir, un producto de Distribuciones Normales:

$$\begin{aligned} p(w | y, X) &\propto \exp\left(-\frac{1}{2\sigma_n^2}(y - X^T w)^T (y - X^T w)\right) \exp\left(-\frac{1}{2} w^T \Sigma_p^{-1} w\right) \\ &\propto \exp\left(-\frac{1}{2}(w - \bar{w})^T \left(\frac{1}{\sigma_n^2} XX^T + \Sigma_p^{-1}\right)(w - \bar{w})\right), \end{aligned}$$

donde $\bar{w} = \sigma_n^{-2} (\sigma_n^{-2} XX^T + \Sigma_p^{-1})^{-1} X y$. Además, definimos la matriz $A = \sigma_n^{-2} XX^T + \Sigma_p^{-1}$, que nos permite determinar que la distribución $p(w | y, X)$ adopta la siguiente Distribución Normal:

$$p(w | y, X) \sim \mathcal{N}\left(\frac{1}{\sigma_n^2} A^{-1} X y, A^{-1}\right). \quad (3.4)$$

Esta distribución a posteriori se puede calcular de forma directa y cerrada, ya que la función de verosimilitud es normal en el caso de la regresión y no hay problemas a la hora de realizar la integral. El producto de distribuciones normales es otra distribución normal, por lo que todo sale de forma correcta. Como se verá más adelante en el Capítulo 4, en los problemas de clasificación la función de verosimilitud no será gaussiana y esto nos traerá bastantes problemas.

Esta distribución a posteriori, al ser normal, verifica que su media coincide con su moda, como se vio en el apartado 2.1. Dar como predicción la moda es lo que suele conocerse

como estimación **Maximum A Posteriori** (MAP), pero en este escenario bayesiano no tiene demasiada importancia. En este caso, debido a las simetrías en el modelo y la distribución a posteriori, sucede que la media de la distribución predictiva es la misma que la predicción en la media de la distribución a posteriori. Sin embargo, esto no es así en general.

En contraste con los enfoques no bayesianos, que suelen elegir un único valor basado en un criterio definido, aquí se promedian todos los valores: Para realizar predicciones en el conjunto de prueba, con unos nuevos inputs que se notan como x_* , se emplea una media ponderada de todos los posibles valores de los parámetros, considerando la distribución a posteriori. Esta técnica genera la **distribución predictiva**, que se notará como $f_* \triangleq f(x_*)$. Esta distribución es de nuevo Gaussiana, con media dada por la media de la distribución a posteriori de los pesos multiplicada por el input de test x_* . La covarianza es una forma cuadrática del input x_* con la matriz de covarianza a posteriori, lo que muestra que las incertidumbres predictivas aumentan con la magnitud del input de prueba, como se esperaría en un modelo lineal.

$$p(f_* | x_*, X, y) = \int p(f_* | x_*, w) p(w | X, y) dw = \mathcal{N}\left(\frac{1}{\sigma_n^2} x_*^T A^{-1} X y, x_*^T A^{-1} x_*\right)$$

Ejemplo 3.1. Vamos a calcular un modelo básico lineal bayesiano, $f(x) = w_1 + w_2 x$, mostrando los resultados en las Figuras 3.1 y 3.2 que vienen a continuación. Se parte de una distribución a priori $p(w) \sim \mathcal{N}(0, I)$, como en (3.2). Para calcular la distribución de verosimilitud, de acuerdo a la expresión (3.1), se utilizan los puntos de entrenamiento que se muestran en la segunda imagen de la Figura 3.2: $(-5, -5.5)$, $(2, 0.75)$ y $(5, 4.75)$. Por último, se calcula la distribución a posteriori siguiendo la expresión (3.4).

La predicción puntual del modelo vendrá dada por la media de la distribución a posteriori: $y_{pred}(x) = -0.4884 + 0.9768x$. Estos resultados, viendo la Figura 3.1, tienen sentido, ya que los tres puntos de entrenamiento están bastante alineados con el resultado.

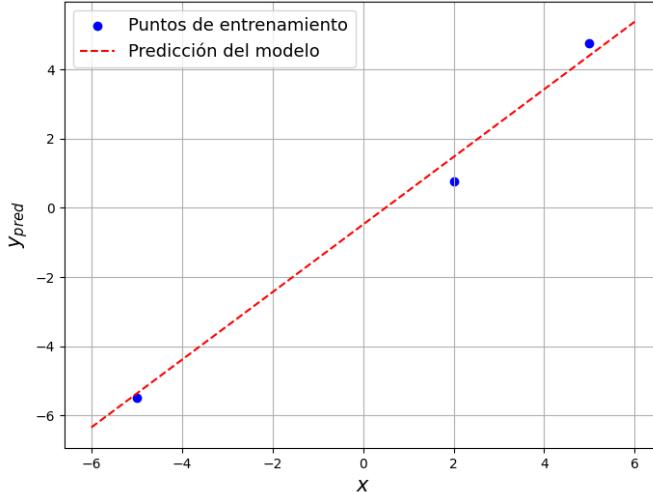


Figura 3.1.: Predicción del Modelo Lineal Bayesiano del Ejemplo 3.1. Código de programación propio, disponible en este [enlace](#).

3. Procesos Gaussianos en Problemas de Regresión

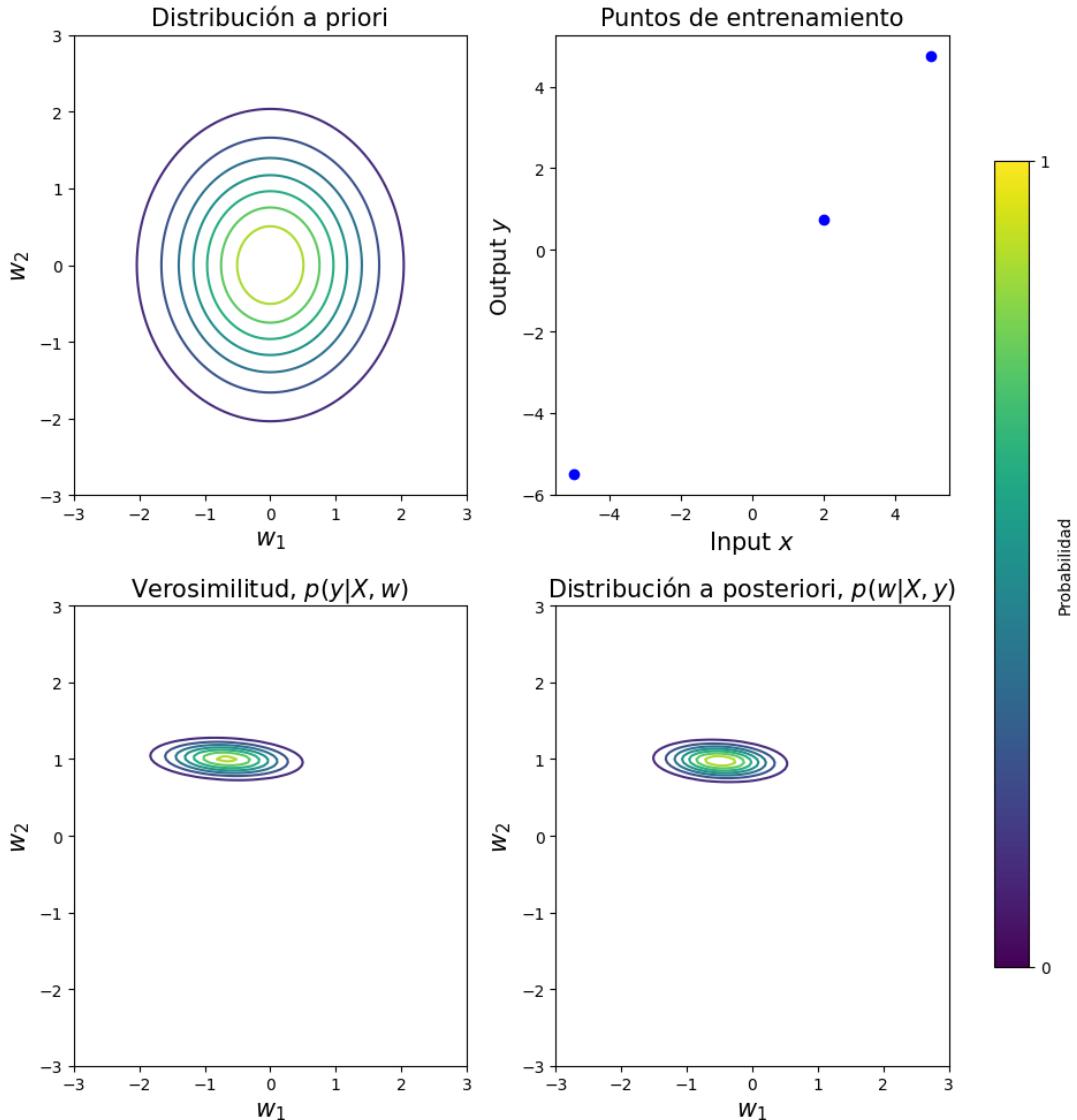


Figura 3.2.: Resultados del Ejemplo 3.1. Los colores más oscuros indican zonas de valores de w_1 y w_2 menos consistentes con los datos observados (menos probabilidad), mientras que los colores más claros indican valores más probables. Código de programación propio, disponible en este [enlace](#).

3.1.1. Proyección al Espacio de Características

En esta sección mejoraremos el modelo bayesiano lineal básico que se estudió en el apartado anterior, superando la falta de expresividad con una idea muy simple: se proyectan los inputs sobre un espacio de mayor dimensión, utilizando un conjunto de funciones de espacio de características base, y luego se aplica el modelo lineal en este espacio en lugar de hacerlo directamente sobre los inputs. En el siguiente ejemplo se ve claramente la idea:

Ejemplo 3.2. Sea el espacio de características $\phi(x) = (1, x, x^2, x^3, \dots, x^{N-1})^T$. Entonces, se po-

drá realizar regresión polinómica aplicando sobre $\phi(x)$ el modelo lineal básico ya estudiado en la sección 3.1.

Siempre y cuando el espacio de características sea **independiente de los parámetros**, w , el modelo seguirá siendo **lineal**. Entonces, sea una función $\phi(x)$ que mapea un vector input x en un espacio de características de N dimensiones y P la matriz formada por las columnas de $\phi(x), \forall x \in X$. Ahora, el modelo se definirá como:

$$f(x) = \phi(x)^T w,$$

donde x ahora es un vector de longitud N . Todo lo estudiado en la sección anterior es aplicable simplemente cambiando x por $\phi(x)$. Por lo tanto, la **distribución predictiva** ahora será

$$f_* | x_*, X, y \sim \mathcal{N}\left(\frac{1}{\sigma_n^2} \phi(x_*)^T A^{-1} P y, \phi(x_*)^T A^{-1} \phi(x_*)\right), \quad (3.5)$$

con $A = \sigma_n^{-2} P P^T + \Sigma_p^{-1}$. A es una matriz de dimensión $N \times N$, y realizar las predicciones implica calcular su inversa. Esto puede llegar a ser muy **costoso computacionalmente**, convirtiéndolo en un proceso muy poco eficiente. Este tamaño N podría llegar a ser incluso infinito. Por ello, hacemos la siguiente reformulación:

$$\begin{aligned} f_* | x_*, X, y &\sim \mathcal{N}\left(\phi(x_*)^T \Sigma_p P(K + \sigma_n^2 I)^{-1} y, \right. \\ &\quad \left. \phi(x_*)^T \Sigma_p \phi(x_*) - \phi(x_*)^T \Sigma_p P(K + \sigma_n^2 I)^{-1} P^T \Sigma_p \phi(x_*)\right), \end{aligned} \quad (3.6)$$

donde $K = P^T \Sigma_p P$. En cuanto a la expresión obtenida para la media, cabe destacar que de las definiciones de A y K se tiene que

$$\sigma_n^{-2} P(K + \sigma_n^2 I) = \sigma_n^{-2} P(P^T \Sigma_p P + \sigma_n^2 I) = A \Sigma_p P.$$

Ahora, multiplicando desde la izquierda por A^{-1} y desde la derecha por $(K + \sigma_n^2 I)^{-1}$, se obtiene que

$$\sigma_n^{-2} A^{-1} P = \Sigma_p P(K + \sigma_n^2 I)^{-1}.$$

En cuanto a la expresión de la varianza, para ver la relación entre $\phi(x_*)^T A^{-1} \phi(x_*)$ y $\phi(x_*)^T \Sigma_p \phi(x_*) - \phi(x_*)^T \Sigma_p P(K + \sigma_n^2 I)^{-1} P^T \Sigma_p \phi(x_*)$ se usa el siguiente lema, conocido como el **Lema de Identidad Matricial de Woodbury [TFPV92]**.

Lema 3.1. Sean Z, U, W y V matrices de dimensiones $n \times n$, $m \times m$, $n \times m$ y $n \times m$ respectivamente. Entonces, se verifica que

$$(Z + UWV^T)^{-1} = Z^{-1} - Z^{-1}U(W^{-1} + V^T Z^{-1}U)^{-1}V^T Z^{-1}.$$

Este lema es una versión en matrices de la **fórmula de Sherman-Morrison**. Tomando $Z^{-1} = \Sigma_p$, $W^{-1} = \sigma_n^2 I$ y $V = U = P$ se llega a la equivalencia de las expresiones (3.5) y (3.6).

Por lo tanto, con esta nueva formulación obtenemos inversas de matrices de dimensión $n \times n$. Esto, en los casos en los que $n < N$ será beneficioso si comparamos con las dimensiones $N \times N$ que obteníamos antes.

Cabe destacar que en esta formulación alternativa, el espacio de características siempre aparece como $\phi(x_*)^T \Sigma_p P$, $\phi(x_*)^T \Sigma_p \phi(x_*)$ o $P^T \Sigma_p \phi(x_*)$. Teniendo en cuenta que $P = \phi(X)$,

3. Procesos Gaussianos en Problemas de Regresión

podemos ver los productos como $\phi(x)^T \Sigma_p \phi(x')$. De tal forma, definimos una función

$$k(x, x') = \phi(x) \Sigma_p \phi(x'),$$

que se llamará **función de covarianza** o **kernel**. Esto será ampliamente estudiado en el Capítulo 5. Por ahora, nos centramos solamente en que, al ser Σ_p una matriz definida positiva, podemos tomar la descomposición :

$$\Sigma_p = UDU^T \quad (\text{siendo } D \text{ diagonal}),$$

de forma que definiendo $\psi(x) = \Sigma_p^{1/2} \phi(x)$ la función kernel se pueda ver como un producto escalar: $k(x, x') = \langle \psi(x), \psi(x') \rangle$.

Con el producto escalar de esta función kernel se podrán realizar operaciones en un espacio de características de mayor dimensión sin necesariamente tener que calcular de forma explícita las transformaciones de características. Este recurso es lo que se conoce como el *kernel trick*.

En esencia, el *kernel trick* se basa en la observación de que muchas operaciones en el espacio de características de mayor dimensión solo dependen de los productos escalares entre pares de muestras. Por lo tanto, en lugar de calcular las nuevas características y luego realizar los productos escalares en este espacio de características de alta dimensión, el *kernel trick* nos permite calcular directamente estos productos escalares en el espacio de características original [HSSo8]. Esta técnica es usada por los Procesos Gaussianos, como veremos más adelante, y nos lleva a considerar el kernel como el objeto de interés principal, mientras que su correspondiente espacio de características adquiere un papel más secundario.

3.2. Espacio de funciones

Los resultados observados en la sección anterior pueden ser obtenidos de manera equivalente utilizando **Procesos Gaussianos** para modelar distribuciones sobre funciones y realizando inferencia directamente sobre estas funciones en lugar de sus parámetros. Con este enfoque se obtiene la ventaja de poder utilizar kernels más generales que no tienen que venir necesariamente de una cantidad finita de proyecciones, aspecto clave respecto al Espacio de Parámetros.

Una función de covarianza especifica la covarianza existente entre cada par de variables aleatorias del Proceso Gaussiano, en concreto en esta sección se utiliza la '*Squared Exponential*' (**SE**), también conocida como **Radial Basis Function** (**RBF**) o **Kernel Gaussiano**. La covarianza rondará el valor 1 cuando ambos inputs sean muy cercanos y decrecerá conforme la distancia de los inputs vaya creciendo, como se puede observar en la Figura 3.3. Cabe destacar que que la covarianza entre las salidas del modelo se escribe como una función dependiente de las entradas:

$$\text{cov}(f(x_p), f(x_q)) = k(x_p, x_q) = \exp\left(-\frac{|x_p - x_q|^2}{2l^2}\right) \quad (3.7)$$

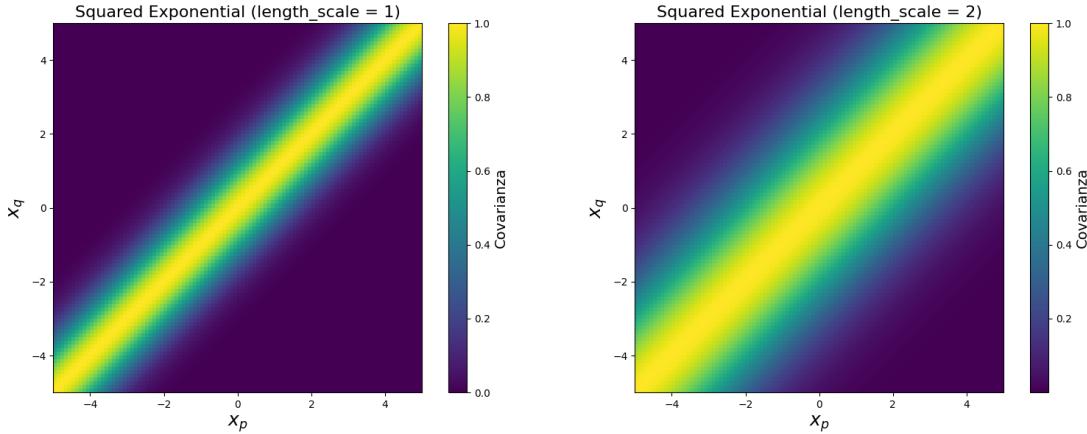


Figura 3.3.: Gráficos de ejemplo de *Squared Exponential* con valores de *length scales* 1 y 2. Código de programación propio, disponible en este [enlace](#).

El parámetro l se llama *length-scale*. Este parámetro sirve para controlar la ‘suavidad’ de las correlaciones entre pares de puntos. Un *length-scale* pequeño hace que la función de covarianza disminuya rápidamente con la distancia, lo que significa que solo los puntos muy cercanos tendrán altas correlaciones. Por otro lado, si el *length-scale* es elevado, la función variará suavemente a lo largo del espacio. Esto se puede ver claramente en la Figura 3.3.

A través de la función de covarianza podemos obtener una distribución sobre funciones. Cogemos como *inputs* un conjunto de puntos X_* y generamos una distribución normal multivariante con media 0 y matriz de covarianzas dada por la evaluación de cada par de puntos $x_p, x_q \in X_*$ según la SE definida en la expresión (3.7).

$$f_* \sim \mathcal{N}(0, K(\mathcal{X}_*, \mathcal{X}_*))$$

A partir de dicha distribución se podrán obtener predicciones de forma idéntica al Ejemplo 2.1 del apartado 2.2: se toma f_* como distribución a priori y a partir de unas observaciones se calcula la distribución a posteriori.

Ahora trabajaremos con dos posibilidades:

- Un primer acercamiento con **observaciones sin ruido**, que facilitará los cálculos pero es una coyuntura poco realista.
- Una situación de **observaciones con ruido**, lo cual será más práctico para aplicarlo a casos concretos de la vida real.

3.2.1. Observaciones sin Ruido

En este caso, al no haber un ruido gaussiano ϵ , entonces se tiene que $y_i = f_i$. Por lo tanto, el dataset podrá escribirse directamente como $\mathcal{D} = \{(x_i, f_i) \mid i = 1, \dots, n\}$. La distribución conjunta de los outputs de entrenamiento (f) y los outputs de test (f_*) será:

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right),$$

3. Procesos Gaussianos en Problemas de Regresión

donde, suponiendo que se tienen n_* puntos de test, entonces $K(X, X_*)$ es la matriz de dimensión $n \times n_*$ que representa las covarianzas de cada par de puntos de entrenamiento y test. De forma análoga, se definen las matrices $K(X, X)$, $K(X_*, X)$ y $K(X_*, X_*)$.

Veamos de nuevo la siguiente imagen, ya estudiada en un ejemplo del apartado 2.2. Una vez se tiene una distribución a priori con media 0 y se procede a la observación de un par de puntos, se podría pensar en simplemente rechazar las que no pasan por estos puntos. Sin embargo, debido a la inmensa cantidad de funciones que habría, esto no es computacionalmente viable.

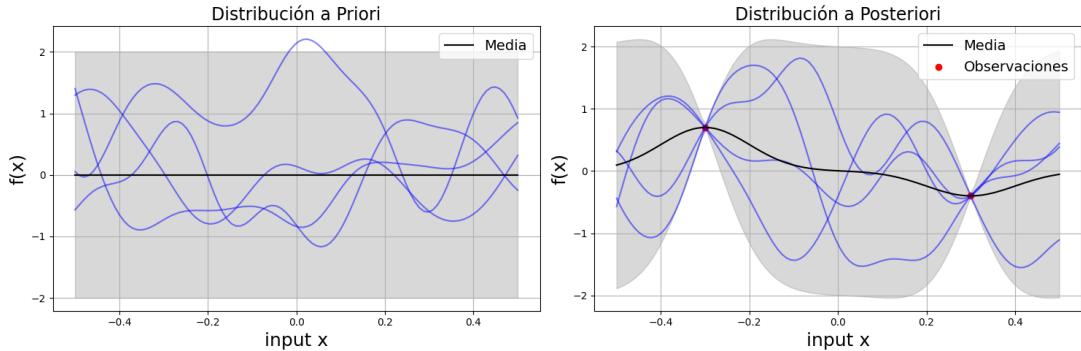


Figura 3.4.: Código de programación propio, disponible en este [enlace](#).

Es por ello que para hacer las predicciones se hace uso de la siguiente distribución condicionada:

$$f_* | X_*, X, f \sim \mathcal{N}(K(X_*, X)K(X, X)^{-1}f, K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)). \quad (3.8)$$

En la Figura 3.4 se hace uso de esta distribución, partiendo de 100 puntos iniciales que generan la distribución a priori y luego se calcula la distribución a posteriori tras observar dos nuevos puntos. Para funciones de más dimensiones sería análogo, simplemente sería más complicado de representar gráficamente.

3.2.2. Observaciones con Ruido

Es más realista pensar en situaciones en las que no disponemos de los valores f_i como tal, sino **aproximaciones ruidosas** de ellos: $y_i = f(x_i) + \epsilon$. En este caso, los Procesos Gaussianos intentan reconstruir la función f eliminando las contaminaciones ruidosas de ϵ . Igual que en la sección 3.1, contaremos con un conjunto de datos $\mathcal{D} = \{(x_i, y_i) | i = 1, \dots, n\}$ y tomaremos ruidos gaussianos independiente e idénticamente distribuido con varianza σ_n^2 , es decir, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.

En este caso, la función de covarianza deberá añadir la varianza σ_n^2 del ruido, por lo que quedará como:

$$\text{cov}(y_p, y_q) = k(x_p, x_q) + \sigma_n^2 \delta_{pq},$$

donde δ_{pq} es la función ***delta de Kronecker***, que se define de la siguiente forma:

$$\delta_{pq} = \begin{cases} 1 & \text{si } p = q \\ 0 & \text{si } p \neq q \end{cases}$$

De forma alternativa, también podemos definir la **covarianza** como:

$$\text{cov}(y) = K(X, X) + \sigma_n^2 I.$$

La distribución conjunta de los datos de entrenamiento (y) y los outputs observados (f_*) se calculará de forma análoga al apartado 3.2.1, añadiendo simplemente la varianza del ruido:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N} \left(0, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right).$$

De igual forma, una vez se han observado nuevos datos que modifican la distribución a priori, se determina la distribución condicional $f_* | X, y, X_*$ para realizar las **predicciones** como en la expresión (3.8):

$$f_* | X, y, X_* \sim \mathcal{N}(\bar{f}_*, \text{cov}(f_*)), \quad (3.9)$$

donde

$$\bar{f}_* \triangleq \mathbb{E}[f_* | X, y, X_*] = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}y, \quad (3.10)$$

$$\text{cov}(f_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X_*). \quad (3.11)$$

Observación 3.1. El resultado también es análogo a la expresión (3.6) en el caso del espacio de características. Basta con tomar:

$$\begin{aligned} K(X, X) &= K, \\ K(X, X_*) &= P^T \Sigma_p \phi(x_*), \\ K(X_*, X) &= \phi(x_*)^T \Sigma_p P, \\ K(X_*, X_*) &= \phi(x_*)^T \Sigma_p \phi(x_*). \end{aligned}$$

Observación 3.2. La covarianza $\text{cov}(f_*)$ no depende de los *targets* observados, sino que solo de los *inputs*. Esto es una propiedad que proviene de ser una Distribución Normal.

Observación 3.3. $\text{cov}(f_*)$ es la diferencia entre dos términos: el primer término, $K(X_*, X_*)$, es simplemente la covarianza a priori en X_* . El segundo es un término corrector que depende de X_* y X y representa la información que las observaciones nos proporcionan sobre la función.

En el caso en el que haya un único punto de test x_* , entonces se nota el vector de covarianzas entre x_* y los n puntos de entrenamiento como k_* . En este caso, la distribución condicional f_* se reduce a:

$$\begin{aligned} \bar{f}_* &= k_*^T (K + \sigma_n^2 I)^{-1} y, \\ \mathcal{V}[f_*] &= k(x_*, x_*) - k_*^T (K + \sigma_n^2 I)^{-1} k_*. \end{aligned} \quad (3.12)$$

Como \bar{f}_* es una combinación lineal de las observaciones de entrenamiento y , en ocasiones

3. Procesos Gaussianos en Problemas de Regresión

es llamada '**predictor lineal**'. Se puede escribir de la siguiente forma, tomando α como $\alpha = (K + \sigma_n^2 I)^{-1}y$:

$$\bar{f}_*(x_*) = \sum_{i=1}^n \alpha_i k(x_i, x_*).$$

En cuanto a la **verosimilitud marginal**, en esta ocasión se pueden observar ligeros cambios respecto a la situación del Espacio de Parámetros, que se estudió en la expresión (3.3). La verosimilitud marginal es la marginación sobre f , que depende de X , cosa que no sucedía con los pesos w . Por lo tanto, ahora la verosimilitud marginal quedará como:

$$p(y | X) = \int p(y | f, X) p(f | X) df. \quad (3.13)$$

Proposición 3.1. *El logaritmo de la verosimilitud marginal, $\log p(y | X)$, se puede expresar como:*

$$\log p(y | X) = -\frac{1}{2}y^T(K + \sigma_n^2 I)^{-1}y - \frac{1}{2}\log|K + \sigma_n^2 I| - \frac{n}{2}\log(2\pi). \quad (3.14)$$

Demostración. La distribución a priori es la siguiente normal:

$$f | X \sim \mathcal{N}(0, K(X, X)),$$

o tomando logaritmo:

$$\log(p(f | X)) = -\frac{1}{2}f^T K^{-1}f - \frac{1}{2}\log|K| - \frac{n}{2}\log(2\pi). \quad (3.15)$$

Por otro lado, la verosimilitud sigue otra normal:

$$y | f \sim \mathcal{N}(f, \sigma_n^2 I).$$

Usando la Proposición 2.7, se obtiene la distribución de la verosimilitud marginal vista en la expresión (3.13):

$$Z^{-1} \mathcal{N}(f | (K^{-1} + (\sigma_n^2 I)^{-1})^{-1}(-y(\sigma_n^2 I)^{-1}), (K^{-1} + (\sigma_n^2 I)^{-1})^{-1}),$$

con

$$Z^{-1} = \frac{1}{(2\pi)^{-p/2}} |K + \sigma_n^2 I|^{-1/2} \exp\left(-\frac{1}{2}y^T(K + \sigma_n^2 I)^{-1}y\right).$$

Se puede factorizar de la integral Z^{-1} y el resultado quedará como $Z^{-1} \times 1$, ya que la integral sobre la normal obviamente será 1 por ser una distribución de probabilidad. Tomando logaritmos se obtiene el resultado buscado.

□

Observación 3.4. Realmente en (3.13) no tenemos dos distribuciones sobre f . Sin embargo, por la expresión de la función de densidad de la distribución normal multivariante, podemos considerar la función de densidad de $y | f \sim \mathcal{N}(f, \sigma_n^2 I)$ como una distribución sobre f con media $-y$: $\mathcal{N}(-y, \sigma_n^2 I)$.

Ahora vemos un implementación completa para Procesos Gaussianos en Regresión. Este algoritmo es en el que se basa Scikit-Learn para su clase *GaussianProcessRegressor*, que

se usará en el apartado 6.3 de la sección de experimentos. El algoritmo proviene, una vez más, del libro de Rasmussen y Williams [RW⁺06]. Como se comentó en la introducción del trabajo, este libro es la guía fundamental para el uso de Procesos Gaussianos en Aprendizaje Automático.

Algorithm 1 . Cálculo de predicciones y logaritmo de la verosimilitud marginal para regresión en Procesos Gaussianos.

Require: X (inputs), y (targets), k (función de covarianza), σ_n^2 (ruido), x_* (test input)

- 1: $K \leftarrow k(X, X)$
- 2: $L \leftarrow \text{cholesky}(K + \sigma_n^2 I)$
- 3: $\alpha \leftarrow L^\top \backslash (L \backslash y)$
- 4: $k_* \leftarrow k(X, x_*)$
- 5: $f_* \leftarrow k_*^\top \alpha$ media: eq. (3.10)
- 6: $v \leftarrow L \backslash k_*$
- 7: $V[f_*] \leftarrow k(x_*, x_*) - v^\top v$ varianza: eq. (3.11)
- 8: $\log p(y|X) \leftarrow -\frac{1}{2}y^\top \alpha - \sum_i \log L_{ii} - \frac{n}{2} \log 2\pi$ log. verosimilitud marginal: eq. (3.14)
- 9: **return** f_* (media), $V\{f_*\}$ (varianza), $\log p(y|X)$ (logaritmo de la verosimilitud marginal).

=0

El algoritmo utiliza la factorización de Cholesky, ya que mejora la complejidad en comparación a realizar la inversa de la matriz directamente. La complejidad del cálculo de la factorización de Cholesky, que se estudia en el apéndice A.1, es $\mathcal{O}(n^3/6)$. Por otro lado, el cálculo de los sistemas triangulares necesarios para determinar α y v son $\mathcal{O}(n^2/2)$.

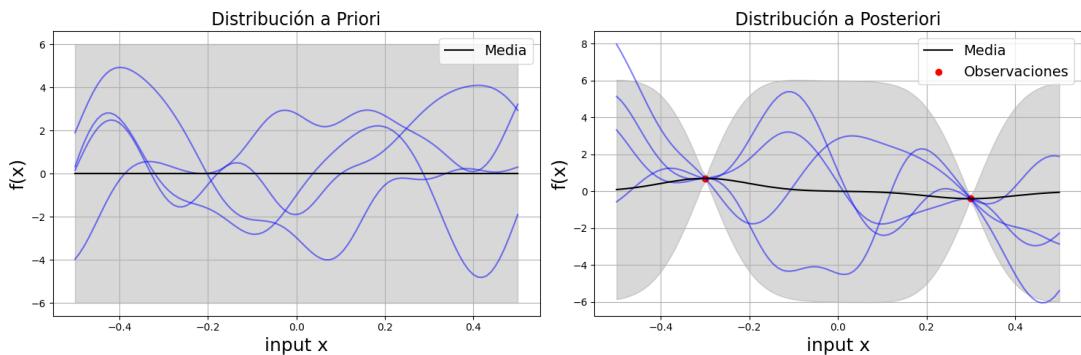
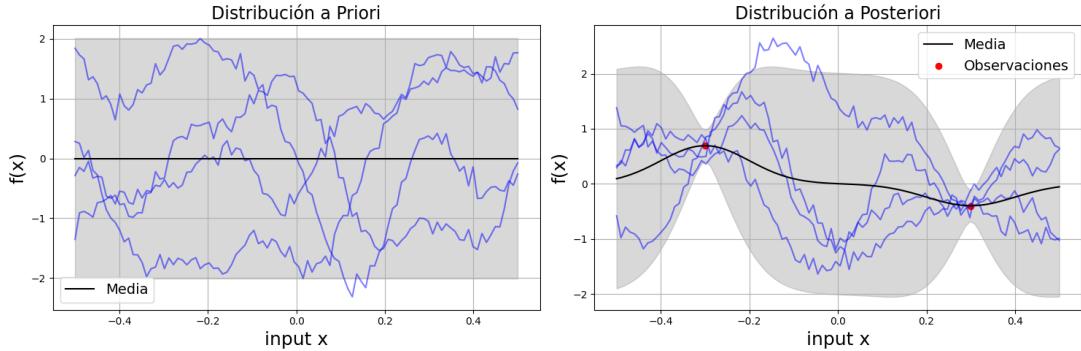
3.3. Hiperparámetros

Las funciones de covarianza en el ámbito de los Procesos Gaussianos pueden tener numerosos **hiperparámetros**, como se verá en profundidad en el capítulo 5. Estas variables son parámetros ajustables que controlan la forma y las características de las funciones de covarianza, siendo cruciales para que el Proceso Gaussiano sea efectivo en la predicción e interpolación de datos. Por ejemplo, hasta ahora hemos trabajado con la *Squared Exponential* vista en (3.7), donde se contaba con un único hiperparámetro, el **length-scale**, cuyos efectos se vieron claramente en la Figura 3.2. Sin embargo, se pueden utilizar más parámetros libres en la SE:

$$k_y(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2l^2} |x_p - x_q|^2\right) + \sigma_n^2 \delta_{pq}.$$

En esta nueva expresión se han añadido dos nuevos hiperparámetros: el *signal variance* (σ_f^2) y el *noise variance* (σ_n^2), que controlan la escala vertical y el ruido en las observaciones. Vemos la influencia de estos dos nuevos hiperparámetros en las Figuras 3.5 y 3.6.

3. Procesos Gaussianos en Problemas de Regresión



3.4. Teoría de Decisión

En el apartado 2.3 se definió la **función de pérdida o riesgo** en el contexto del Aprendizaje Automático y se vio que, durante el entrenamiento, se busca minimizar esta función para obtener el modelo. En nuestro caso, el objetivo es realizar una predicción y_{guess} y calcular la pérdida respecto al valor real y_{true} . Nos enfocamos en minimizar una '**pérdida esperada**',

$\tilde{R}_{\mathcal{L}}$, utilizando la verosimilitud de nuestros datos observados (x_*, y_*) :

$$\tilde{R}_{\mathcal{L}}(y_{guess} | x_*) = \int \mathcal{L}(y_*, y_{guess}) p(y_* | x_*, \mathcal{D}) dy_*.$$

Nuestro modelo buscará minimizar esta pérdida esperada, es decir:

$$y_{optimal} | x_* = \operatorname{argmin}_{y_{guess}} \tilde{R}_{\mathcal{L}}(y_{guess} | x_*).$$

En general, si se utiliza como función de pérdida el error absoluto, el valor buscado será la **mediana** de $p(y_* | x_*, \mathcal{D})$. Si se utilizase un error cuadrático, sería su **media**. Como hemos visto, nuestra distribución de predicción será una normal, por lo que ambos valores coinciden. Así que en el caso de utilizar cualquiera de estas dos funciones de pérdida, la predicción que da el modelo es la media de la distribución a posteriori.

Durante los desarrollos del Capítulo 3, se tiene una búsqueda cerrada, en la que simplemente calculamos la distribución a posteriori que nos dará el resultado buscado. Si hacemos una reflexión minuciosa, en el entrenamiento lo único que se ha hecho es calcular una matriz inversa: Viendo las expresiones (3.10) y (3.11), en la media y covarianza de la distribución a posteriori lo único que no depende de los puntos de 'test' es $[K(X, X) + \sigma_n^2 I]^{-1}$.

En el Capítulo 5 se verán modelos basados en Procesos Gaussianos más sofisticados, en los que durante el entrenamiento también se realizarán una búsqueda de los mejores hiperparámetros.

4. Procesos Gaussianos en Problemas de Clasificación

En este Capítulo pasaremos de los problemas de regresión vistos en el Capítulo 3 a los **problemas de clasificación**. Este tipo de problemas en Aprendizaje Automático Supervisado fueron introducidos en el apartado 2.3: Tratan de asignar a cada input x una de las diferentes clases que tendrá el problema, C_1, C_2, \dots, C_C . Dependiendo del número de clases, distinguiremos problemas **binarios** ($C = 2$) y **multiclas** ($C > 2$), siendo los binarios los problemas que se tratarán más a fondo en este trabajo. En el apartado 4.1 se estudiarán los clasificadores probabilísticos y sus enfoques, generativos o discriminativos. En el apartado 4.2 se introducirá un modelo básico lineal de clasificación para posteriormente en la sección 4.3 tratar los Procesos Gaussianos en problemas de clasificación (GPC). En el apartado 4.4 se estudiará la aproximación de Laplace, elemento clave para la implementación que se dará del algoritmo GPC.

4.1. Clasificadores Probabilísticos

En este capítulo, se trabajará con Procesos Gaussianos para generar **clasificadores probabilísticos**. En el contexto del Aprendizaje Automático, estos clasificadores son aquellos que dan las predicciones en forma de **probabilidades**. Es decir, dada una observación de entrada, el clasificador genera una distribución de probabilidad sobre un conjunto de clases, en lugar de dar simplemente una clase como salida, que sería la forma de proceder de un clasificador puntual. Claramente, se podría transformar un clasificador probabilístico en uno puntual de una forma muy sencilla, haciendo que el modelo simplemente devuelva una única clase como predicción: la más probable. En situaciones en las que esta $\max_i p(C_i | x)$ sea relativamente baja (supongamos por ejemplo que no supera un umbral θ), se esperaría que el modelo pudiese cometer error en la predicción. En estos casos se puede añadir una opción de rechazo en el clasificador, dejando la decisión para un sistema más sofisticado. En problemas multiclas también se puede exigir una distancia mínima entre la mayor $p(C_i | x)$ y la segunda mayor.

Tanto la regresión como la clasificación se pueden ver como problemas de aproximación de funciones. Sin embargo, ahora tenemos una diferencia con regresión que nos complicará la forma de proceder: la **verosimilitud no puede ser Gaussiana**, por lo que el proceso del cálculo va a ser más costoso. En el capítulo anterior se contaba con una distribución a priori gaussiana y una función de verosimilitud gaussiana, lo que nos daba una distribución a posteriori gaussiana. Ahora, en problemas de clasificación se tienen *targets* discretos, por lo que una verosimilitud gaussiana no tiene sentido. En este capítulo se estudiarán formas de hacer aproximaciones cuando no es factible realizar inferencia exacta.

En cuanto a la teoría de decisión, notaremos como $\mathcal{L}(c, c')$ la **pérdida** al decidir que la clase es $C_{c'}$ cuando realmente era C_c . Ante esta decisión se define como $R_{\mathcal{L}}(c' | x) = \sum_c \mathcal{L}(c, c') p(C_c | x)$ la **pérdida esperada**, también conocida como **riesgo**. Por tanto, la decisión óptima será la que minimice el riesgo $R_{\mathcal{L}}(c' | x)$.

4. Procesos Gaussianos en Problemas de Clasificación

Una de las funciones de pérdida más comunes es la **pérdida cero-uno**, que se define de la siguiente forma:

$$\mathcal{L}(y, \hat{y}) = \begin{cases} 0 & \text{si } y = \hat{y} \\ 1 & \text{si } y \neq \hat{y} \end{cases}$$

El clasificador óptimo según esta función de pérdida es llamado **clasificador Bayes**. Sin embargo, no siempre es adecuado, ya que por ejemplo en medicina un falso positivo no es igual de perjudicial que un falso negativo.

En problemas de clasificación, las **regiones de decisión** son subconjuntos del espacio de características en los que todas las observaciones son asignadas a la misma clase. Es decir, una región de decisión para una clase particular C_k está formado por todos los puntos tales que la decisión del clasificador les asigna C_k . Las zonas en las que el clasificador cambia su predicción de una clase a otra son llamadas **fronteras de decisión**. Véase la Figura 4.1.

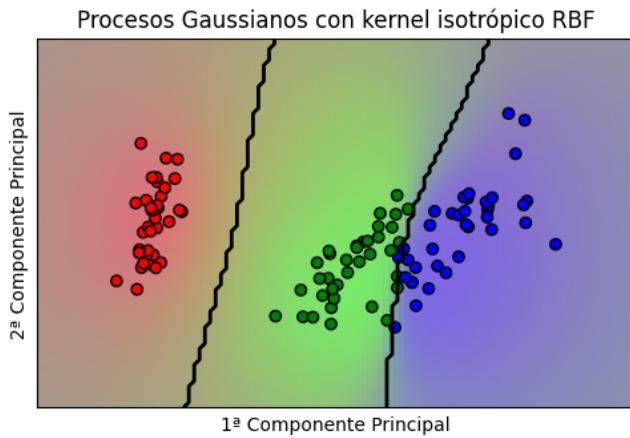


Figura 4.1.: Ejemplo de regiones de decisión y fronteras de decisión sobre el problema de clasificación de flores iris que se estudiará en el apartado 6.2.2. Código de programación propio, disponible en este [enlace](#).

Los clasificadores probabilísticos se pueden ver como una tarea de dos pasos: primero calcular la distribución a posteriori y luego combinarla con la función de pérdida para realizar la decisión. Sin embargo hay autores, como por ejemplo Vapnik [Vap13], que comentan que si el objetivo es realizar la predicción entonces directamente se debería aproximar la función que minimice el riesgo, es decir,

$$R_{\mathcal{L}}(c) = \int \mathcal{L}(y, c(x)) p(y, x) dy dx,$$

donde $p(y, x)$ es la distribución de probabilidad conjunta de *targets* e *inputs* y $c(x)$ es una función de clasificación que asigna a un *input* x una clase. Como $p(y, x)$ es desconocida, a menudo en este enfoque se busca minimizar una función objetivo que incluye el riesgo empírico, $\sum_{i=1}^n \mathcal{L}(y_i, c(x_i))$, y un factor de regularización.

Aunque este es un método razonable, el enfoque probabilístico permite reutilizar la misma

etapa de inferencia con diferentes funciones de pérdida, lo cual puede ayudar a incorporar conocimientos previos sobre la función y el modelo de ruido. Además, tiene la ventaja de proporcionar predicciones probabilísticas, lo cual puede ser útil, por ejemplo, para la opción de rechazo.

Del Teorema de Bayes (2.2) se obtiene fácilmente que la expresión de la probabilidad de la intersección de dos sucesos A y B ($P(A, B)$) se puede expresar de dos formas:

$$P(B)P(A | B) = P(A, B) = P(A)P(B | A).$$

Tomando la etiqueta de una clase, y , y el *input* x , se obtiene que la **probabilidad intersección**, $p(y, x)$, es:

$$p(x)p(y | x) = p(y, x) = p(y)p(x | y). \quad (4.1)$$

Distinguiremos dos enfoques, generativo o discriminativo, dependiendo de como descompongamos la distribución conjunta $p(y, x)$.

4.1.1. Enfoque Generativo

En primer lugar, estudiaremos el enfoque generativo. Un ejemplo muy conocido de este tipo de clasificadores es Naive Bayes [R+01]. Este enfoque **modela las distribuciones condicionales** $p(x | y)$ para cada clase $y = C_1, \dots, C_C$, así como las **probabilidades a priori** de cada clase. Posteriormente, calcula la distribución a posteriori de cada clase con la expresión

$$p(y | x) = \frac{p(y)p(x | y)}{\sum_{c=1}^C p(C_c)p(x | C_c)}.$$

Esta expresión se obtiene de despejar $p(y | x)$ en la igualdad (4.1) para posteriormente utilizar el **Teorema de la Probabilidad Total** sobre $p(x)$. Se recuerda a continuación este Teorema:

Teorema 4.1. *Sea $\{A_n\}_{n \in \mathbb{N}} \subset \mathcal{A}$ una secuencia de sucesos que define una partición del espacio muestral Ω , siendo $P(A_i) > 0$, para $i \in \mathbb{N}$, entonces*

$$P(B) = \sum_{i=1}^n P(B | A_i)P(A_i).$$

Gracias a tener la probabilidad $p(x)$ a través de la expresión $p(x) = \sum_y p(y)p(x | y)$, este enfoque puede ser útil para tratar problemas relacionados con el *input* x , como podrían ser valores faltantes, valores atípicos y datos sin etiquetar.

Para esta situación, calcular las distribuciones condicionales de x puede ser un gran problema, especialmente cuando x tenga una dimensión elevada. Se podría pensar en tomar las distribuciones $p(x | C_c)$ como normales $\mathcal{N}(\mu_c, \Sigma_c)$. Sin embargo, esto es una suposición muy fuerte sobre estas densidades condicionales, por lo que podría generar modelos que den lugar a unos malos resultados.

Por lo tanto, podríamos pensar que se está resolviendo un problema que podría resolverse de una forma más sencilla, como se verá a continuación.

4.1.2. Enfoque Discriminativo

El enfoque discriminativo, que será el utilizado por los Procesos Gaussianos, es la gran alternativa a lo visto en el apartado 4.1.1. En este caso se **modela la distribución a posteriori** $p(y | x)$ **directamente**.

Además de que modela directamente lo que buscamos, $p(y | x)$, el enfoque discriminativo tiene la ventaja de que tendrá menos parámetros que determinar. Se centra en estimar las fronteras de clasificación, en lugar de estimar directamente las densidades [VC⁺17].

Una idea simple para **problemas de clasificación binarios** podría ser transformar la salida de un modelo de regresión a una clase de probabilidad, usando una **función de respuesta** (la inversa de una **función de enlace**), que ‘comprime’ su argumento al intervalo $[0, 1]$, dándole así una interpretación probabilística. Un ejemplo podría ser el siguiente modelo de regresión logístico lineal:

$$p(C_1 | x) = \lambda(x^T w) \quad (4.2)$$

donde $\lambda(z) = \frac{1}{1+exp(-z)}$ recibe el nombre de **función logística**, que es una función sigmoide con la característica forma de ‘S’:

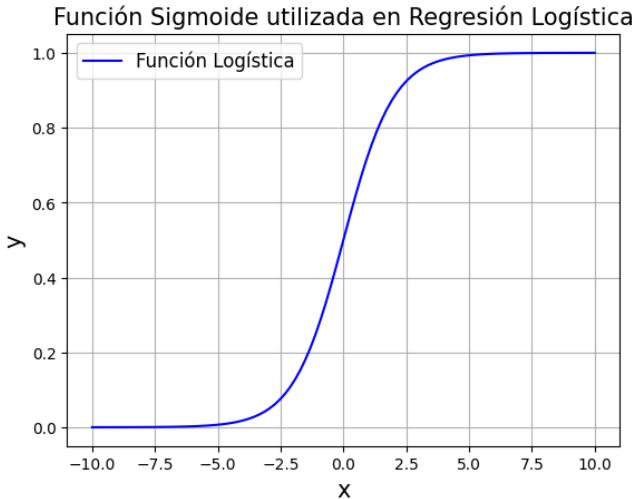


Figura 4.2.: La función logística es una función simétrica que transforma cualquier valor real z al rango $(0, 1)$, dando una interpretación probabilística al modelo de regresión lineal estudiado en el apartado 3.1. Código de programación propio, disponible en este [enlace](#).

Otra opción de función de respuesta es la conocida como **regresión probit**, que básicamente es la función de distribución de una normal estándar:

$$\Phi(z) = \int_{-\infty}^z \mathcal{N}(x | 0, 1) dx. \quad (4.3)$$

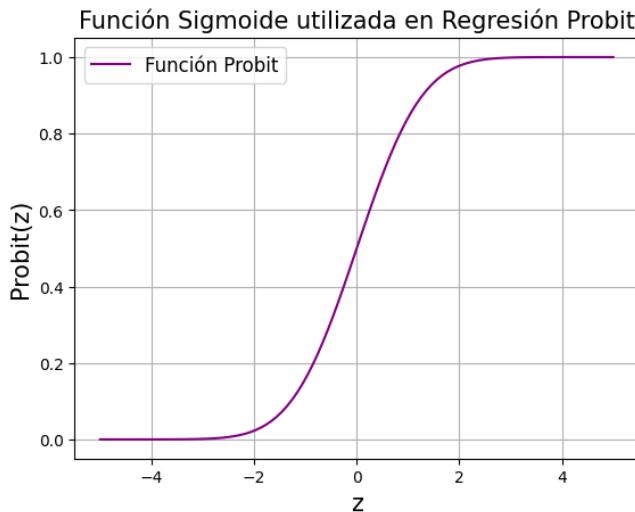


Figura 4.3.: Función probit, muy similar a la función logística. Código de programación propio, disponible en este [enlace](#).

Ambas funciones son muy similares y tienden a producir resultados similares. Si juntamos las dos en un mismo gráfico veremos que hay pequeñas diferencias: la función logística es ligeramente más suave, con menos pendiente.

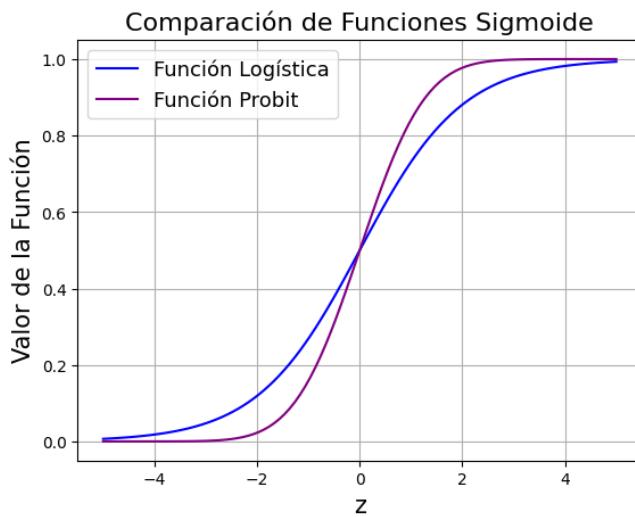


Figura 4.4.: Comparación de ambas funciones. Código de programación propio, disponible en este [enlace](#).

4.2. Modelos Lineales de Clasificación

De forma análoga a lo que se hizo en el Capítulo 3 con el Modelo Lineal Bayesiano para introducir los Procesos Gaussianos en problemas de regresión, en este apartado se tratarán

4. Procesos Gaussianos en Problemas de Clasificación

modelos lineales de **clasificación binaria** como la Regresión Logística y la Regresión Probit para posteriormente, en el apartado 4.3, introducir los clasificadores basados en Procesos Gaussianos.

Las dos clases existentes se notarán con las etiquetas $y = +1$ e $y = -1$. Dados el vector de pesos w y una función sigmoide $\sigma(x)$, se define la **función de verosimilitud** como $p(y = +1 | x, w) = \sigma(x^T w)$. Al ser probabilidades, se puede utilizar el complementario para expresar la otra clase, es decir: $p(y = -1 | x, w) = 1 - p(y = +1 | x, w) = 1 - \sigma(x^T w)$.

Cuando la función sigmoide utilizada sea la función logística definida en la expresión (4.2), el modelo se conocerá como **Modelo Lineal de Regresión Logística**. Cuando se utilice la función dada en (4.3), el modelo se llamará **Modelo Lineal de Regresión Probit**.

Tanto el Modelo de Regresión Logística como el Modelo de Regresión Probit, al ser funciones simétricas como se puede ver en la Figura 4.4, cumplen que $\sigma(-z) = 1 - \sigma(z)$. Unido a que las dos posibles clases son $y = +1$ e $y = -1$, la función de verosimilitud se puede expresar de una forma más compacta de la siguiente forma:

$$p(y_i | x_i, w) = \sigma(y_i x_i^T w).$$

Definiendo la función de transformación **logit** como $\text{logit}(x) = \log(p(y = +1 | x)/p(y = -1 | x))$, entonces el Modelo de Regresión Logística puede escribirse como

$$\text{logit}(x) = x^T w.$$

Supongamos que se cuenta de nuevo con un conjunto de datos $\mathcal{D} = \{(x_i, y_i) | i = 1, \dots, n\}$. Entonces, usando la distribución Gaussiana a priori del capítulo anterior, $w \sim \mathcal{N}(0, \Sigma_p)$, se obtiene el siguiente logaritmo de la distribución a posteriori (no normalizado):

$$\log p(w | X, y) \stackrel{c}{=} -\frac{1}{2}w^T \Sigma_p^{-1}w + \sum_{i=1}^n \log \sigma(y_i x_i^T w), \quad (4.4)$$

teniendo en cuenta que la distribución a posteriori en el modelo lineal bayesiano del apartado 3.1 seguía la distribución dada en la expresión (3.4). Como ya se comentó anteriormente, para la clasificación se tenía el problema de no se puede calcular de una forma cerrada la distribución a posteriori. Sin embargo, podemos usar las funciones sigmoides ya presentadas para facilitar los cálculos: se cumple que el logaritmo de la verosimilitud es una función **cóncava** de w fijado un conjunto de datos \mathcal{D} . Además, la penalización cuadrática en los parámetros w también es una función cóncava, lo que implica que la función a posteriori logarítmica total es cóncava. Debido a ello, tiene un único máximo que es relativamente fácil de encontrar utilizando el método de Newton. En el contexto de la clasificación, este método se conoce como el algoritmo de mínimos cuadrados reponderados iterativamente (IRLS).

Para realizar una **predicción** en el punto x_* basada en el dataset \mathcal{D} , se tiene la siguiente distribución:

$$p(y_* = +1 | x_*, \mathcal{D}) = \int p(y_* = +1 | w, x_*) p(w | \mathcal{D}) dw \quad (4.5)$$

donde $p(y_* = +1 | w, x_*) = \sigma(x_*^T w)$.

El Modelo de Regresión Logística puede ser generalizado para soportar múltiples clases directamente, evitando así la necesidad de entrenar y combinar varios clasificadores binarios. Esto es comúnmente denominado **Regresión Softmax** o **Regresión Logística Multinomial**. La idea es sencilla: dada una instancia x y un vector w_c de pesos para cada clase c , el modelo calcula primero una 'puntuación' para cada c con la fórmula $x^T w_c$ y posteriormente estima la probabilidad $p(y = C_c | x, W)$ de que la instancia pertenezca a la clase c aplicando la Función Softmax:

$$p(y = C_c | x, W) = \frac{e^{x^T w_c}}{\sum_{c'} e^{x^T w_{c'}}}.$$

La función calcula el exponencial de cada puntuación y luego las normaliza (dividiendo por la suma de todos los exponentiales). Las puntuaciones suelen denominarse *logits* o *log-odds*.

Ejemplo 4.1. Veamos ahora un ejemplo de un Modelo de Regresión Logística Lineal en un problema de clasificación binaria, de forma análoga a lo realizado para regresión en el Ejemplo 3.1 del Capítulo 3. Este ejemplo está basado en la Figura 3.1 del libro de Rasmussen y Williams [RW+06]. Se parte de una **distribución a priori normal** $p(w) = \mathcal{N}(0, I)$ y de 6 puntos de **entrenamiento**, en la Figura 4.5 marcados como círculos si son de clase +1 y como cruces si son de clase -1.

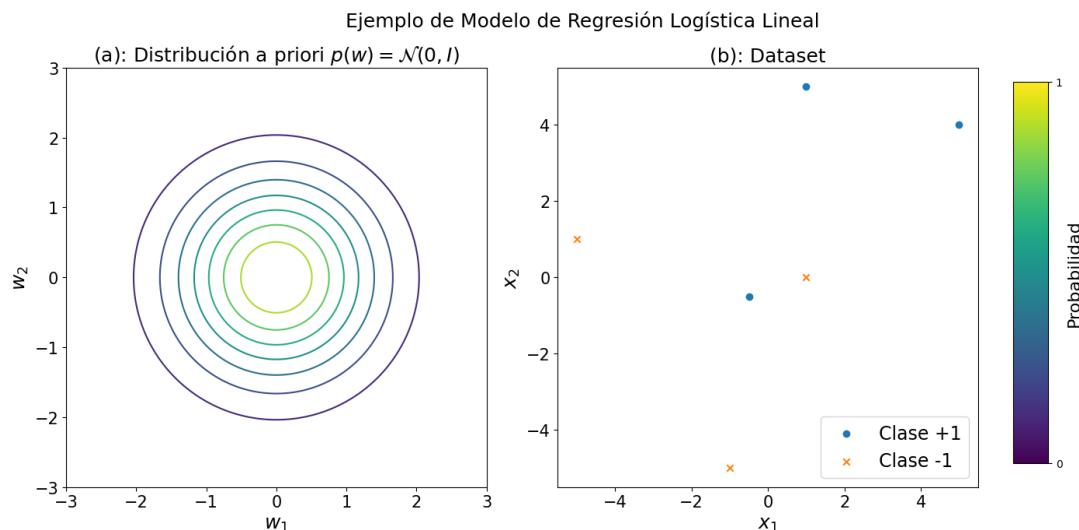


Figura 4.5.: Datos iniciales del Ejemplo 4.1. Los colores más oscuros indican zonas de valores de w_1 y w_2 menos consistentes con los datos observados (menos probabilidad), mientras que los colores más claros indican valores más probables. Código de programación propio, disponible en este [enlace](#).

A partir de esta distribución a priori y estos datos de entrenamiento se puede calcular la **distribución a posteriori**, $p(w | X, y)$, realizando la exponencial de la expresión (4.4). Esta distribución **no será normal**, como se ha comentado a lo largo del apartado y se puede ver en la Figura 4.6. Utilizando esta distribución a posteriori, a través de la expresión (4.5) se ha

4. Procesos Gaussianos en Problemas de Clasificación

calculado la **distribución predictiva** para la clase $y = +1$.

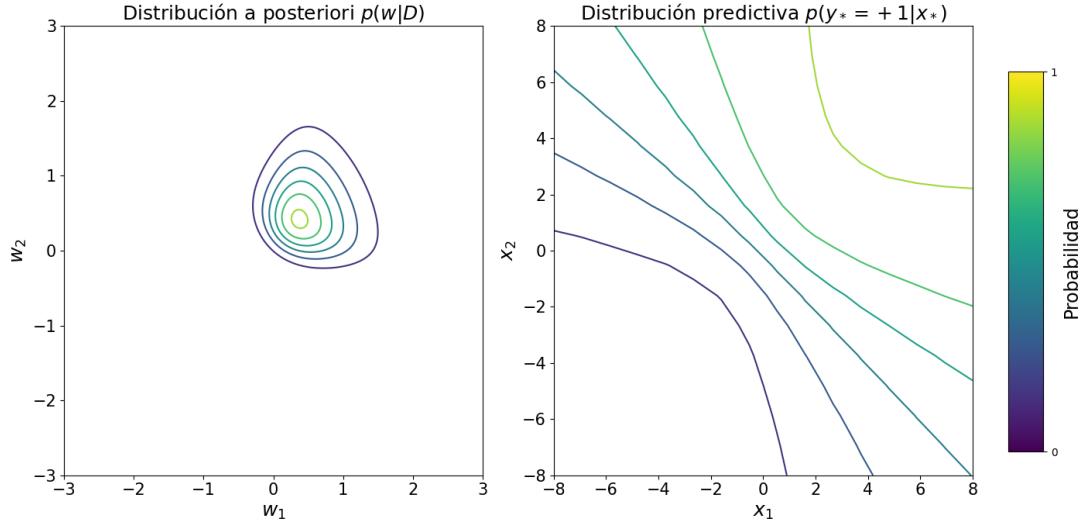


Figura 4.6.: Distribución a posteriori y distribución predictiva de la clase +1. Código de programación propio, disponible en este [enlace](#).

Como estamos en un problema binario, es obvio que la distribución predictiva para la clase $y = -1$ será el **complementario** de la imagen anterior. Se comprueba este hecho en la Figura 4.7, que se muestra a continuación:

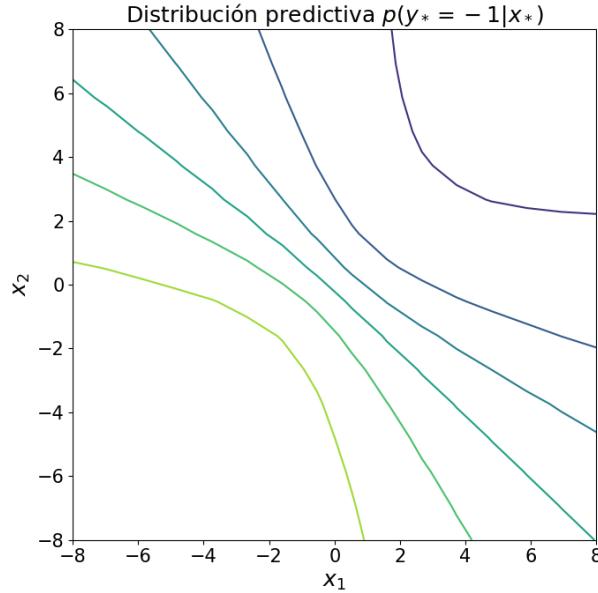


Figura 4.7.: Distribución predictiva de la clase -1. Código de programación propio, disponible en este [enlace](#).

4.3. Uso de Procesos Gaussianos

Para la predicción en problemas binarios de clasificación con Procesos Gaussianos, aplicamos lo ya estudiado en regresión durante el Capítulo 3. Se emplea una *función latente* $f(x)$ que se ‘comprime’ al intervalo $[0, 1]$ a través de la función logística, obteniendo así la función de probabilidad de la clase:

$$\pi(x) \triangleq p(y = +1|x) = \sigma(f(x)).$$

Es una generalización natural de lo ya visto anteriormente. En concreto, es simplemente reemplazar en la expresión (4.4) la función $x_i^T w$ del modelo logístico por un Proceso Gaussiano y la consideración *a priori* sobre w por una distribución *a priori* de Proceso Gaussiano. Cabe destacar que, debido a que la función latente f es una función estocástica, π también lo es. En el siguiente ejemplo se puede observar como se transforma una función latente $f(x)$ en $\pi(x)$:

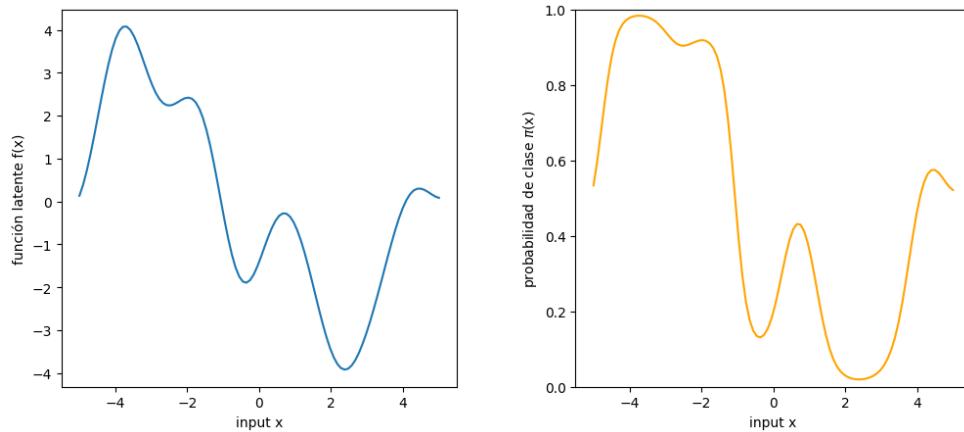


Figura 4.8.: En la primera imagen se muestra una función latente, $f(x)$. En la segunda imagen se muestra el resultado de la aplicación de la función logit $\sigma(x) = (1 + \exp(-x))^{-1}$ sobre f , obteniendo así la probabilidad de clase $\pi(x) = \sigma(f(x))$. Código de programación propio, disponible en este [enlace](#).

La función latente f obtenida del Proceso Gaussiano no es de especial interés, ya que lo realmente importante son los resultados de π , concretamente las predicciones $\pi(x_*)$. Cabe destacar también que en este apartado suponemos que el Proceso Gaussiano no tiene ruido.

El cálculo de la predicción probabilística deseada se separa en dos fases. Primero, se calcula la **distribución para la función latente** y, posteriormente, se usa esta distribución para calcular la predicción probabilística.

Sea la **distribución a posteriori** sobre la variable latente

$$p(f | X, y) = \frac{p(y | f)p(f | X)}{p(y | X)}, \quad (4.6)$$

4. Procesos Gaussianos en Problemas de Clasificación

se define la distribución de probabilidad en el caso de test para la variable latente como:

$$p(f_* | X, y, x_*) = \int p(f_* | X, x_*, f) p(f | X, y) df. \quad (4.7)$$

Con esta distribución $p(f_* | X, y, x_*)$, se determina la **distribución de la predicción probabilística** sobre la clase $y = +1$ como:

$$\pi_* \triangleq p(y_* = +1 | X, y, x_*) = \int \sigma(f_*) p(f_* | X, y, x_*) df_*. \quad (4.8)$$

En el Capítulo 3 de regresión, como se tenía una verosimilitud gaussiana el cálculo de las predicciones era bastante **directo** debido a que las integrales relevantes tenían distribuciones gaussianas y podían ser evaluadas analíticamente. Sin embargo, en el caso de los problemas de clasificación, la verosimilitud no gaussiana presentada en la ecuación (4.6) complica el proceso, ya que la integral (4.7) no puede ser tratada de manera analítica. De forma similar, la ecuación (4.8) puede presentar desafíos analíticos, especialmente para ciertas funciones sigmoides. Aunque, en el caso binario, esta ecuación se reduce a una integral unidimensional, lo que facilita el uso de técnicas numéricas simples para su evaluación.

Por tanto, es necesario recurrir a aproximaciones analíticas de integrales o buscar soluciones mediante el método de Montecarlo. En la próxima sección, se examinará la **Aproximación de Laplace**. Además, hay otras técnicas disponibles, como la '*Expectation Propagation*' (EP), que también aproximan la distribución a posteriori no gaussiana con una que sí lo es.

4.4. Aproximación de Laplace

En este apartado se va a estudiar la **Aproximación de Laplace**, centrándonos en el caso de **clasificación binaria**. Este método genera una estimación $q(f | X, y)$ para la distribución a posteriori $p(f | X, y)$ que se detalla en la igualdad (4.6) y es utilizada en la expresión (4.7). Este proceso se lleva a cabo realizando el siguiente razonamiento: **no tenemos una distribución normal, pero queremos aproximarla**, por lo que como **media** cogeremos su valor máximo. En cuanto a la **varianza**, si miramos la función de densidad de la MND en la expresión (2.1), aplicando logaritmos se cumple que el hessiano será la inversa de la matriz de covarianza con signo cambiado. La aproximación toma la siguiente expresión:

$$q(f | X, y) = \mathcal{N}(f | \hat{f}, A^{-1}) \propto \exp\left(-\frac{1}{2}(f - \hat{f})^T A (f - \hat{f})\right)$$

donde

$$\begin{aligned} \hat{f} &= \operatorname{argmax}_f p(f | X, y), \\ A &= -\nabla \nabla \log p(f | X, y) |_{f=\hat{f}}. \end{aligned}$$

Primero, se estudia el cálculo de \hat{f} : Observando la expresión (4.6) de $p(f | X, y)$, como $p(y | X)$ es independiente de f (de forma análoga a lo que sucedía en regresión), entonces será constante y para maximizar $p(f | X, y)$ respecto de f bastará maximizar el numerador de la igualdad (4.6). Es decir, la distribución a posteriori no normalizada, que es el siguiente

producto:

$$p(y | f)p(f | X). \quad (4.9)$$

Tomando logaritmos sobre el producto (4.9) y utilizando la expresión dada para $\log(f | X)$ en la igualdad (3.15), se obtiene:

$$\begin{aligned} \psi(f) &\triangleq \log(y | f) + \log(f | X) \\ &= \log(y | f) - \frac{1}{2}f^T K^{-1}f - \frac{1}{2}\log|K| - \frac{n}{2}\log(2\pi). \end{aligned}$$

Y realizando 2 veces la diferencial sobre f se obtiene:

$$\begin{aligned} \nabla \psi(f) &= \nabla \log p(y | f) - K^{-1}f \\ \nabla \nabla \psi(f) &= \nabla \nabla \log p(y | f) - K^{-1} = -W - K^{-1} \end{aligned}$$

con $W \triangleq -\nabla \nabla \log p(y | f)$ matriz diagonal, ya que la distribución para y_i depende solo de f_i , no de $f_{j \neq i}$.

Como la función $\psi(f)$ es cóncava, entonces \hat{f} será su **máximo**. Por lo tanto, para determinar \hat{f} igualamos a 0 la diferencial:

$$\nabla \psi(f) = 0 \implies \hat{f} = K \left(\nabla \log p(y | \hat{f}) \right). \quad (4.10)$$

Esta ecuación no puede resolverse directamente al no ser lineal $\nabla \log p(y | \hat{f})$, pero utilizando el método de Newton se obtiene la siguiente expresión:

$$f^{new} = f - (\nabla \nabla \psi)^{-1} \nabla \psi = (K^{-1} + W)^{-1} (Wf + \nabla \log p(y | f)).$$

Finalmente, podemos definir la **Aproximación de Laplace de la distribución gaussiana a posteriori** como:

$$q(f | X, y) = \mathcal{N}(\hat{f}, (K^{-1} + W)^{-1}).$$

Una vez se tiene la distribución a posteriori aproximada, se puede buscar la **distribución predictiva**, $q(f_* | X, y, x_*)$, para un nuevo input x_* . Para determinar la **media** de esta distribución se combinan las expresiones (3.12) y (4.10):

$$\mathbb{E}_q[f_* | X, y, x_*] = k(x_*)^T K^{-1} \hat{f} = k(x_*)^T \nabla \log p(y | \hat{f}). \quad (4.11)$$

El valor real sin aproximar $\mathbb{E}_p[f_* | X, y, x_*]$, dado por Opper y Winther en [OWoo], viene dado por la siguiente expresión:

$$\begin{aligned} \mathbb{E}_p[f_* | X, y, x_*] &= \int \mathbb{E}[f_* | f, X, x_*] p(f | X, y) df = \int k(x_*)^T K^{-1} f p(f | X, y) df = \\ &= k(x_*)^T K^{-1} \mathbb{E}[f | X, y]. \end{aligned} \quad (4.12)$$

Si comparamos las dos expresiones, (4.11) y (4.12), vemos que la diferencia entre ambas es que la media de la distribución a posteriori $p(f | X, y)$ que generaba los problemas, $\mathbb{E}[f | X, y]$, se aproxima por \hat{f} .

En cuanto a la **varianza** de la distribución predictiva, $q(f_* | X, y, x_*)$, se calculará a través

4. Procesos Gaussianos en Problemas de Clasificación

de la siguiente expresión:

$$\begin{aligned}\mathcal{V}_q[f_* | X, y, x_*] &= \mathbb{E}_{(p(f_* | X, x_*, f))} [f_* - \mathbb{E}[f_* | X, x_*, f])^2] \\ &\quad + \mathbb{E}_{q(f | X, y)} [(\mathbb{E}[f_* | X, x_*, f] - \mathbb{E}[f_* | X, y, x_*])^2].\end{aligned}$$

El primer sumando es la varianza de f condicionada a un valor de f , X y obviamente x_* , que ya se calculó en la expresión (3.11). El segundo sumando se debe al hecho de que $\mathbb{E}[f_* | X, x_*, f] = k(x_*)^\top K^{-1}f$ depende de f y, por lo tanto, hay un término adicional de $k(x_*)^\top K^{-1}\text{cov}(f | X, y)K^{-1}k(x_*)$. Bajo la aproximación gaussiana, $\text{cov}(f | X, y) = (K^{-1} + W)^{-1}$. Por lo tanto, la varianza será:

$$\mathcal{V}_q[f_* | X, y, x_*] = k(x_*, x_*) - k_*^T K^{-1} k_* + k_*^T (K^{-1} + W)^{-1} K^{-1} k_*.$$

Y usando el **Lema de Identidad Matricial de Woodbury** [TFPV92] visto en el Lema 3.1, la varianza quedará como:

$$\mathcal{V}_q[f_* | X, y, x_*] = k(x_*, x_*) - k_*^T (K + W)^{-1} k_*. \quad (4.13)$$

Una vez se tienen la media y la covarianza de f_* , se pueden hacer predicciones con la aproximación de Laplace. Sea $q(f_* | X, y, x_*)$ la distribución con medias y covarianzas dadas por las expresiones (4.11) y (4.13), se tiene:

$$\bar{\pi}_* \simeq \mathbb{E}_q[\pi_* | X, y, x_*] = \int \sigma(f_*) q(f_* | X, y, x_*) df_*. \quad (4.14)$$

Si únicamente se quiere dar la clase más probable, entonces no haría falta calcular la expresión anterior, sino únicamente \hat{f} . Si se quiere dar una probabilidad exacta para cada clase, entonces sí habría que calcular la integral dada en la expresión (4.14).

A continuación se dan los algoritmos para los dos casos comentados. Cabe destacar que la clase *GaussianProcessClassifier* de Scikit-Learn que será utilizada en los experimentos del Capítulo 6 utiliza internamente esta aproximación de Laplace.

Algorithm 2 . Caso 1: Solo se busca la clase con mayor probabilidad [RW+06]

Require: K (matriz de covarianza), y (± 1 targets), $p(y|f)$ (función de verosimilitud)

1: $f := 0$	inicialización
1: repeat	iteraciones algoritmo de Newton
2: $W := -\nabla \nabla \log p(y f)$	
3: $L := \text{cholesky}(I + W^{1/2} K W^{1/2})$	
4: $b := Wf + \nabla \log p(y f)$	
5: $a := b - W^{1/2} L^\top \backslash (L \backslash (W^{1/2} Kb))$	
6: $f := Ka$	
6: until convergencia: $-\frac{1}{2}a^\top f + \log p(y f)$	
7: return $\hat{f} := f$ (moda distribución a posteriori).	=0

Algorithm 3 . Caso 2: Se buscan las probabilidades de las clases [RW⁺o6]

Require: \hat{f} (moda distribución a posteriori), X (inputs), y (targets ± 1), k (función de covarianza), $p(y|f)$ (función de verosimilitud), x_* (test input)

1: $W := -\nabla \nabla \log p(y \hat{f})$	
2: $L := \text{cholesky}(I + W^{1/2} K W^{1/2})$	
3: $\bar{f}_* := k(x_*)^\top \nabla \log p(y \hat{f})$	eq. 4.11
4: $v := L \backslash (W^{1/2} k(x_*))$	
5: $V[f_*] := k(x_*, x_*) - v^\top v$	eq. 4.13
6: $\pi_* := \int \sigma(z) \mathcal{N}(z \bar{f}_*, V[f_*]) dz$	eq. 4.14
7: return π_* (función predictiva probabilística para la clase 1).	$=0$

5. Función de Covarianza y Estimación de Hiperparámetros

Una vez estudiados los problemas de regresión y clasificación en los Capítulos 3 y 4, ahora pasaremos a ver algunos aspectos importantes del uso de los Procesos Gaussianos. En primer lugar, en la sección 5.1 se estudiarán las funciones de covarianza, cuya importancia ya ha sido discutida en capítulos anteriores. En los apartados 5.1.1 y 5.1.2 se verán diferentes tipos de función de covarianza, mientras que en el apartado 5.1.3 se estudiará cómo crear nuevas funciones de covarianza partiendo de una ya existente. Posteriormente, en el apartado 5.2 se estudiará la estimación de hiperparámetros en los modelos de Procesos Gaussianos.

5.1. Función de Covarianza

En los modelos de Procesos Gaussianos, como hemos visto en los capítulos anteriores, la noción de similaridad entre inputs cercanos es muy importante, ya que es una suposición bastante obvia que dos puntos x e x' muy cercanos es probable que tengan valores objetivo y similares. De igual forma, los puntos de entrenamiento que están cerca de un punto de prueba deberían ser informativos sobre la predicción en ese punto. En los Procesos Gaussianos, las **Funciones de Covarianza** son las encargadas de definir esa 'cercanía' o 'similitud' entre puntos.

Estas funciones ya fueron definidas en la expresión (2.4) y son de vital importancia, ya que contienen las suposiciones sobre la función que se desea aprender. En general, cualquier función con un par de puntos como parámetros no será una función de covarianza válida. Vemos a continuación algunas de las condiciones que deben cumplir las funciones de covarianza.

Definición 5.1. Sea $X \subseteq \mathbb{R}^D$. Una **función kernel** es una función $k : X \times X \rightarrow \mathbb{R}$ tal que $\forall x, x' \in X$ se cumple que

$$k(x, x') = \langle \phi(x), \phi(x') \rangle,$$

donde ϕ es una aplicación de X a un espacio de características \mathcal{F} [STCo4].

Definición 5.2. Un kernel k se dirá **simétrico** si:

$$k(x, x') = k(x', x) \quad \forall x, x' \in X.$$

Definición 5.3. Un kernel k se dirá **semidefinido positivo** si, para todo conjunto de puntos $\{x_i \mid i = 1, \dots, n\}$, la matriz de Gram K formada por $K_{ij} = k(x_i, x_j)$ verifica que es semidefinida positiva, es decir,

$$Q(v) = v^T K v \geq 0 \quad \forall v \in \mathbb{R}^n. \tag{5.1}$$

5. Función de Covarianza y Estimación de Hiperparámetros

Además, si la matriz K es definida positiva (es decir, la expresión (5.1) cambiando el signo \geq por $>$), el kernel k también se dirá **definido positivo**.

En el contexto de los Procesos Gaussianos, se cumple que **las funciones de covarianza son kernels simétricos semidefinidos positivos**.

Observación 5.1. La función $Q(v)$ definida en la ecuación (5.1) recibe el nombre de **forma cuadrática**.

Observación 5.2. Cuando el kernel k sea una función de covarianza, la matriz K de la definición 5.3 se llamará **matriz de covarianza**, como ya se vió en el apartado 2.1.1.

Una función de covarianza se dirá que es **estacionaria** si es una función que depende de $x - x'$, es decir, que es invariante a traslaciones en el espacio de entrada. Además, se dirá que es una función de covarianza **isotrópica** si es una función de $|x - x'|$, esto es, si es invariante a movimientos rígidos. Por ejemplo, la *squared exponential* definida en la expresión (3.7) es una función de covarianza estacionaria e isotrópica.

Observación 5.3. En procesos estocásticos, un proceso que tiene una media constante y cuya función de covarianza es invariante ante traslaciones se llama **débilmente estacionario**. Un proceso f_X es estrictamente estacionario si todas sus distribuciones finito-dimensionales ($f_X = (f(x_1), \dots, f(x_D))$) son invariantes ante traslaciones.

Por otro lado, se tienen los kernels **anisotrópicos**, que serán aquellos en los que las propiedades de simetría y escala varían dependiendo de la dirección. En el contexto del Aprendizaje Automático estos kernels serán útiles en estructuras de datos complejos, como se verá en el Capítulo 6.

Las funciones de covarianza que dependan de x y x' únicamente a través de $\langle x, x' \rangle$ se llamarán **funciones de covarianza de producto escalar**. Estas funciones de covarianza destacan por ser invariantes por giros respecto al origen, pero no por traslaciones. Un ejemplo es la siguiente función de covarianza

$$k(x, x') = \sigma_0^2 + \langle x, x' \rangle, \quad (5.2)$$

que se puede obtener a partir de la regresión lineal al poner distribuciones a priori $\mathcal{N}(0, 1)$ en los coeficientes de x_d , ($d = 1, \dots, D$) y una distribución a priori $\mathcal{N}(0, \sigma_0^2)$ en el sesgo (hecho detalladamente en la ecuación 2.15 del libro de Rasmussen y Williams [RW+06]).

5.1.1. Funciones de Covarianza Estacionarias

En esta sección consideraremos los kernels como funciones de X en \mathbb{C} , en lugar de \mathbb{R} . También, al igual que se ha hecho en situaciones anteriores, tomaremos Procesos Gaussianos con media 0 para simplificar los cálculos. Por tanto, la función de covarianza quedará definida como

$$k(x, x') = \mathbb{E}[f(x)f^*(x')],$$

donde f^* es la conjugación compleja.

Ahora definiremos la noción de continuidad en media cuadrática para procesos estocásticos, que será necesaria para el siguiente resultado.

Definición 5.4. Sea x_1, x_2, \dots una secuencia de puntos. Sea $x_* \in \mathbb{R}^D$ fijo tal que $|x_k - x_*| \rightarrow 0$ cuando $k \rightarrow \infty$. Entonces, se dice que un proceso $f(x)$ es **continuo en media cuadrada** en x_* si $\mathbb{E}[|f(x_k) - f(x_*)|^2] \rightarrow 0$ cuando $k \rightarrow \infty$.

A continuación se verá el **Teorema de Bochner**, que demostrará que una función de covarianza de una función estacionaria puede representarse como la transformada de Fourier de una medida finita positiva.

Observación 5.4. Una función estacionaria, como ya se ha comentado, es una función de $x - x'$. Tomaremos $\tau = x - x' \in \mathbb{R}^D$.

Teorema 5.1. *Sea una función $k : \mathbb{R}^D \rightarrow \mathbb{C}$. Entonces, k será función de covarianza de un proceso débilmente estacionario continuo en media cuadrática si y solo si puede representarse como*

$$k(\tau) = \int_{\mathbb{R}^D} e^{2\pi i \langle s, \tau \rangle} d\mu(s),$$

donde μ es una medida finita positiva.

La demostración del Teorema de Bochner puede encontrarse en [GS04].

Si μ tiene densidad $S(s)$, entonces S es conocida como la **densidad espectral** de k . En el caso de que la densidad espectral exista, entonces la función de covarianza y la densidad espectral son duales de Fourier entre sí, como se muestra en la expresión (5.3). Esto es conocido como el **Teorema de Wiener-Khintchine**.

$$k(\tau) = \int S(s) e^{2\pi i \langle s, \tau \rangle} ds, \quad S(s) = \int k(\tau) e^{-2\pi i \langle s, \tau \rangle} d\tau \quad (5.3)$$

5.1.1.1. Funciones de Covarianza Isotrópicas

Ahora, en este apartado se tratará con funciones de covarianza isotrópicas, que únicamente dependen de $|\tau| \triangleq r$. A continuación, se verán algunos ejemplos de funciones de covarianza isotrópicas dadas en forma normalizada, es decir, $k(0) = 1$.

- **Función de covarianza Squared Exponential (SE) o Radial Basis Function (RBF):** Es la función de covarianza más usada y conocida, ya ha sido utilizada durante el trabajo, especialmente en el Capítulo 3. Su fórmula es la siguiente:

$$k_{SE} = \exp\left(-\frac{r^2}{2l^2}\right).$$

Tiene un parámetro de escala llamado *length-scale*, cuya influencia ya fue estudiada en la Figura 3.3. Es infinitamente diferenciable y muy suave, lo que le confiere propiedades útiles en diversas aplicaciones. Sin embargo, algunos autores consideran que la

5. Función de Covarianza y Estimación de Hiperparámetros

suposición de suavidad del modelo es demasiado fuerte para procesos físicos, por lo que prefieren el kernel Matérn, que se tratará a continuación [Ste99].

Este kernel es *infinitamente divisible*, es decir, que cumple que $\forall t > 0$ $(k(r))^t$ es otro kernel RBF válido, únicamente varía el *length-scale*. Su densidad espectral es $S(s) = (2\pi l^2)^{D/2} \exp(-2\pi^2 l^2 s^2)$.

- **Función de Covarianza de la Clase de Matérn:** Lleva el nombre del estadístico sueco Bertil Matérn, que introdujo esta familia de funciones de covarianza en el contexto del análisis espacial y la teoría de procesos estocásticos [Mat60]. Se puede considerar como una generalización del SE que incorpora un nuevo parámetro ν para controlar la suavidad de las funciones generadas. Su expresión es la siguiente:

$$k_{\text{Matern}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{l}\right)^\nu K_\nu\left(\frac{\sqrt{2\nu}r}{l}\right),$$

donde $\nu, l > 0$, Γ es la función Gamma y K_ν es la función de Bessel modificada [Ste65]. Cuanto menor sea ν menos suave será la función resultante, como se puede ver en la Figura 5.1. Para comparar correctamente se fija *length-scale* = 1 en los 3 casos.

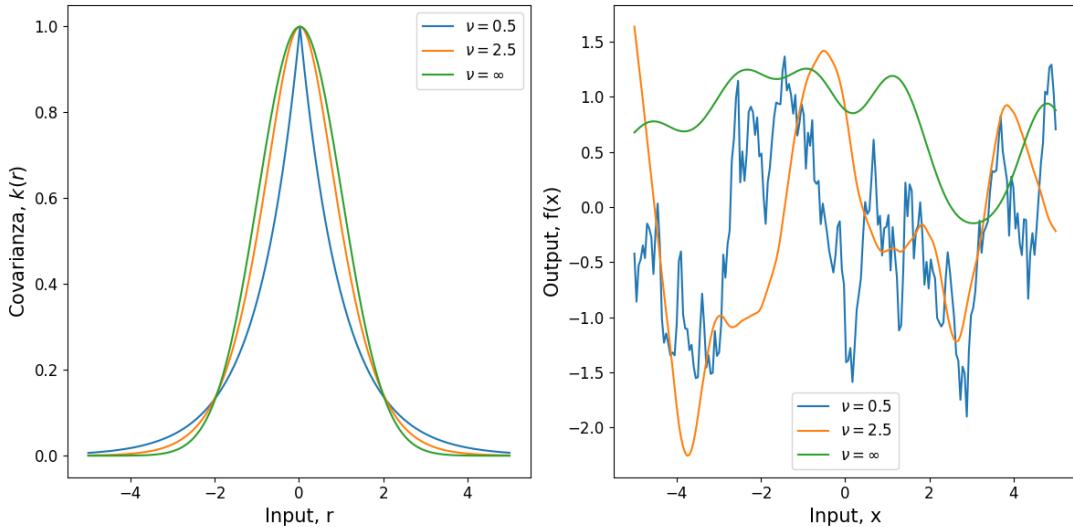


Figura 5.1.: Comparación de funciones de covarianza de clase Matérn para diferentes valores de ν . En la primera imagen se muestran las funciones de covarianza según los valores de ν , mientras que en la segunda se dibujan muestras aleatorias tomadas de la distribución a priori del Proceso Gaussiano generado con dichas funciones de covarianza. Código de programación propio, disponible en este [enlace](#).

Además, para valores $\nu \rightarrow \infty$ esta función de covarianza se approxima a la SE [Ste99]. Véase la Figura 5.2.

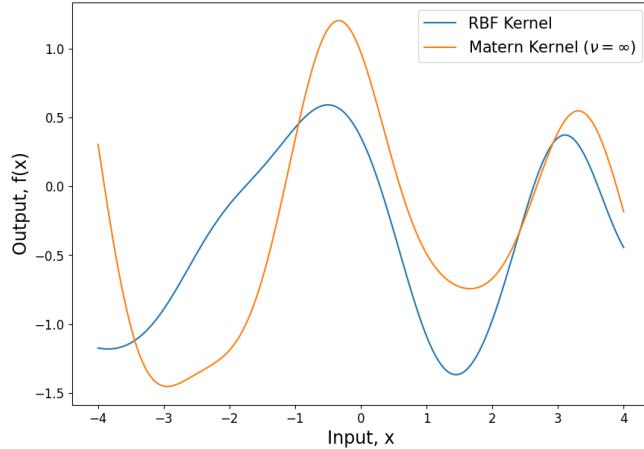


Figura 5.2.: Comparación del kernel RBF con el Matérn con $\nu \rightarrow \infty$. Código de programación propio, disponible en este [enlace](#).

La Función de Covarianza de Matérn es muy interesante para $\nu = n + \frac{1}{2}$ donde $n \in \mathbb{Z}$. En estos casos la función es producto de una exponencial y un polinomio de orden n :

$$k_{\nu=n+1/2} = \exp\left(-\frac{\sqrt{2\nu}r}{l}\right) \frac{\Gamma(n+1)}{\Gamma(2n+1)} \sum_{i=0}^n \frac{(n+i)!}{i!(n-i)!} \left(\frac{\sqrt{8\nu}r}{l}\right)^{n-i}.$$

Para el uso en Aprendizaje Automático, los casos más utilizados son $\nu = \frac{3}{2}$ y $\nu = \frac{5}{2}$. Para $1/2$ es demasiado 'irregular' y para $\nu \geq 7/2$ ya se aproxima al kernel RBF.

- **Función de Covarianza Exponencial:** Es un caso particular de la función de covarianza Matérn, concretamente para $\nu = \frac{1}{2}$. Es el caso azul de la Figura 5.1, que es generado por la siguiente función:

$$k(r) = \exp(-r/l).$$

Para una dimensión ($D = 1$) esta función de covarianza genera el conocido como **Proceso de Ornstein-Uhlenbeck** [UO30].

- **Función de Covarianza γ - exponencial:** Es una familia de funciones de covarianza que incluye la Función de Covarianza Exponencial y la Squared Exponential. Es similar a la función de Matérn, pero menos flexible. Viene dada por:

$$k(r) = \exp(-(r/l)^\gamma), \quad \text{para } 0 < \gamma \leq 2, \quad 0 < l.$$

Para $\gamma = 2$ coincide con la función de covarianza SE. En la Figura 5.3 se comparan las funciones de covarianza para valores de $\gamma = 0.75$, $\gamma = 1.5$, $\gamma = 2$. Se observa como las funciones aleatorias generadas con la distribución a priori ganan suavidad conforme aumenta el valor de γ .

5. Función de Covarianza y Estimación de Hiperparámetros

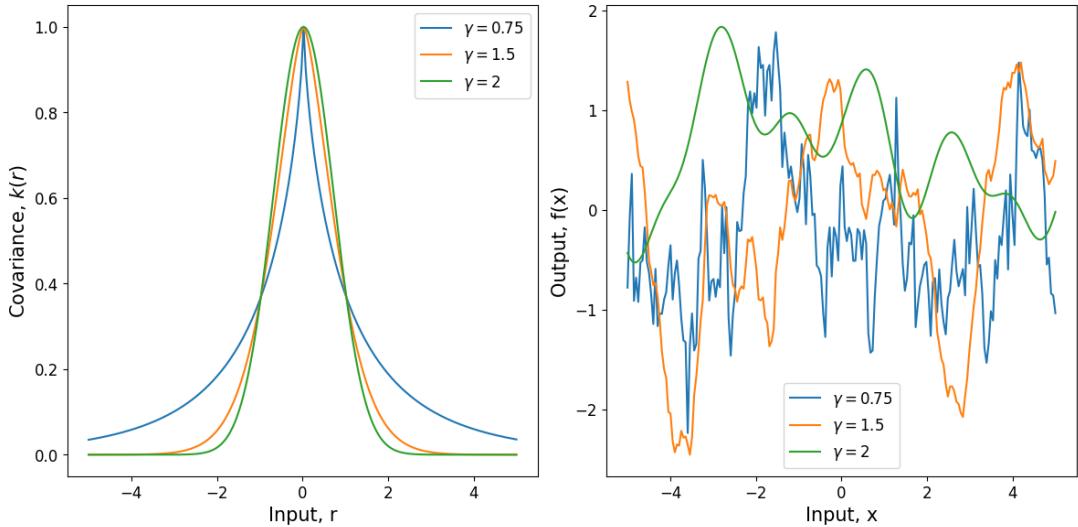


Figura 5.3.: Comparación de funciones de covarianza γ -exponencial para diferentes valores de γ . En la primera imagen se muestran las funciones de covarianza según los valores de γ , mientras que en la segunda se dibujan muestras aleatorias tomadas de la distribución a priori del Proceso Gaussiano generado con dichas funciones de covarianza. Código de programación propio, disponible en este [enlace](#).

- **Función de Covarianza Cuadrática Racional (RQ):** Se puede considerar como un 'scale mixture' (suma infinita) de Squared Exponentials con length-scales distintos. Es de la forma:

$$k_{RQ}(r) = \left(1 + \frac{r}{2\alpha l^2}\right)^{-\alpha},$$

con $\alpha, l > 0$. En la Figura 5.4 se pueden observar diferentes funciones de covarianza RQ dependiendo de su valor de α . Para comparar correctamente, se deja nuevamente el length-scale = 1 en los 3 casos. En este caso vemos que entre ellos varían menos que en los kernels Matérn.

- **Funciones de Covarianza Polinómicas por Partes con Soporte Compacto:** Es una familia de funciones de covarianza en las que por soporte compacto se entiende que para los puntos que su distancia excede un umbral, la función de covarianza será 0. En este tipo de función de covarianza es complejo garantizar que sea definida positiva. Algunos ejemplos de este tipo de función de covarianza son:

$$k_{ppD,0}(r) = (1 - r)_+^j$$

$$k_{ppD,1}(r) = (1 - r)_+^{j+1}((j+1)r + 1)$$

$$k_{ppD,2}(r) = (1 - r)_+^{j+2}((j^2 + 4j + 3)r^2 + (3j + 6)r + 3)/3$$

$$k_{ppD,3}(r) = (1 - r)_+^{j+3}((j^3 + 9j^2 + 23j + 15)r^3 + (6j^2 + 36j + 45)r^2 + (15j + 45)r + 15)/15$$

$$\text{donde } j = \frac{D}{2} + q + q.$$

Cabe destacar que todas las funciones de covarianza vistas en este apartado decaen mo-

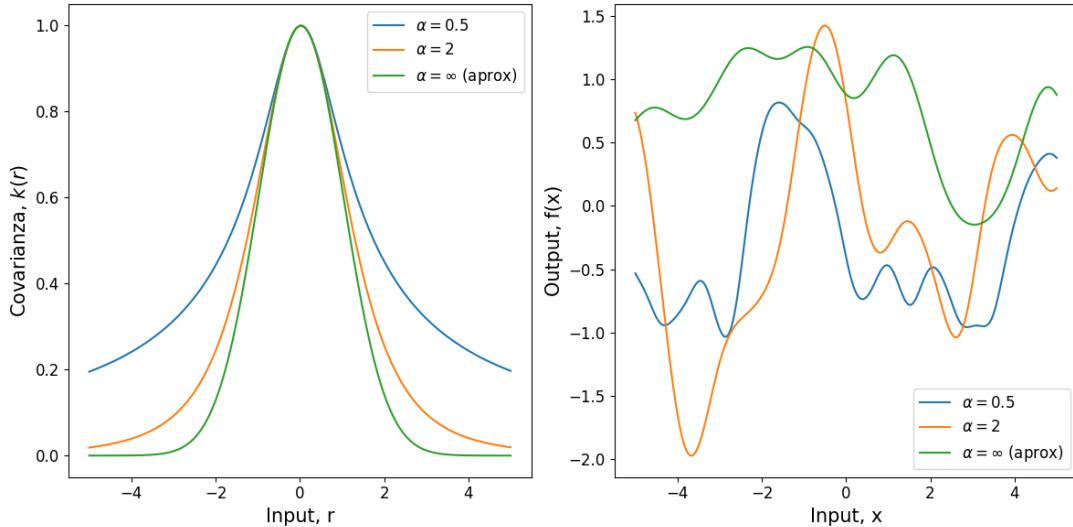


Figura 5.4.: Comparación de funciones de covarianza cuadráticas racionales para diferentes valores de α . En la primera imagen se muestran las funciones de covarianza según los valores de α , mientras que en la segunda se dibujan muestras aleatorias tomadas de la distribución a priori del Proceso Gaussiano generado con dichas funciones de covarianza.. Código de programación propio, disponible en este [enlace](#).

nótonamente con r y son siempre positivas, aunque esto no es una condición necesaria de las funciones de covarianza. También es importante resaltar que se podrían hacer **versiones anisotrópicas** de las funciones isotrópicas vistas usando $r^2 = (x - x')^T M (x - x')$ para una matriz semidefinida positiva M . De forma más general, se podría considerar M como $M = \Lambda \Lambda^T + \Psi$, donde Λ es una matriz de dimensión $D \times k$ cuyas columnas definen k direcciones de alta relevancia y Ψ es una matriz diagonal con valores positivos.

5.1.2. Funciones de Covarianza No Estacionarias

En este apartado se verán algunos ejemplos de funciones de covarianza que no son estacionarias. Existen numerosas opciones, pero en este apartado solo nombraremos algunas de las más importantes, empezando con los basados en el producto escalar. Como se mencionó anteriormente, el kernel $k(x, x') = \sigma_0^2 + \langle x, x' \rangle$ visto en la expresión (5.2) se puede obtener de la regresión lineal. Si $\sigma_0^2 = 0$, se llama **Kernel Lineal Homogéneo**. En caso contrario, se dirá **Kernel Lineal No Homogéneo**.

Se puede generalizar a $k(x, x') = \sigma_0^2 + x^T \Sigma_p x'$, usando como matriz de covarianzas Σ_p como se hizo en el Capítulo 3 en la expresión (3.2). También sería una función de covarianza $k(x, x') = (\sigma_0^2 + x^T \Sigma_p x')^n$, con $n \in \mathbb{N}$.

Otra opción sería la función de covarianza **Modulated Squared Exponential**, que dados un

5. Función de Covarianza y Estimación de Hiperparámetros

$u \sim \mathcal{N}(0, \sigma_u^2 I)$ y una escala σ_g se define como:

$$h(x; u) = \exp(-|x - u|^2 / 2\sigma_g^2).$$

También se pueden introducir funciones de covarianza no estacionarias mediante una **transformación no lineal arbitraria** (o deformación) $u(x)$ de la entrada x , y luego usar una función de covarianza estacionaria en el espacio u . Por ejemplo, con $u(x) = (\cos(x), \sin(x))$, obteniendo así una función **periódica** de x . Si usásemos el kernel Squared Exponential en este espacio u , obtendríamos:

$$k(x, x') = \exp\left(-\frac{2\sin^2(\frac{x-x'}{2})}{l^2}\right),$$

ya que $(\cos(x) - \cos(x'))^2 + (\sin(x) - \sin(x'))^2 = 4\sin^2(\frac{x-x'}{2})$.

Paciorek y Schervish idearon una fórmula para obtener funciones de covarianza no estacionarias a partir de funciones de covarianza isotrópicas. Definieron la forma cuadrática

$$Q_{ij} = (x_i - x_j)^T \left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} (x_i - x_j),$$

y utilizando el kernel isotrópico k_S idearon la siguiente función de covarianza no estacionaria:

$$k_{KS}(x_i, x_j) = 2^{D/2} |\Sigma_i|^{1/4} |\Sigma_j|^{1/4} |\Sigma_i + \Sigma_j|^{-1/2} k_S(\sqrt{Q_{ij}}).$$

Una idea que podría surgir para crear nuevas funciones de covarianza no estacionarias es variar los *length-scales* con las entradas x . Sin embargo, esto en general no creará funciones de covarianza válidas.

5.1.3. Generación de Nuevos Kernels a partir de Otros Existentes

Para finalizar con la sección de las funciones de covarianza, estudiaremos como se pueden crear nuevos kernels a través de otros ya conocidos.

- La **suma** de dos kernels es un kernel: Sean f_1 y f_2 dos procesos estocásticos independientes con kernels k_1 y k_2 . Entonces, la suma de ambos es $f = f_1 + f_2$ y su kernel es $k(x, x') = k_1(x, x') + k_2(x, x')$. Este resultado es útil por ejemplo para añadir kernels con diferentes *length-scales*.
- El **producto** de dos kernels es un kernel: Sean f_1 y f_2 dos procesos estocásticos independientes con kernels k_1 y k_2 . Entonces, el producto de ambos es $f = f_1 f_2$ y su kernel es $k(x, x') = k_1(x, x') k_2(x, x')$. Gracias a este resultado podemos determinar que, para un kernel $k(x, x')$, $k^n(x, x')$ con $n \in \mathbb{N}$ será también un kernel.
- Sean $a(x)$ una función y $f(x)$ un proceso estocástico. Considerando $g(x) = a(x)f(x)$, entonces $\text{cov}(g(x), g(x')) = a(x)k(x, x')a(x')$. Esto puede ser usado para normalizar

kernels, tomando $a(x) = k^{-1/2}(x, x)$ asumiendo que $k(x, x) > 0 \forall x$:

$$\tilde{k}(x, x') = \frac{k(x, x')}{\sqrt{k(x, x)} \sqrt{k(x', x')}}.$$

5.2. Estimación de Hiperparámetros

En muchas aplicaciones prácticas no es fácil especificar todos los aspectos de la función de covarianza. Algunas propiedades, como el ser estacionarias o no, pueden determinarse fácilmente a partir del contexto, pero generalmente solo contamos con información bastante superficial sobre otras propiedades. En la práctica, para utilizar de manera efectiva los Procesos Gaussianos, es necesario establecer herramientas para **seleccionar el modelo** a emplear, incluyendo la **función de covarianza** y los **hiperparámetros**.

La selección del modelo puede ayudar tanto a refinar las predicciones del mismo como a proporcionar una interpretación valiosa al usuario sobre las propiedades de los datos. Como se vio en la sección 5.1, existen una multitud de posibles familias de funciones de covarianza. La elección de una función de covarianza para una aplicación específica comprende tanto el ajuste de hiperparámetros dentro de una familia como la comparación entre diferentes familias, aunque ambos problemas serán tratados por los mismos métodos. Durante este apartado, se proporcionan ejemplos de elecciones de parametrizaciones de medidas de distancia para funciones de covarianza estacionarias en Procesos Gaussianos.

A la hora de elegir el modelo, se suelen seguir 3 pasos:

1. Calcular la probabilidad del modelo dados los datos.
2. Estimar el error de generalización.
3. Acotar el error de generalización.

Con **error de generalización** nos referimos al error promedio en ejemplos de prueba no vistos durante el entrenamiento. Cabe destacar que el error de entrenamiento suele ser un mal indicador del error de generalización, ya que el modelo puede ajustarse demasiado en específico al conjunto de entrenamiento (sobreajuste), lo que conduce a un bajo error de entrenamiento pero a un pobre rendimiento de generalización.

En los Capítulos 3 y 4 se han utilizado **modelos bayesianos** para introducir el uso de los Procesos Gaussianos. En esta ocasión haremos lo mismo, viendo como se elige el modelo en estos modelos para, posteriormente, aplicarlo a problemas de regresión y clasificación con Procesos Gaussianos.

Se suele realizar una especificación jerárquica de los modelos. A un **primer nivel**, se eligen los parámetros, w . Por ejemplo, podrían ser los pesos en un modelo lineal o una red neuronal. A un **segundo nivel**, están los hiperparámetros θ , que controlan la distribución de los parámetros en el nivel anterior (como podría ser el *weight decay* en redes neuronales). A un **tercer nivel**, se pueden considerar un posible conjunto discreto de estructuras del modelo, \mathcal{H}_i . La idea consiste en realizar inferencia según estos niveles.

5. Función de Covarianza y Estimación de Hiperparámetros

Con el Teorema de Bayes, se realiza inferencia al primer nivel, obteniendo la siguiente **distribución a posteriori** que combina la información de la distribución a priori y los datos:

$$p(w | y, X, \theta, \mathcal{H}_i) = \frac{p(y | X, w, \mathcal{H}_i) p(w | \theta, \mathcal{H}_i)}{p(y | X, \theta, \mathcal{H}_i)}, \quad (5.4)$$

donde $p(y | X, w, \mathcal{H}_i)$ es la **verosimilitud** y $p(w | \theta, \mathcal{H}_i)$ es la **distribución a priori**. El denominador normalizador es la **verosimilitud marginal** y viene dada por la siguiente expresión:

$$p(y | X, \theta, \mathcal{H}_i) = \int p(y | X, w, \mathcal{H}_i) p(w | \theta, \mathcal{H}_i) dw. \quad (5.5)$$

A segundo nivel, se realiza inferencia sobre θ :

$$p(\theta | y, X, \mathcal{H}_i) = \frac{p(y | X, \theta, \mathcal{H}_i) p(\theta | \mathcal{H}_i)}{p(y | X, \mathcal{H}_i)},$$

donde $p(\theta | \mathcal{H}_i)$ recibe el nombre de **hyper-prior** (distribución a priori para los hiperparámetros) y la constante normalizadora del denominador viene dada por:

$$p(y | X, \mathcal{H}_i) = \int p(y | X, \theta, \mathcal{H}_i) p(\theta | \mathcal{H}_i) d\theta. \quad (5.6)$$

Al tercer nivel, se calcula la distribución a posteriori como:

$$p(\mathcal{H}_i | y, X) = \frac{p(y | X, \mathcal{H}_i) p(\mathcal{H}_i)}{p(y | X)},$$

donde $p(y | X) = \sum_i p(y | X, \mathcal{H}_i) p(\mathcal{H}_i)$.

Como se ve, la implementación de la inferencia bayesiana conlleva la evaluación de varias **integrales**, que dependiendo de los detalles del modelo podrán ser tratables analíticamente o no, por lo que es probable que se deban utilizar aproximaciones o Métodos de Montecarlo basados en cadenas de Markov. En la práctica, la integral que da más problemas es la que aparece en la expresión (5.6) de la verosimilitud marginal del segundo nivel, que puede ser aproximada con el método de Laplace ya estudiado en el apartado 4.4. Otra opción es evitar usar esta expresión y maximizar la verosimilitud marginal en la ecuación (5.5) con respecto a los hiperparámetros, θ .

Para la selección de hiperparámetros del modelo también es común el uso de algunas técnicas ya comentadas en el apartado 2.3, como el **Cross-Validation**. Esta técnica está basada en **hold-out**, que consiste en dividir el dataset en dos conjuntos disjuntos, uno para entrenamiento y otro para validación. Este segundo conjunto se utiliza para monitorear el rendimiento a través de un indicador de error de generalización, con el cual se va a llevar a cabo la selección del modelo.

En la práctica, se utiliza el método de **k -fold cross validation**, que divide el dataset en k conjuntos disjuntos de igual tamaño. Se realiza validación en un único subconjunto, utilizando los $k - 1$ restantes para el entrenamiento. Se realiza el proceso k veces, cada vez con un subconjunto diferente para validación. El mayor problema de **k -fold cross validation** es que en

lugar de 1 modelo entrenas k , estando normalmente este número k utilizado en el intervalo $[3 - 10]$. Junto a *cross validation* se puede utilizar cualquier función de pérdida, aunque la más usual en regresión es MSE y en clasificación el *accuracy*. En modelos probabilísticos como los Procesos Gaussianos también es una opción utilizar como pérdida la probabilidad logarítmica negativa.

Un caso extremo de *k-fold cross validation* es ***leave-one-out*** (LOO), donde $k = n$, siendo n el tamaño del conjunto de entrenamiento. Lógicamente, normalmente este método solo será utilizado cuando n sea pequeño, ya que si no tendría un altísimo coste computacional, haciéndolo inviable. Sin embargo, en algunos casos concretos, como con los problemas de regresión en Procesos Gaussianos, existen algunos métodos simplificados para hacer 'atajos' en los cálculos.

Ahora, nos centraremos en los **problemas de regresión**. Los Procesos Gaussianos en problemas de regresión son una excepción en el uso de inferencia bayesiana en Aprendizaje Automático: normalmente, se necesitan cálculos que no se pueden llevar a cabo de forma analítica, lo que conduce al uso de aproximaciones que no siempre llevan a buenos resultados. Esto no sucede en los Procesos Gaussianos, donde además, el modelo que se genera tiene mucha flexibilidad. Para aplicar inferencia en los problemas de regresión, derivaremos la verosimilitud marginal e interpretaremos los resultados.

Si procedemos a aplicar inferencia bayesiana sobre el **primer nivel**, vemos que lo visto en las expresiones (5.4) y (5.5) ya se realizó en el Capítulo 3: la distribución predictiva de la distribución a posteriori es la vista en la expresión (3.9) y la verosimilitud marginal se dio en la ecuación (3.13).

Además, en la Proposición 3.1 se demostró que el logaritmo de esta verosimilitud marginal se podía reescribir como:

$$\log p(y | X, \theta) = -\frac{1}{2}y^T K_y^{-1}y - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi, \quad (5.7)$$

donde $K_y = K_f + \sigma_n^2 I$.

Los tres términos de la expresión (5.7) tienen roles claros:

- $-\frac{1}{2}y^T K_y^{-1}y$ es el único término que involucra los *targets* observados es el ajuste de los datos.
- $\log|K_y|$ es la penalización por complejidad que depende únicamente de la función de covarianza.
- $\frac{n}{2}\log 2\pi$ es una constante de normalización.

Para determinar los hiperparámetros, maximizamos la verosimilitud marginal buscando las derivadas parciales de la expresión (5.7). Para ello, utilizaremos el siguiente resultado:

Proposición 5.1. *Dada una matriz K , la derivada de su inversa se calcula con la siguiente expresión:*

$$\frac{\partial K^{-1}}{\partial \theta} = -K^{-1} \frac{\partial K}{\partial \theta} K^{-1}$$

5. Función de Covarianza y Estimación de Hiperparámetros

Además, la derivada del logaritmo del determinante de K es:

$$\frac{\partial \log |K|}{\partial \theta} = \text{tr} \left(K^{-1} \frac{\partial K}{\partial \theta} \right)$$

Utilizando la Proposición 5.1, se tiene que la derivada de la expresión (5.7) es:

$$\begin{aligned} \frac{\partial}{\partial \theta_j} \log p(y | X, \theta) &= \frac{1}{2} y^T K^{-1} y - \frac{1}{2} \text{tr}(K^{-1} \frac{\partial K}{\partial \theta_j}) \\ &= \frac{1}{2} \text{tr}((\alpha \alpha^T - K^{-1}) \frac{\partial K}{\partial \theta_j}) \end{aligned} \quad (5.8)$$

donde $\alpha = K^{-1}y$. La dificultad del cálculo dependerá de la inversa de K , que, considerando la matriz K de dimensión $n \times n$, es de complejidad $\mathcal{O}(n^3)$. Una vez se tiene K^{-1} , la complejidad de la derivada de la expresión (5.8) es solo de $\mathcal{O}(n^2)$ por hiperparámetro. Cabe resaltar que no se tiene garantía de que exista un único máximo de la verosimilitud marginal.

En cuanto al caso de los **problemas de clasificación**, el cálculo de las derivadas de la verosimilitud marginal dependerá del método de aproximación utilizado, pero en general son cálculos muy engorrosos. Por ejemplo, Opper y Winther [OWoo] utilizaron el algoritmo EP con distribuciones de cavidad.

6. Experimentos

En este apartado se llevarán a cabo pruebas sobre el contenido estudiado previamente. Todo el trabajo está implementado en **Python** utilizando cuadernos de **Google Colab** y los códigos se incluyen en el siguiente [repositorio de GitHub](#). En la sección 6.1, se explorarán las diferencias entre un modelo determinístico, como la Regresión Kernel Ridge, y un modelo probabilístico, como los Procesos Gaussianos. Posteriormente, en las secciones 6.2 y 6.3, se abordarán casos específicos de problemas de clasificación y regresión, respectivamente.

6.1. Modelado Determinístico vs Modelado Probabilístico

En primer lugar, para ilustrar de forma rápida y sencilla las diferencias entre el enfoque probabilístico de los Procesos Gaussianos y el enfoque determinístico de la estimación puntual, llevaré a cabo un breve **experimento visual**, ya que no se entrará a evaluar los errores de forma numérica y exacta. En particular, presentaré un ejemplo de un problema de regresión utilizando tanto **Regresión Kernel Ridge** como **Procesos Gaussianos**. Este caso está inspirado en dos ejemplos proporcionados por *Scikit-Learn*: uno sobre la [regresión con Procesos Gaussianos](#) y otro con [una comparación concreta de estos dos modelos](#).

Tanto la Regresión Kernel Ridge como los Procesos Gaussianos usan el conocido como '*kernel trick*', que ya se estudió en el apartado 3.1.1. Esta técnica permite que los modelos sean lo suficientemente expresivos como para ser capaces de ajustarse a los datos de entrenamiento. Sin embargo, veremos que hay notables diferencias en el comportamiento de ambos modelos en problemas de Aprendizaje Automático.

La Regresión Ridge trata de buscar la función objetivo que sea capaz de minimizar el **Error Cuadrático Medio**, que es la métrica que utiliza como **Función de Pérdida**. Sin embargo, como ya se ha visto en varias ocasiones durante este trabajo, los Procesos Gausianos utilizan una solución probabilística usando el Teorema de Bayes, dando una distribución a posteriori a partir de una distribución a priori y una función de verosimilitud. Se podría decir que los Procesos Gausianos son una generalización bayesiana de la Regresión Ridge.

Se busca aproximar la función $f(x) = x \sin(x)$ en el intervalo $[0, 10]$, la cual representamos con 1000 puntos igualmente espaciados. De los cuales, se escogen de forma aleatoria 6 para ser utilizados como datos de entrenamiento. Véase la Figura 6.1.

6. Experimentos

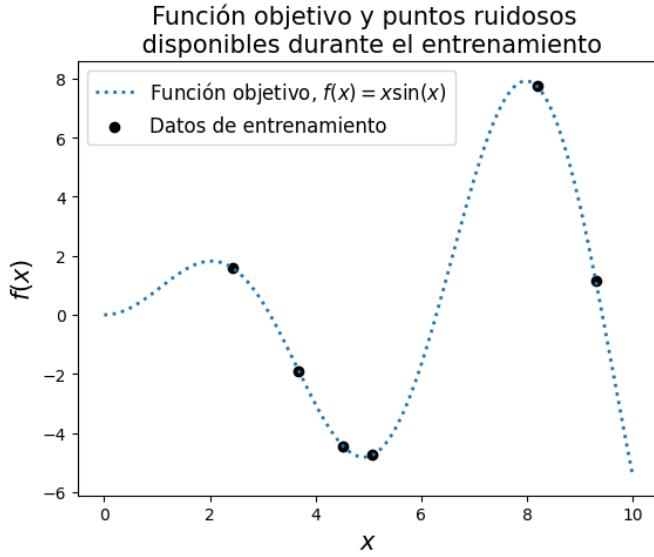


Figura 6.1.: Función objetivo y datos para el entrenamiento

En un primer lugar, pruebo a modelar la función con Regresión Ridge Simple. Obviamente, no se puede ajustar correctamente a la función objetivo y tiene claras limitaciones, como se puede observar en la siguiente Figura 6.2.

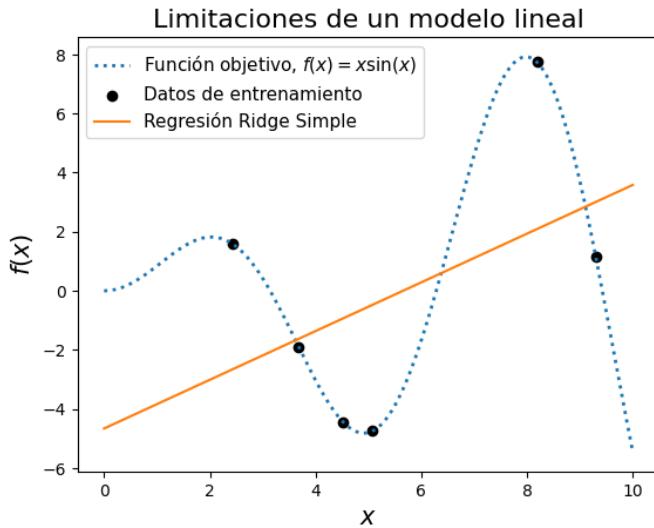


Figura 6.2.: Función objetivo y datos para el entrenamiento

Ante esta situación, utilice un kernel **RBF (Radial Basis Function)**, también conocido como **Squared Exponential**. Este tipo de kernel ha sido ampliamente estudiado a lo largo del trabajo, tanto en el Capítulo 3 como en el Capítulo 5. En primera instancia se han dejado los parámetros por defecto de este kernel, obteniendo unos resultados bastante deficientes. Por ello, he realizado una búsqueda con el método **RandomizedSearchCV** de *Scikit-Learn*, que

6.1. Modelado Determinístico vs Modelado Probabilístico

realiza una búsqueda aleatoria de los mejores hiperparámetros α y $length\ scale$, obteniendo unos valores de 0.0016 y 0.3355 respectivamente. Con estos valores, como se puede observar en la Figura 6.3, el modelo se acerca mucho a la función objetivo $f(x) = x \sin(x)$.

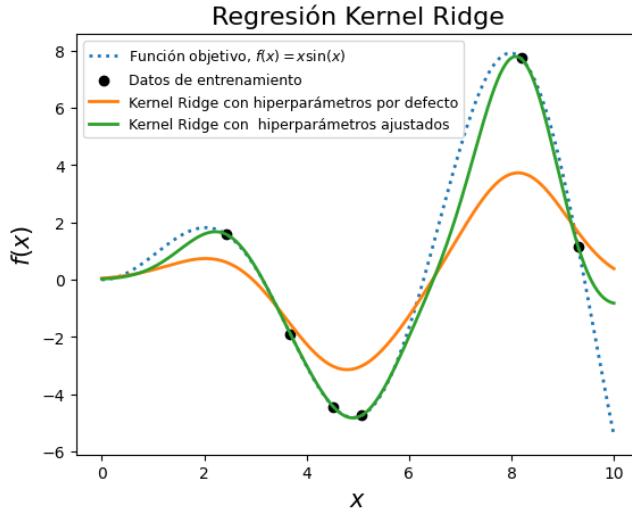


Figura 6.3.: Predicciones con Regresión Ridge utilizando un kernel RBF.

Para los Procesos Gaussianos, he utilizado el mismo kernel *RBF* o *Squared Exponential*. Al ajustar el modelo con el método '*fit*' los valores del kernel se optimizan automáticamente. El kernel obtenido finalmente tras el entrenamiento es: $5.02^2 * RBF(length_scale = 1.43)$.

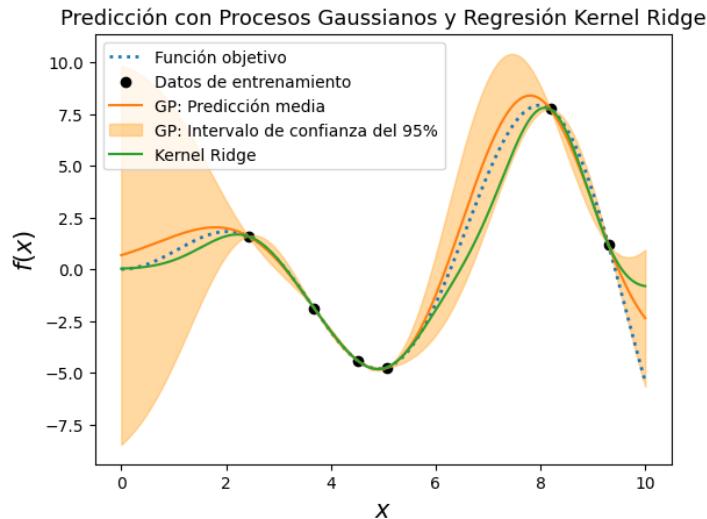


Figura 6.4.: Solución conjunta Procesos Gaussianos y Regresión Kernel Ridge. De naranja, la predicción media de los GP, con el sombreado mostrando un intervalo de confianza del 95 %. En verde se muestra la solución anterior con el Kernel Ridge.

En la Figura 6.4 se observa que los resultados de Kernel Ridge y Procesos Gaussianos son

6. Experimentos

similares (de hecho podemos ver que la predicción Kernel Ridge no sale del intervalo del 95 % de confianza del Proceso Gaussiano). Sin embargo, este último también ofrece información sobre la incertidumbre (sombreado naranja), una característica ausente en el Kernel Ridge. Gracias a la formulación probabilística de las funciones objetivo, el Proceso Gaussiano puede proporcionar la desviación típica, además de las predicciones medias de las funciones objetivo. Esta incertidumbre puede ser de gran utilidad en determinadas circunstancias, como en situaciones de alto riesgo como podría ser una aplicación médica en la que es muy importante que el sistema sea consciente de sus limitaciones.

6.2. Clasificación

En este apartado, utilizaré la clase '*GaussianProcessClassifier*' de la biblioteca Scikit-Learn para abordar problemas de clasificación. Esta clase utiliza la **aproximación de Laplace** estudiada en el apartado 4.4 y está basada en el algoritmo aportado por Rasmussen y Williams en [RW⁺06]. En concreto, durante este apartado se tratarán el problema de clasificación de dígitos manuscritos en la sección 6.2.1 y el problema de clasificación de flores iris en la sección 6.2.2. Ambos archivos .ipnyb se encuentran en el [repositorio de GitHub](#).

6.2.1. Optical Recognition of Handwritten Digits

En este apartado, el objetivo es asignar números del 0 al 9 a cada una de las instancias del conjunto de datos disponible en el siguiente [enlace](#) del *UC Irvine Machine Learning Repository*. Este dataset está formado por 5620 instancias (imágenes) que contienen información sobre dígitos manuscritos. Cada una consta de 64 valores enteros en el rango de [0, 16], junto con una etiqueta que indica el dígito manuscrito a clasificar, situado entre 0 y 9. Cada característica representa la intensidad de un píxel en la imagen



Figura 6.5.: Ejemplos de instancias

Además de utilizar Procesos Gaussianos, probaré con otro par de modelos de Aprendizaje Automático para comparar resultados:

- **Random Forest:** Este algoritmo se basa en la construcción de múltiples **árboles de decisión** independientes y no correlacionados entre sí. Cada árbol se entrena utilizando una porción aleatoria del conjunto de datos de entrenamiento y solo un subconjunto aleatorio de características en cada división del árbol. Esto se conoce como *bagging* o *bootstrap aggregating*, que ayuda a reducir la varianza y el sobreajuste del modelo. Una vez que se han entrenado todos los árboles de decisión, Random Forest realiza predicciones combinando las predicciones individuales de cada árbol. Para problemas de clasificación, se utiliza una estrategia de **voto mayoritario**, donde la clase que recibe la mayoría de votos entre todos los árboles se elige como la predicción final.

- **KNN:** Tiene un funcionamiento sencillo pero bastante efectivo. En el entrenamiento, el algoritmo simplemente almacena todos los puntos de datos y sus respectivas etiquetas en el espacio de características y cuando se realiza una predicción para un nuevo punto de datos, el algoritmo busca los ***k* puntos más cercanos** a ese punto en el espacio de características. La cercanía se determina generalmente mediante una medida de distancia, como la distancia euclíadiana. A la hora de clasificar el algoritmo asigna la **etiqueta más frecuente** entre los *k* vecinos más cercanos al nuevo punto como su etiqueta de predicción.

Se reservan el 30% de los datos para evaluar en test, y además, se realizará durante el entrenamiento una validación cruzada de 5 particiones (**5 cross-validation**). Para crear los *fold* se ha empleado *StratifiedKFold*, de esta forma aseguro proporciones similares de cada clase en comparación con el conjunto de datos completo. Las métricas usadas, que también se utilizarán con los datos de test, son las siguientes:

-**Accuracy (Exactitud):** Indica la proporción de **predicciones correctas** realizadas por el modelo sobre el total de predicciones. Es una medida simple pero importante, ya que proporciona una visión general del rendimiento del modelo. Sin embargo, no distingue entre los diferentes tipos de errores que el modelo puede cometer. Cabe destacar que puede ser engañosa en casos de conjuntos de datos desbalanceados, donde una clase puede dominar en términos de frecuencia (no es nuestro caso).

-**F1-Score:** Es una métrica que combina la **precisión** y el **recall** (proporción de verdaderos positivos frente a falsos negativos) en una sola medida. Es especialmente útil en problemas de clasificación donde las clases están desbalanceadas. Se calcula como la media armónica entre precisión y recall, donde precisión es la proporción de verdaderos positivos respecto a todos los casos positivos predichos, y recall es la proporción de verdaderos positivos respecto a todos los casos positivos reales. El F1-score toma valores en el intervalo [0, 1], siendo 1 su mejor valor y 0 el peor.

A la hora de comenzar a trabajar con el problema, primero se lleva a cabo el **preprocesamiento de datos**, empezando por la comprobación de la ausencia de datos faltantes (*missing values*). En cuanto a la existencia de posibles *outliers*, los calculo utilizando el **Rango Intercuartílico (IQR)**. Se obtiene que la mayoría de instancias tienen 0 o 1 outliers o muchos (500, 600, 200...). Si por ejemplo quitase los 593 outliers de la característica 1 estaría quitando demasiadas instancias. Por lo tanto, me voy a decantar por no eliminar los outliers en este

6. Experimentos

problema.

Utilizando **diagramas de caja y bigotes (boxplots)**, se explora individualmente cada una de las 64 características presentes en cada instancia del conjunto de datos. Esto revela que existen características donde todos los valores son idénticos, o bien son en su mayoría iguales, con excepción de casos muy aislados. Estas características son las 0, 8, 24, 32, 39 y 56, que decido eliminarlas ya que no aportan nada. Véase la Figura 6.6.

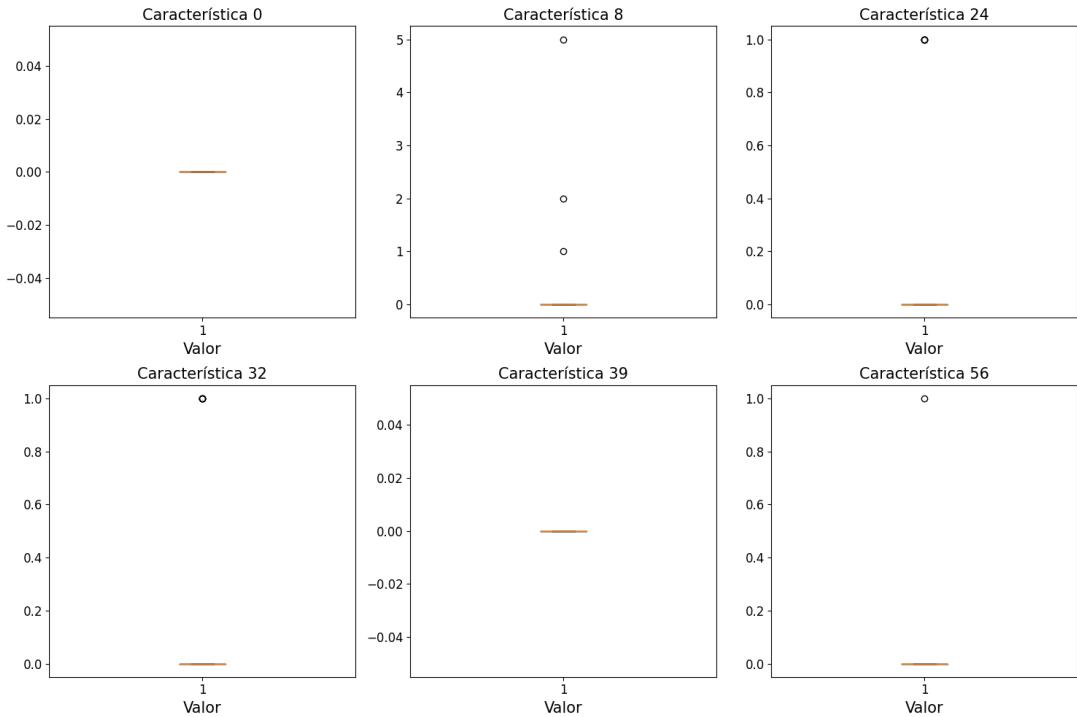


Figura 6.6.: Boxplots de las características inútiles que son eliminadas.

Aun así, sigue habiendo una gran cantidad de características, por lo que aplico a los datos un **Análisis de Componentes Principales (PCA)**. En el ámbito del Aprendizaje Automático, el PCA se utiliza como una técnica de preprocesamiento de datos para reducir la complejidad de los modelos y mejorar su rendimiento, especialmente en conjuntos de datos con muchas características redundantes o irrelevantes. Transforma un conjunto de datos con muchas variables correlacionadas en un conjunto de datos con menos variables no correlacionadas, llamadas **componentes principales**. Estas componentes están ordenadas por la cantidad de varianza que capturan en los datos originales. Las primeras componentes principales conservan la mayor cantidad de varianza, lo que permite resumir la información más importante del conjunto de datos.

Las componentes principales son ortogonales entre sí, lo que significa que son independientes linealmente. Esto facilita la interpretación de los patrones presentes en los datos y puede ayudar a eliminar la multicolinealidad entre las variables. En concreto, reduzco a 25 componentes principales, manteniendo el 92.8 % de la varianza total explicada como se

observa en la Figura 6.7. Conviene mencionar que antes de aplicar el PCA he normalizado los datos con la clase *MinMaxScaler*, lo cual es muy importante ya que si las características tienen diferentes escalas, aquellas con mayor varianza (debido a una escala más grande) dominarán en la determinación de las componentes principales. Las características con menor varianza podrían ser ignoradas, aunque realmente puedan ser igualmente importantes.

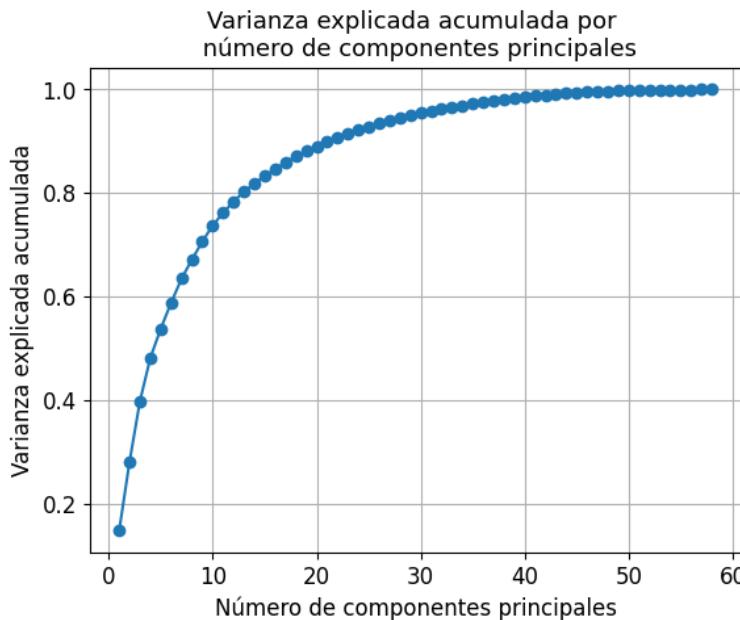


Figura 6.7.: Gráfico utilizado para determinar como 25 el número de componentes principales. Mantiene un 92.8 % de la varianza total explicada.

A la hora de buscar los mejores hiperparámetros para los modelos KNN y Random Forest, he utilizado el método *GridSearchCV*. Esto no es necesario para los Procesos Gaussianos, ya que el método *fit* hace la búsqueda internamente.

En Random Forest los mejores hiperparámetros obtenidos han sido:

- ***n_estimators*** = 200: Es el número de árboles en el bosque del modelo de Random Forest. Un mayor número de árboles puede mejorar la precisión del modelo, pero también incrementa el tiempo de entrenamiento.
- ***min_samples_split*** = 2: Es el número mínimo de muestras requeridas para dividir un nodo interno. Un valor más alto puede llevar a árboles más pequeños y menos sobreajustados.
- ***max_depth*** = 20: Es la profundidad máxima de cada árbol en el modelo. Controla cuánto puede crecer cada árbol. Si se establece en *None*, los nodos se expandirán hasta que todas las hojas sean puras o contengan menos muestras que *min_samples_split*.
- ***min_samples_leaf*** = 1: Es el número mínimo de muestras que debe tener un nodo hoja. Este parámetro puede evitar que los árboles creen nodos hoja con muy pocas muestras,

6. Experimentos

lo cual puede reducir el sobreajuste.

En KNN el único hiperparámetro introducido en el *GridSearchCV* ha sido *n_neighbors*. Es el número de vecinos más cercanos a considerar al hacer la predicción. Un valor pequeño puede hacer que el modelo sea más sensible al ruido, mientras que un valor grande puede hacer que el modelo sea más general y menos preciso. El valor óptimo obtenido ha sido 3.

Por otro lado, en el *GaussianProcessClassifier* es relevante destacar que, al no tener suposiciones sobre cual podría ser el **kernel** más adecuado, en la inicialización del *GaussianProcessClassifier* dejé el kernel por defecto: el RBF, ampliamente estudiado en este trabajo. En el entrenamiento el kernel se ajusta y se convierte en un *CompoundKernel*, que es una combinación de varios kernels básicos, que en este caso son todos kernels RBF con una longitud de escala igual a 1. El *CompoundKernel* da al modelo la posibilidad de capturar relaciones más complejas en los datos al sumar o multiplicar las características de los kernels individuales.

Como se ha comentado durante el trabajo y se ha visto en el experimento anterior, el trato de la **incertidumbre** es posiblemente la gran ventaja de usar Procesos Gaussianos. La clase *GaussianProcessClassifier* no proporciona intervalos de confianza explícitos, pero con el método *predict_proba* se pueden obtener las probabilidades por clase que da para una instancia el modelo, lo que se puede interpretar como una medida de la incertidumbre en las predicciones. Por ejemplo, vemos en la siguiente instancia las probabilidades que da el modelo para cada número. El modelo tiene muy claro que el número se trata de un 0.

- **Clase real: 0.** Predicciones:

- Clase 0: Probabilidad 0.890
- Clase 1: Probabilidad 0.010
- Clase 2: Probabilidad 0.011
- Clase 3: Probabilidad 0.011
- Clase 4: Probabilidad 0.011
- Clase 5: Probabilidad 0.013
- Clase 6: Probabilidad 0.010
- Clase 7: Probabilidad 0.011
- Clase 8: Probabilidad 0.012
- Clase 9: Probabilidad 0.021

En la Figura 6.8 se muestran las **curvas de aprendizaje** del modelo *GaussianProcessClassifier*. La puntuación de precisión en el conjunto de entrenamiento es muy alta (cercana a 1) de forma consistente. Esto indica que el modelo está ajustando casi perfectamente los datos de entrenamiento, lo cual podría sugerirnos que el dataset es muy sencillo o que también existe la posibilidad de estar cometiendo *overfitting*. Pero viendo la curva de validación vemos que no hay sobreajuste, ya que la puntuación de precisión en el conjunto de validación es también alta y se va acercando a la puntuación de entrenamiento, lo que sugiere que el modelo generaliza bien a nuevos datos. La cercanía entre ambas curvas es un buen indicio de que el modelo está bien ajustado y no está sufriendo un sobreajuste significativo. A continuación, confirmaremos que tenemos un modelo bien ajustado observando la matriz de confusión y comparando las tablas de resultados.

6.2. Clasificación

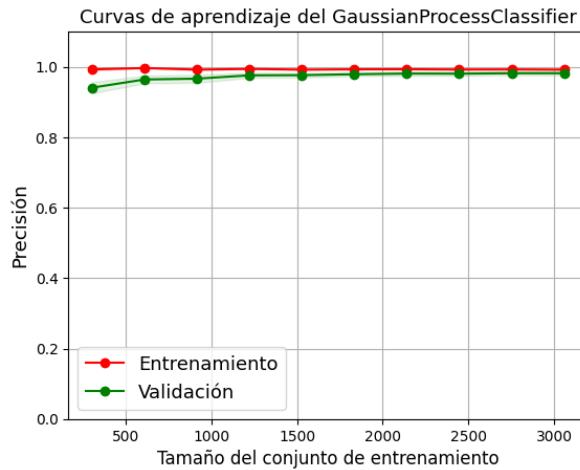


Figura 6.8.: Curvas de aprendizaje

En la siguiente imagen podemos ver la **matriz de confusión** del modelo de Procesos Gaussianos, que muestra el rendimiento del clasificador. La diagonal principal representa las predicciones correctas, que son amplia mayoría. Se puede observar que las clases 7 y 8 tienen más errores en comparación con otras clases, lo que indica que estas clases son más difíciles de distinguir para el modelo. Además, los errores no están concentrados al dar como predicción una clase específica, lo que sugiere que no hay un sesgo significativo hacia una clase en particular. En general, la matriz de confusión muestra que el *GaussianProcessClassifier* tiene un rendimiento muy bueno en este conjunto de datos. La alta precisión en la mayoría de las clases respalda las curvas de aprendizaje previamente analizadas, que indicaban un buen ajuste del modelo tanto en el conjunto de entrenamiento como en el de validación. Veremos a continuación una tabla con el accuracy y el F1-Score de los tres modelos que se han desarrollado.

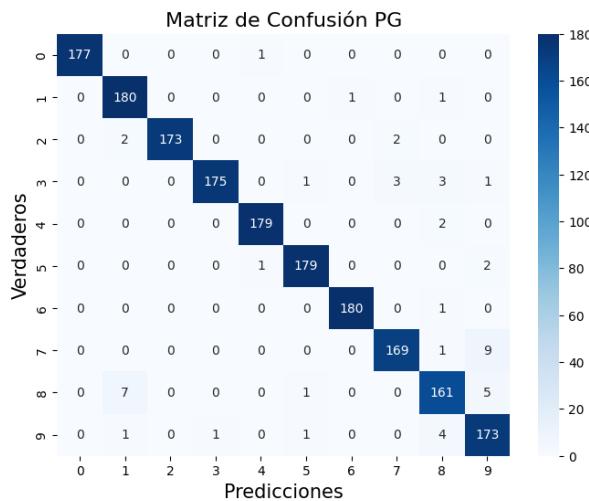


Figura 6.9.: Matriz de confusión.

6. Experimentos

Los resultados obtenidos en la validación del entrenamiento han sido los de la Tabla 6.1:

Modelo	Accuracy en cross-validation	F1 Score en cross-validation
Procesos Gaussianos	0.9819	0.9820
Random Forest	0.9717	0.9702
KNN	0.9832	0.9822

Tabla 6.1.: Resultados en la validación del entrenamiento

Mientras que en test, se ha obtenido la Tabla 6.2:

Modelo	Accuracy en test	F1 Score en test
Procesos Gaussianos	0.9716	0.9715
Random Forest	0.9610	0.9610
KNN	0.9783	0.9782

Tabla 6.2.: Resultados en test

Como conclusión general, los tres modelos muestran un rendimiento excelente, alcanzando una precisión de aproximadamente 97 % en el conjunto de test y un 98 % en el conjunto de entrenamiento. Aunque las diferencias entre ellos son mínimas, se pueden destacar algunos apuntes:

El modelo Random Forest muestra un rendimiento ligeramente inferior (1 % menos en precisión), lo cual es casi despreciable en muchos contextos. Por otro lado, a pesar de proporcionar una precisión alta similar a los otros modelos, el tiempo de predicción de los Procesos Gaussianos es significativamente mayor (6 minutos frente a unos segundos en los otros casos). Esto lo hace menos práctico para aplicaciones en tiempo real o con grandes volúmenes de datos, aunque también puede dar una predicción probabilística con incertidumbre.

En resumen, mientras que todos los modelos tienen un desempeño muy bueno, la elección del modelo adecuado puede depender de las necesidades específicas (precisión, tiempo de computación, incertidumbre...).

6.2.2. Iris

En esta sección se va a tratar el famoso problema de clasificar flores iris según 3 de sus especies: *Iris setosa*, *Iris virginica* o *Iris versicolor*. Este problema ya fue tratado en 1936 por **Fisher** en su paper 'The use of multiple measurements in taxonomic problems' [Fis36]. El dataset consta de 150 instancias, 50 por clase, y cada instancia incluye las siguientes características: longitud y ancho del sépalo, así como longitud y ancho del pétalo. El conjunto de datos está disponible en el siguiente [enlace](#).

6.2. Clasificación



Figura 6.10.: Fuente: [Wikipedia](#).

Como se tiene un número bajo de instancias, realizo una separación **estratificada** para asegurar una correcta división por clases en *train* y *test*. Se reservan el 25% de los datos (38 instancias) para test, utilizando las 112 instancias restantes para el entrenamiento. La mayor dificultad de este problema es la poquíssima cantidad de datos con la que se dispone. Al disponer de tan **pocas instancias**, existe el riesgo de que:

- El dataset no sea totalmente **representativo** de la verdadera distribución del problema que se intenta modelar.
- El modelo no generalice a datos no vistos durante el entrenamiento, es decir, que se cometa *overfitting*.
- Sea difícil para los algoritmos de aprendizaje detectar **patrones complejos**.
- La optimización de los hiperparámetros de los modelos no sea adecuada, ya que se tendrán muy pocos datos para dividir entre entrenamiento y validación, lo que puede llevar a una evaluación **menos fiable** de los modelos y sus configuraciones.

Se empieza realizando un **análisis estadístico** de las características de nuestro conjunto de datos de entrenamiento, utilizando el método *describe* de la librería *Pandas*. Así, en la Tabla 6.3 se ven los cuartiles, máximo, mínimo, media, desviación típica y mediana de cada característica. Se observa que los datos tienen unos valores relativamente bajos, no superiores a 8, lo cual tiene sentido ya que están dando los tamaños en centímetros de los pétalos y sépalos de las flores.

Tabla 6.3.: Resumen estadístico de los datos de entrenamiento

	Longitud (sépalo)	Ancho (sépalo)	Longitud (péntalo)	Ancho (péntalo)
Nº instancias	112	112	112	112
Media	5.8776	3.0660	3.7642	1.1937
Desv. Típica	0.8532	0.4436	1.7837	0.7703
Min	4.3	2	1.1	0.1
25 %	5.1	2.8	1.575	0.3
50 %	5.8	3	4.3	1.3
75 %	6.4	3.325	5.1	1.825
Max	7.9	4.4	6.9	2.5

6. Experimentos

Para ver la distribución de cada característica y descubrir posibles outliers, se vuelven a utilizar **diagramas boxplots** y el **Rango Intercuartílico (IQR)**. Como se observa en la Figura 6.11, se obtienen 3 outliers en la característica del ancho del sépalo, que proceden a ser eliminados del conjunto de entrenamiento: se tienen muy pocos datos, por lo que debemos asegurarnos de que al menos sean representativos

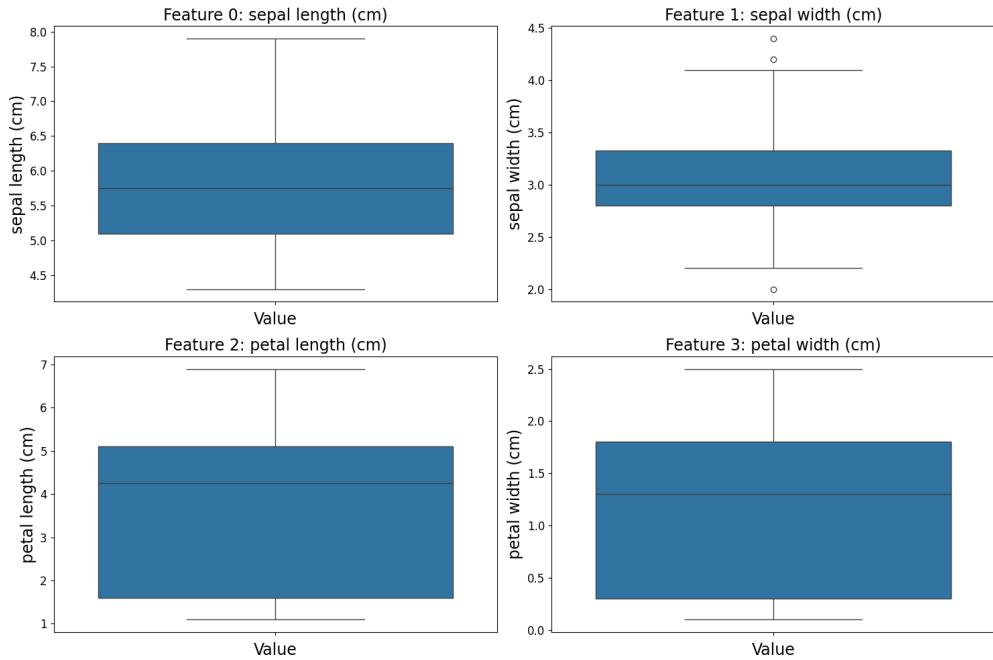


Figura 6.11.: Diagramas de cajas y bigotes (boxplots)

Para tener una mejor idea de la distribución exacta de nuestros datos según la clase, se muestran en la Figura 6.12 dos gráficas, distribuyendo las instancias de entrenamiento por longitud y ancho de pétalo y sépalo. Se observa que las características del pétalo dividen de una forma bastante clara los datos, mientras que el sépalo separa claramente las instancias de *setosa* pero mezcla un poco las instancias de *virginica* y *versicolor*.

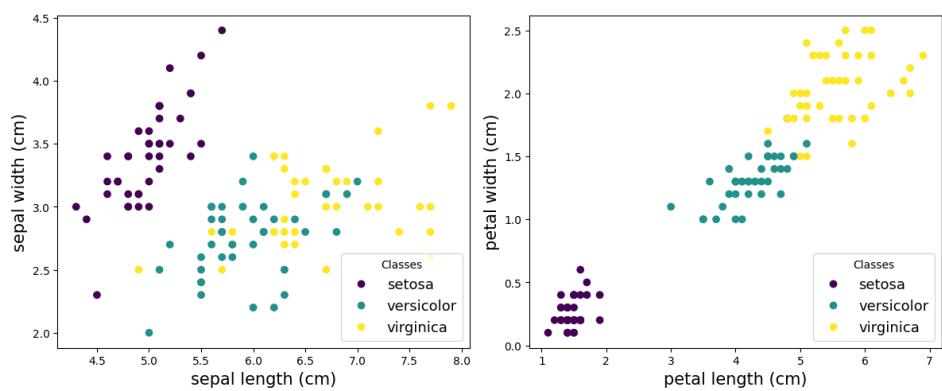


Figura 6.12.: Instancias de entrenamiento.

Viendo la Figura 6.12 se puede intuir que se podría reducir la **dimensionalidad** del problema, por lo que hago una normalización de mis datos y posteriormente estudio una posible reducción a través de **PCA**. En la Figura 6.13 se comprueba que perfectamente se podría disminuir a 2 o 3 componentes principales, conservando prácticamente la totalidad de la varianza. Me decanto por reducir a 3 dimensiones, ya que tenemos muy pocos datos, por lo que tampoco habrá problemas por el tamaño. Además, he probado con 2 componentes principales y los resultados empeoraban ligeramente en la validación del *cross-validation*. En la Figura 6.14 se observa como quedan los datos de entrenamiento distribuidos por el espacio según sus **3 componentes principales**.

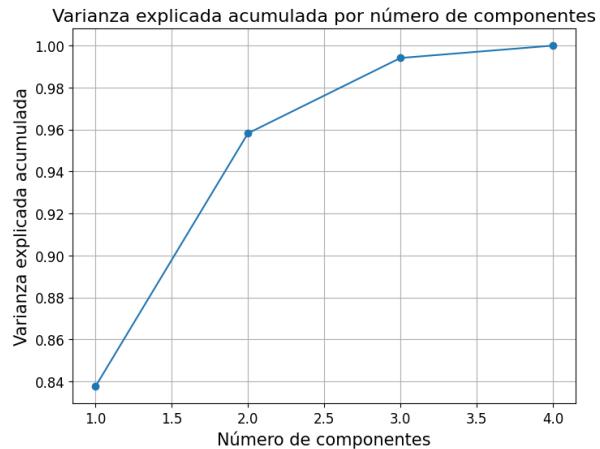


Figura 6.13.: Varianza explicada acumulada.

Datos reducidos a 3 componentes con PCA

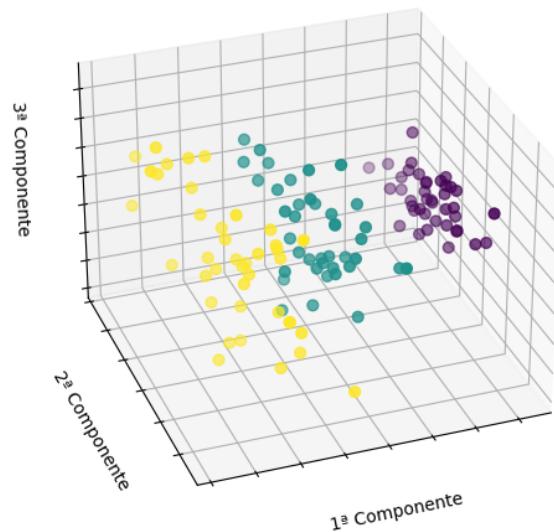


Figura 6.14.: Distribución por clase de las instancias de entrenamiento.

6. Experimentos

En esta ocasión voy a volver a utilizar las mismas métricas que en el apartado 6.2.1 para la validación experimental: **accuracy** y **f1-score**. Sin embargo, habrá cambios en cuanto a los algoritmos usados para los modelos: Crearé dos modelos *GaussianProcessClassifier*, con dos kernels RBF diferentes: uno **isotrópico** y otro **anisotrópico**, que han sido estudiados en la Sección 5. Además, para comparar con los Procesos Gaussianos utilizaré **KNN** (ya utilizado en el problema anterior) y **Support Vector Machines (SVM)**.

SVM es un modelo para problemas de aprendizaje automático que durante el entrenamiento busca encontrar el **hiperplano** que separe mejor las clases en el espacio de características. Es decir, aquel que maximiza la distancia entre el hiperplano y los puntos de datos más cercanos de cada clase, que son llamados **vectores de soporte**, de ahí el nombre del modelo. A la hora de realizar la predicción, SVM asignará una etiqueta dependiendo del lado del hiperplano en el que se encuentre el nuevo punto. En los casos en que los datos no son linealmente separables, SVM utiliza el **kernel trick**, ya estudiado en el apartado 3.1.1. También hay que destacar que SVM es sensible a la escala de las características y que puede tener un alto coste computacional cuando se trabaje en grandes conjuntos de datos.

En problemas de clasificación multiclas, como es nuestro caso (tenemos 3 clases: *Iris setosa*, *Iris virginica* e *Iris versicolor*), SVM se puede adaptar mediante las estrategias '*one vs one*' o '*one vs rest*'. La clase SVC de Scikit Learn, que será la que usaremos, de forma predeterminada emplea la estrategia '*one vs one*'. Supongamos que disponemos de n clases. Entonces, con esta estrategia el algoritmo entrena un clasificador SVM para cada par de clases, resultando en $\frac{n(n-1)}{2}$ clasificadores. Para una nueva muestra, se realiza una 'votación' entre todos los clasificadores para determinar la clase que se dará como predicción. En la estrategia '*one vs rest*', el algoritmo entrena n clasificadores, un clasificador SVM para cada una de las clases vs todas las demás clases. Finalmente, la clase que obtuviese la mayor puntuación es la que el modelo predice como solución.

Como he comentado anteriormente, crearé dos modelos *GaussianProcessClassifier*, que serán definidos con los siguientes kernels:

- Kernel RBF isotrópico: $kernel_iso = 1.0 * RBF([1.0])$
- Kernel RBF anisotrópico: $kernel_ani = 1.0 * RBF([1.0, 1.0, 1.0])$

Luego, durante el entrenamiento, se ajustan con el método *fit* y se convierten en los siguientes *CompoundKernel*:

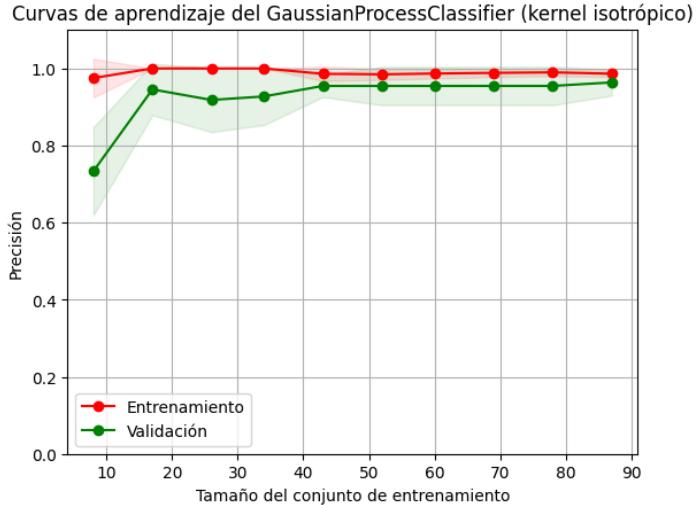
- *CompoundKernel* RBF isotrópico:

$$\begin{aligned} \text{Kernel 1} &: 38.2^2 * RBF(\text{length_scale} = 0.981) \\ \text{Kernel 2} &: 13.6^2 * RBF(\text{length_scale} = 0.459) \\ \text{Kernel 3} &: 22.9^2 * RBF(\text{length_scale} = 0.801) \end{aligned}$$

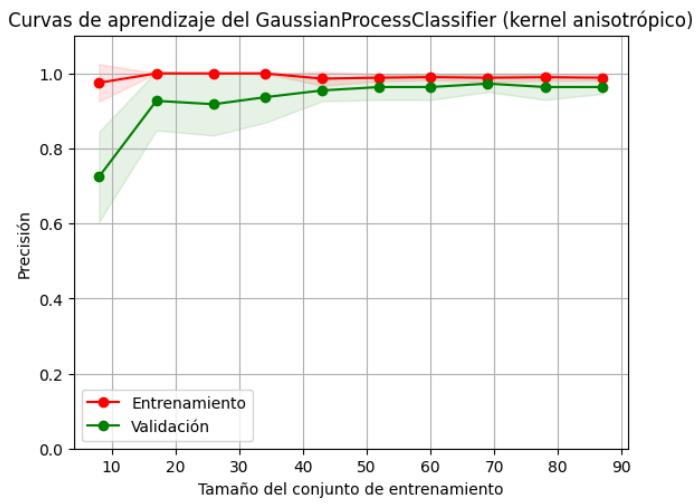
- *CompoundKernel* RBF anisotrópico:

$$\begin{aligned} \text{Kernel 1} &: 51.1^2 * RBF(\text{length_scale} = [0.78, 1.67e+04, 1.26e+03]) \\ \text{Kernel 2} &: 24.9^2 * RBF(\text{length_scale} = [0.348, 1.78, 2.78]) \\ \text{Kernel 3} &: 34.9^2 * RBF(\text{length_scale} = [0.55, 2.53, 3.42]) \end{aligned}$$

Las **curvas de aprendizaje** vemos que en esta ocasión no son tan buenas como en el problema anterior, pero aún así se obtiene un altísimo porcentaje de aciertos. En los gráficos de la Figura 6.15 podemos ver entre ambos modelos *GaussianProcessClassifier* no hay apenas cambios, son prácticamente idénticas.



(a) Kernel isotrópico



(b) Kernel anisotrópico

Figura 6.15.: Curvas de aprendizaje de los modelos *GaussianProcessClassifier*.

Sin embargo, a la hora de predecir se pueden observar pequeñas diferencias entre ambos modelos. Como ya se ha comentado, una de las mayores ventajas de los Procesos Gaussianos es el trato de la incertidumbre, y con el método *predict_proba* se pueden obtener las probabilidades que se predicen para cada clase. En la Figura 6.16 se muestran las instancias de

6. Experimentos

entrenamiento como puntos, distribuidos según las dos primeras componentes principales (por colores según la clase). También se genera una malla que se colorea según las probabilidades que dan los modelos sobre cada una de las 3 clases, utilizando el método `predict_proba`. Se observa que el modelo con kernel anisotrópico 'alarga' verticalmente más las predicciones en el espacio, mientras que el kernel isotrópico las 'redondea' más.

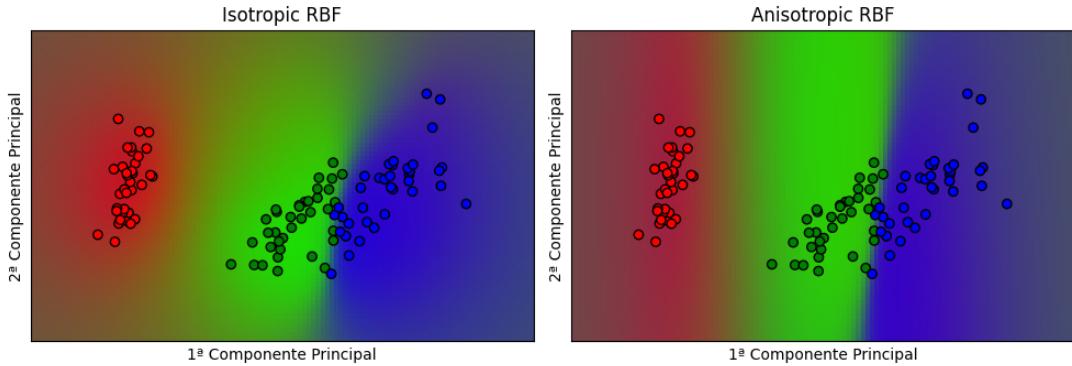


Figura 6.16.: Predicción probabilística de ambos modelos *GaussianProcessClassifier*.

Ahora veamos algunos ejemplos con instancias que han predicho los 2 modelos, comparando de forma más exacta la incertidumbre que dan ambos en forma de probabilidades:

- **Caso 1. Clase real = 0**

- Predicciones GPC con kernel RBF isotrópico:
 - Clase 0: Probabilidad 0.766
 - Clase 1: Probabilidad 0.094
 - Clase 2: Probabilidad 0.140
- Predicciones GPC con kernel RBF anisotrópico:
 - Clase 0: Probabilidad 0.596
 - Clase 1: Probabilidad 0.152
 - Clase 2: Probabilidad 0.252

- **Caso 2. Clase real = 1**

- Predicciones GPC con kernel RBF isotrópico:
 - Clase 0: Probabilidad 0.197
 - Clase 1: Probabilidad 0.728
 - Clase 2: Probabilidad 0.075
- Predicciones GPC con kernel RBF anisotrópico:
 - Clase 0: Probabilidad 0.193
 - Clase 1: Probabilidad 0.764
 - Clase 2: Probabilidad 0.043

- **Caso 3. Clase real = 1**

- Predicciones GPC con kernel RBF isotrópico:
 - Clase 0: Probabilidad 0.126
 - Clase 1: Probabilidad 0.853
 - Clase 2: Probabilidad 0.021
- Predicciones GPC con kernel RBF anisotrópico:
 - Clase 0: Probabilidad 0.172
 - Clase 1: Probabilidad 0.786
 - Clase 2: Probabilidad 0.042

Los modelos predicen la clase correctamente en los 3 casos. Esto es normal, ya que como se verá en las Tablas 6.4 y 6.5, se aciertan más del 90 % de las instancias del conjunto de test y el 96 % en entrenamiento. Además, las probabilidades de las clases se asemejan notablemente en ambos casos. Solo varía ligeramente en el primer ejemplo (no llega a 0.2 la diferencia). En la Figura 6.17 se marcan los 3 puntos de test utilizados y vemos que el punto de la izquierda es el que mostraba más diferencia. La explicación es que el kernel anisotrópico, como ya hemos comentado antes, 'alarga' verticalmente más las predicciones en el espacio, mientras que el kernel isotrópico 'redondea' más. Como esta instancia está a las 'afueras' del conjunto de las instancias de entrenamiento (mirándolo horizontalmente), entonces da algo menos de probabilidad a la clase roja. Aún así, sigue siendo la opción que predice el modelo.

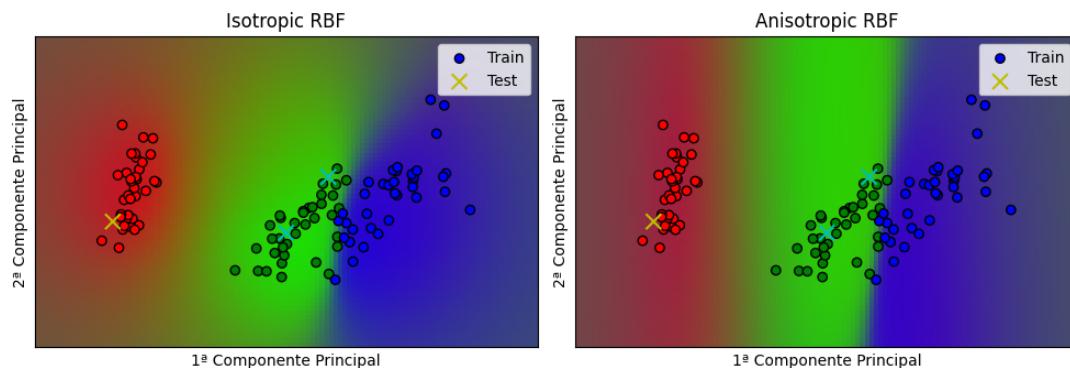


Figura 6.17.: Predicción probabilística de ambos modelos *GaussianProcessClassifier*.

En cuanto a los hiperparámetros de los modelos KNN y SVM, se buscan los valores óptimos con *GridSearchCV*. Para KNN el mejor valor de *n_neighbors* es 11, mientras que para en SVM se obtienen los siguientes hiperparámetros:

- $C = 1$: Es un parámetro de regulación, controla el equilibrio entre el ajuste de los datos de entrenamiento y la simplicidad del modelo.
- $\gamma = 1$: Es el coeficiente del término del kernel. Determina la influencia de un solo punto de entrenamiento. Hay que tener en cuenta que un valor alto de γ podría llevar a un modelo muy ajustado.
- $\text{kernel} = \text{RBF}$: Determina el tipo de kernel utilizado por el modelo. En este caso se ha obtenido el mismo tipo de kernel que se ha utilizado para los Procesos Gaussianos.

6. Experimentos

Los resultados obtenidos en la validación del entrenamiento se muestran en la Tabla 6.4:

Modelo	Accuracy (cross-validation)	F1-Score (cross-validation)
GP (kernel RBF isotrópico)	0.9636	0.9637
GP (kernel RBF anisotrópico)	0.9636	0.9637
SVM	0.9727	0.9726
KNN	0.9545	0.9632

Tabla 6.4.: Resultados de validación cruzada en el conjunto de entrenamiento

En general, el rendimiento de los modelos es muy bueno. Desde el principio se veían los datos muy bien distribuidos en el espacio y parecía lógico que se consiguiese una correcta clasificación, aunque la poca cantidad de datos disponibles podría perjudicar algo, como se explicó en un principio. Tanto la precisión como el F1-Score son muy elevados, siendo en algunos casos casi idénticos los valores. Por otro lado, los resultados en test se muestran en la Tabla 6.5:

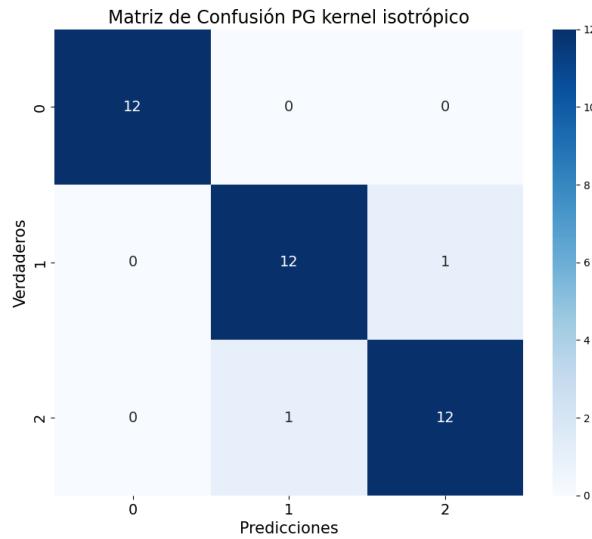
Modelo	Accuracy (test)	F1-Score (test)
GP (kernel RBF isotrópico)	0.9474	0.9487
GP (kernel RBF anisotrópico)	0.9211	0.9230
SVM	0.9211	0.9230
KNN	0.9737	0.9743

Tabla 6.5.: Resultados en test

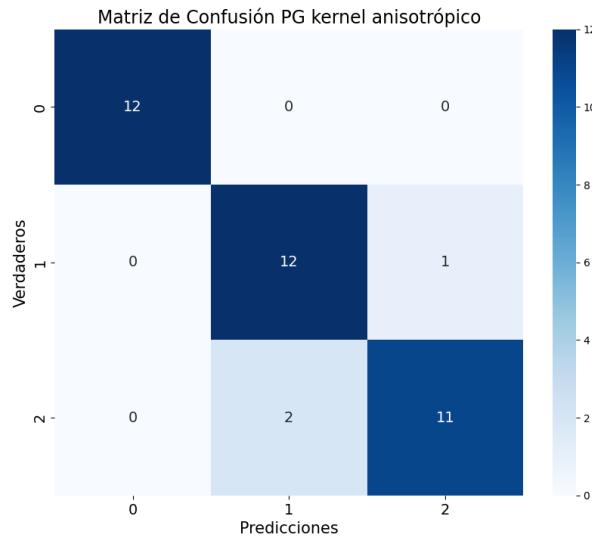
Tanto al *GaussianProcessClassifier* con kernel anisotrópico como al SVM les cuesta algo más generalizar a los datos de test que no han sido vistos durante el entrenamiento, seguramente por la poca cantidad de datos disponible para entrenar. Sin embargo, los resultados siguen siendo buenos, superiores al 90 %. Cabe resaltar que, sorprendente, el KNN mejora los resultados que obtuvo en el *cross-validation*.

En cuanto a los tiempos de ejecución, en esta ocasión no ha habido grandes diferencias entre modelos, a diferencia del apartado 6.2.1. Debido a la poquíssima cantidad del datos disponibles todas las ejecuciones se completaban al instante.

Para estudiar a fondo las predicciones en test de los modelos *GaussianProcessClassifier* se pueden utilizar **matrices de confusión** como las de la Figura 6.18. Debido a los pocos datos de los que se disponen, la diferencia de 2.5 % de accuracy entre ambos modelos está causada por un único fallo más. También se observa que los fallos se cometan entre las clases que quedaban más mezcladas en las visualizaciones de datos. En contraste, la clase que se encontraba más separada se predice sin errores, lo cual es completamente lógico.



(a) Kernel isotrópico



(b) Kernel anisotrópico

Figura 6.18.: Matrices de confusión de ambos modelos *GaussianProcessClassifier*.

6.3. Regresión

La clase de Scikit-Learn creada para abordar problemas de regresión, '[GaussianProcessRegressor](#)', será el elemento fundamental de este apartado y permitirá poner en práctica los conceptos estudiados durante el Capítulo 3. Conviene mencionar que, como indica la propia documentación, la implementación de la clase está basada en el algoritmo 1. En el aparta-

6. Experimentos

do [6.3.1](#) utilizaremos Procesos Gaussianos para hacer un estudio exhaustivo del problema de las mediciones de CO₂ en el volcán Mauna Loa. Cabe destacar que la clase *GaussianProcessRegressor* tiene algunos detalles que pueden ser de gran utilidad en determinados momentos:

- Utilizando la distribución a priori del Proceso Gaussiano es capaz de realizar predicciones sin necesidad de ajustar el modelo a los datos previamente.
- El método *log_marginal_likelihood(theta)* calcula el logaritmo de la verosimilitud marginal dados unos hiperparámetros *theta*. Este método puede ayudar notablemente a la selección de los mejores hiperparámetros, eligiendo los que maximicen esta probabilidad marginal.
- El método *sample_y(X)* permite evaluar con el modelo un conjunto de entradas X. Puede utilizarse tanto a partir de la distribución a priori como de la posteriori, resultando útil para entender la variabilidad y la distribución de las posibles predicciones del modelo.

6.3.1. Mediciones de CO₂ en el Volcán Mauna Loa

Este problema está basado en un ejemplo del libro de Williams y Rasmussen (apartado 5.4.3, [\[RW⁺06\]](#)), que también ha sido [utilizado como ejemplo por algunos de los desarrolladores de Scikit-Learn](#). Trata sobre mediciones de CO₂ que se han captado en el Observatorio del volcán Mauna Loa, en Hawái, desde 1958 hasta 2001. El dataset está disponible en la [página web de la Oficina Nacional de Administración Oceánica y Atmosférica de los Estados Unidos](#).

El dataset contiene 2225 **instancias** con las siguientes **7 características**: *year, month, day, weight, flag, station* y CO₂. Primero se comprueba que no haya *missing values* y procedemos al estudio de los datos: El CO₂ es el valor a predecir, que viene dado en la medida partes por millón (ppm). Las variables *weight* y *flag* no nos interesan para este ejercicio, por lo que serán eliminadas de nuestro conjunto de datos. Además, juntamos las columnas de día, mes y año en una única variable 'fecha', que posteriormente será transformado a una variable numérica. El propósito de este ejercicio es intentar crear un modelo con *GaussianProcessRegressor* que se adapte a los datos y nos permita extrapolar más allá del año 2001, que es cuando se acaban las mediciones disponibles.

Con el siguiente gráfico de la Figura [6.19](#) comprobamos que los datos son consistentes y se tienen datos a lo largo de todo el eje temporal:

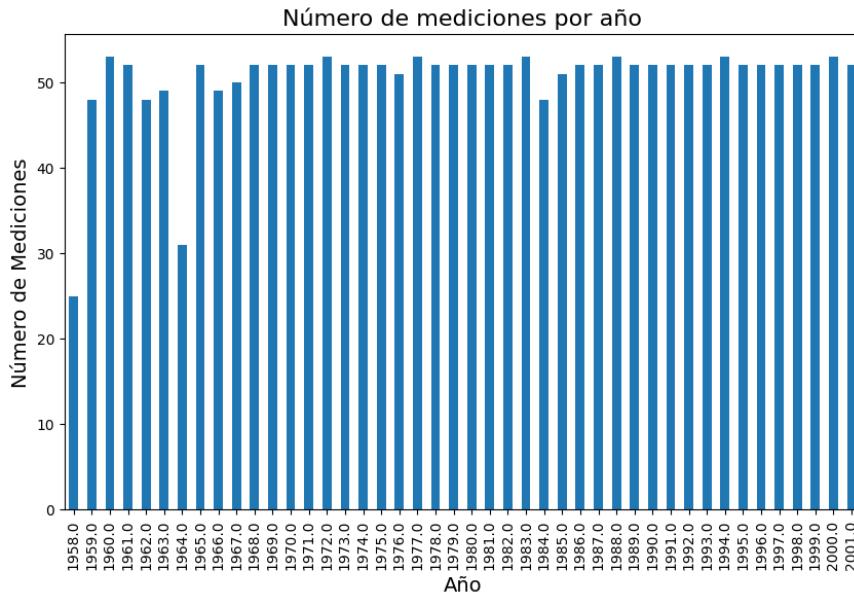


Figura 6.19.: Distribución anual de las mediciones de CO2.

Para suavizar los datos se lleva a cabo el siguiente preprocesamiento: Se toma la **media mensual** y se eliminarán los meses en los que no se haya recogido datos sobre los niveles de CO2. Una vez realizado este cambio, se procede a separar en **test** y **train**, reservando el 20 % de los datos para calificar los modelos en test. Obviamente, el valor X será la característica de la fecha, mientras que la y será el valor de la medición de CO2.

En la Tabla 6.6 se muestra un **resumen estadístico** de los datos de entrenamiento. Vemos que el número de instancias de entrenamiento ha quedado reducido a 416 una vez se ha hecho la media mensual. También podemos ver el rango de los valores de CO2, que van desde 313 ppm hasta 373ppm.

Tabla 6.6.: Resumen estadístico de los datos de entrenamiento

	Fecha	CO2 (ppm)
Nº instancias	416	416
Media	1980.88	340.17
Desv. Típica	12.34	16.81
Min	1958.33	313.4
25 %	1970.23	324.95
50 %	1980.88	339.22
75 %	1991.27	354.67
Max	2001.83	373.8

Ahora vemos en los **histogramas** de la Figura 6.20 la distribución de nuestros datos por variable. Se obtiene que las mediciones en el conjunto de entrenamiento están bastante distribuidas por el tiempo, no hay años con falta excesiva de mediciones, únicamente los

6. Experimentos

años 1963 y 1964 tienen menos de 6 meses con mediciones. En cuanto a los datos de CO₂, vemos que hay mayor cantidad de datos con valores bajos de ppm.

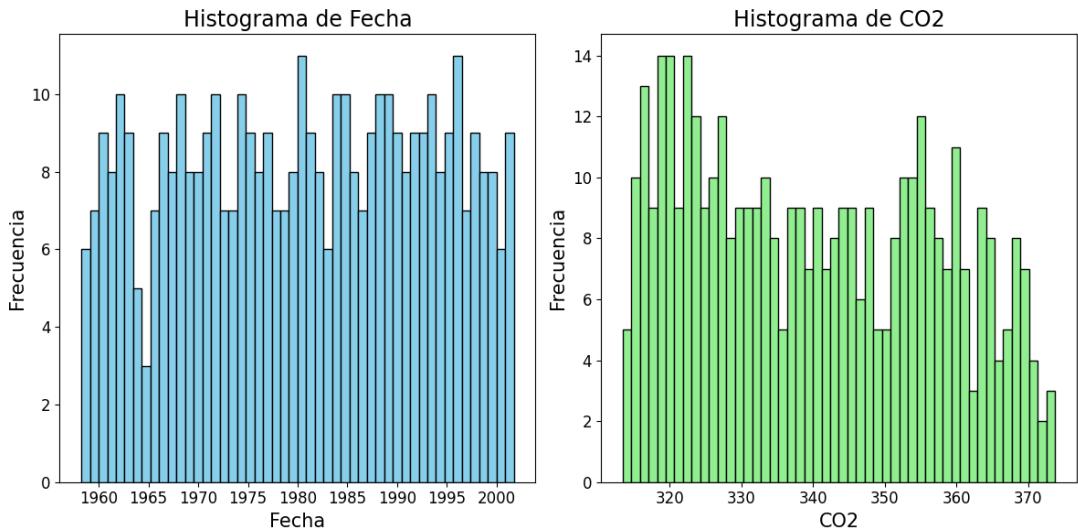


Figura 6.20.: Histograma de las variables.

Para tener una mejor idea de como se distribuyen de forma conjunta los **datos de entrenamiento**, observamos el gráfico de la Figura 6.21. Se puede intuir un claro patrón en los datos que los modelos deberían ser capaces de captar.

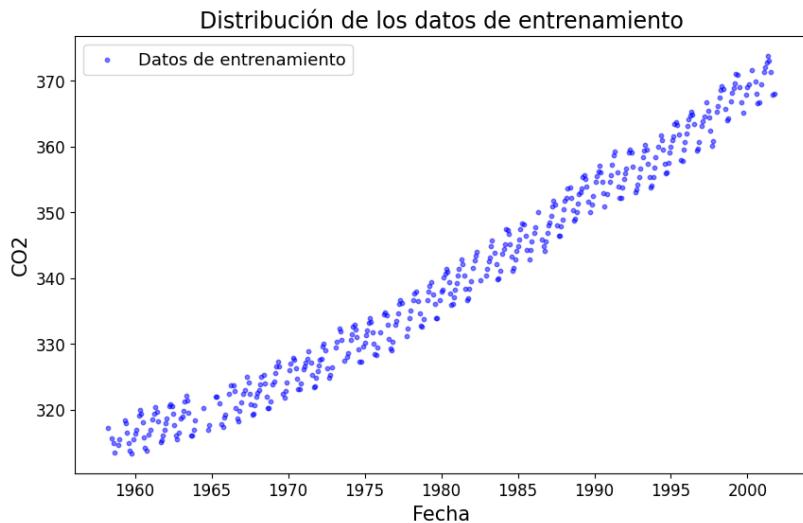


Figura 6.21.: Datos de entrenamiento.

Para la resolución del problema utilizaré 3 **modelos** diferentes: además del *GaussianProcessRegressor*, crearé un modelo con **Support Vector Machines** y otro con **Gradient Boosting**.

En el apartado 6.2.2 se trabajó también con SVM, sin embargo, como en esta ocasión se trabaja con un problema de regresión, habrá ciertas diferencias con el SVC utilizado anteriormente. Se usará la clase **Support Vector Regression (SVR)** de Scikit-Learn, que a diferencia del SVC no busca encontrar un hiperplano para separar las clases, sino que busca ajustar el mayor número posible de instancias en la 'calle', limitando al mismo tiempo las instancias que quedan fuera del margen. La anchura del margen se determina a través de un hiperparámetro ϵ .

En cuanto al **Gradient Boosting**, es un potente algoritmo de aprendizaje supervisado que se puede utilizar tanto en problemas de clasificación como de regresión. Utiliza la técnica del **boosting**, que consiste en juntar varios predictores débiles para hacerlos fuertes. En concreto, junta múltiples **árboles de decisión** que se entranan de forma secuencial, de forma que los árboles intentan corregir los errores que han cometido los árboles anteriores en el entrenamiento: cada árbol se ajusta a los residuos (diferencia entre los valores reales y los predichos) del modelo anterior en lugar de los datos originales. Este proceso se realiza iterativamente, minimizando una función de pérdida mediante el **descenso de gradiente**. Para evitar el overfitting este modelo utiliza regularización.

Las métricas utilizadas para observar el rendimiento de los modelos, tanto en test como en la validación del cross-validation, serán el **Error Cuadrático Medio (MSE)** y el **Coeficiente de Determinación (R^2)**.

El Error Cuadrático Medio es una medida comúnmente utilizada para evaluar el rendimiento de un modelo de regresión. Se calcula tomando la media de los cuadrados de las diferencias entre los valores predichos por el modelo y los valores reales del conjunto de datos de prueba. Cuanto menor sea el valor de MSE, mejor será el rendimiento del modelo, lo que indica que las predicciones del modelo están más cerca de los valores reales. Su fórmula es:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

donde n será el número de observaciones, y_i el valor real de la i -ésima observación e \hat{y}_i el valor que predice el modelo para la i -ésima observación.

Por otro lado, el R^2 es una medida que indica cuánta variabilidad en la variable de destino es explicada por las variables independientes del modelo. Un valor de 1 indica un ajuste perfecto, mientras que un valor de 0 indica que el modelo no explica ninguna variabilidad. Su fórmula es:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

donde \bar{y} es el valor medio de las observaciones reales, y_i el valor real de la i -ésima observación e \hat{y}_i el valor que predice el modelo para la i -ésima observación.

Para determinar los **mejores hiperparámetros** de los modelos, de nuevo se vuelve a repetir la misma situación de los apartados anteriores. En los Procesos Gaussianos se adaptarán los mejores hiperparámetros del kernel durante el entrenamiento, mientras que en SVR y Gradient Boosting se utilizará *GridSearchCV*.

6. Experimentos

En SVR los mejores hiperparámetros obtenidos (que ya han sido explicados anteriormente) han sido:

$$C : 100, \text{ epsilon} : 1, \text{ gamma} : 0.001, \text{ kernel} : rbf$$

En Gradient Boosting se han obtenido los siguientes:

- ***learning_rate*** = 0.1: Controla la contribución de cada árbol al modelo. Si el ***learning_rate*** es bajo se necesitarán más árboles pero el modelo generalizará mejor.
- ***max_depth*** = 5: Es la profundidad máxima de cada árbol de decisión. Árboles más profundos pueden capturar patrones más complejos pero pueden caer más fácilmente en *overfitting*.
- ***min_samples_leaf*** = 1: Es el número mínimo de muestras requeridas para ser consideradas como una hoja. Esto puede contribuir a la regularización del modelo.
- ***min_samples_split*** = 5: Es el número mínimo de muestras requeridas para dividir un nodo interno.
- ***n_estimators*** = 300: Determina el número de árboles de decisión del modelo. Un número mayor puede dar al modelo mayor precisión, pero también puede disparar el coste computacional.
- ***subsample*** = 0.8: Es la proporción de muestras que se utilizarán para entrenar cada árbol. Un valor pequeño puede reducir el *overfitting*.

En esta ocasión, para el Proceso Gaussiano no tiene sentido usar un kernel anisotrópico, ya que tenemos una sola característica. A diferencia de los apartados anteriores, en lugar de utilizar un kernel RBF, escogeré una combinación más compleja de kernels para que se adapten mejor a nuestros datos. Haremos algunas suposiciones sobre los datos de entrenamiento: se observa una tendencia creciente a largo plazo, una variación estacional pronunciada y algunas irregularidades más pequeñas. Con esta idea, combinaremos los siguientes kernels:

- Primero, la tendencia creciente a largo plazo podría ajustarse utilizando un **kernel RBF** con un parámetro ***length_scale*** elevado, que hará que esta componente sea suave. La escala específica y la amplitud son hiperparámetros libres que se ajustarán con exactitud durante el entrenamiento.

$$\text{long_term_trend_kernel} = 50.0^2 * RBF(\text{length_scale} = 50.0)$$

- La variación estacional se explica mediante el **kernel periódico exponencial seno cuadrado** con una periodicidad fija de 1 año. La escala de longitud de este componente periódico, que controla su suavidad, es un parámetro libre. Para permitir *decay* lejos de la periodicidad exacta, se utiliza el producto con un kernel RBF. La escala de longitud de este componente RBF controla el tiempo de *decay* y es otro parámetro libre. Este tipo de kernel también se conoce como kernel localmente periódico.

$$\begin{aligned} \text{seasonal_kernel} &= (2.0^2 * RBF(\text{length_scale} = 100.0) * \text{ExpSineSquared}(\\ &\quad \text{length_scale} = 1.0, \text{periodicity} = 1.0, \text{periodicity_bounds} = "fixed")) \end{aligned}$$

- Las pequeñas irregularidades observadas en los datos se explican mediante un componente de **kernel cuadrático racional**, cuyo *length_scale* y parámetro α , que cuantifica la difusión de los *length_scales*, deben determinarse durante el entrenamiento. Un kernel cuadrático racional es equivalente a un kernel RBF con varias escalas de longitud y se adaptará mejor a las diferentes irregularidades.

$$\text{irregularities_kernel} = 0.5^2 * \text{RationalQuadratic}(\text{length_scale} = 1.0, \alpha = 1.0)$$

- Por último, el ruido en el conjunto de datos se puede tener en cuenta con un kernel que consiste en una contribución del kernel **RBF**, el cual explica los componentes de ruido correlacionados como fenómenos meteorológicos locales, y una contribución de **White Kernel** para el ruido blanco. Las amplitudes relativas y el *length_scale* del RBF son parámetros libres adicionales.

$$\text{noise_kernel} = 0.1^2 * \text{RBF}(\text{length_scale} = 0.1) + \text{WhiteKernel}$$

$$\text{noise_level} = 0.1^2, \text{noise_level_bounds} = (1e-5, 1e5)$$

El kernel final con el que se inicializa el GPR se crea juntando los 4 kernels:

$$\text{co2_kernel} = (\text{long_term_trend_kernel} + \text{seasonal_kernel} + \text{irregularities_kernel} + \text{noise_kernel})$$

Las **curvas de aprendizaje** durante el *fit* se muestran en la Figura 6.22. Se observa como no comete prácticamente error, ni en el entrenamiento ni en la validación del *cross-validation*, por lo que presumiblemente el modelo se ajustará muy bien a los datos.

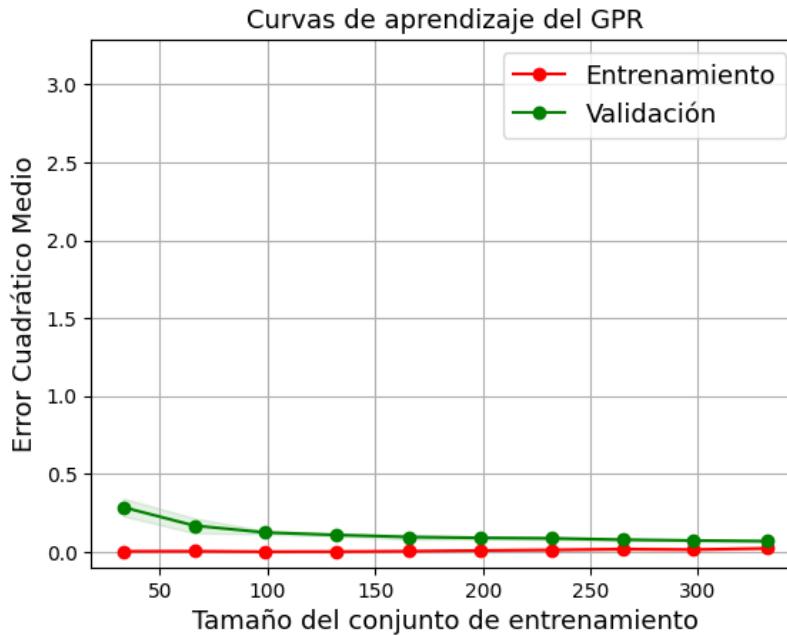


Figura 6.22.: Curvas de aprendizaje.

6. Experimentos

Hay que tener en cuenta que los datos de CO₂ van desde 313 hasta 373, por lo que un ECM menor que 0.5 es despreciable.

Veamos ahora el ECM y el R^2 de los tres modelos en la validación del *cross-validation* en la Tabla 6.7:

Modelo	MSE (cross-validation)	R^2 (cross-validation)
GaussianProcessRegressor	0.4470	0.9983
SVM	4.7683	0.9827
Gradient Boosting	1.4245	0.9949

Tabla 6.7.: Resultados de validación cruzada en el conjunto de entrenamiento

Los resultados son muy buenos una vez más, y el modelo *GaussianProcessRegressor* en esta ocasión es el mejor de los 3. Cabe destacar que se obtienen unos coeficiente de determinación prácticamente perfectos: casi toda la variabilidad en la variable de destino es explicada por las variables independientes de los modelos. Ahora se estudiarán los resultados obtenidos en test.

En la Figura 6.23 vemos la predicción que da el modelo en los datos de test, que se ajustan muy bien a los valores reales. Posteriormente se verá el MSE y R^2 , pero parece ser muy buen modelo.

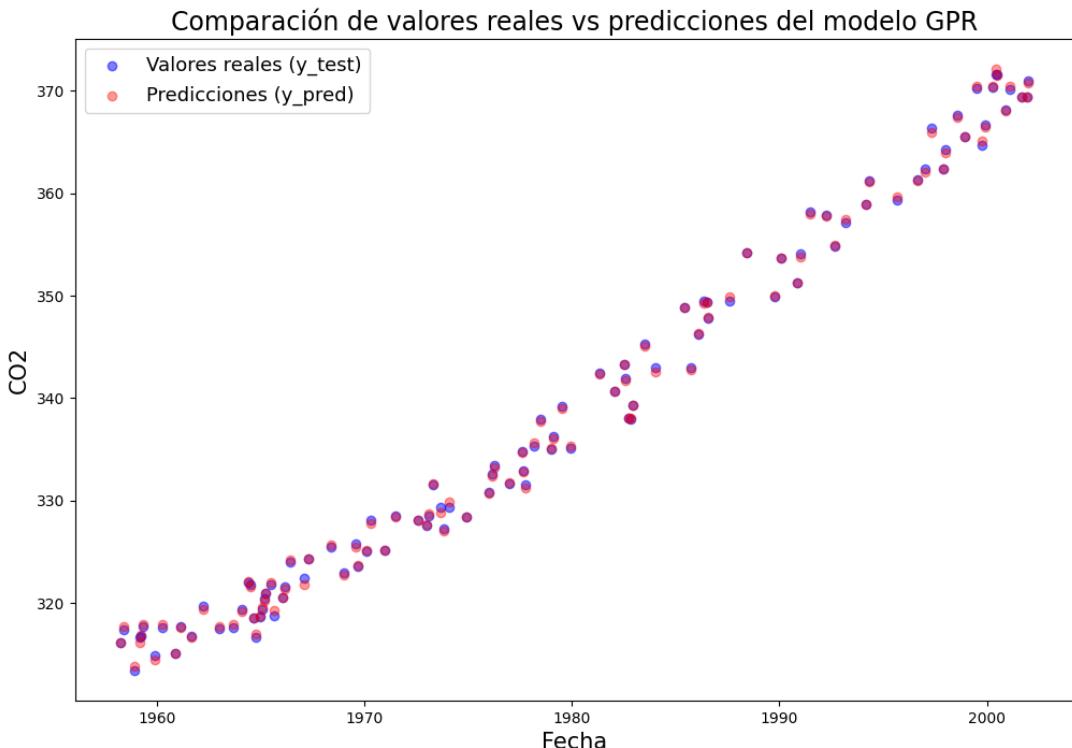


Figura 6.23.: Predicción en test.

En la Figura 6.24 vemos la predicción media del modelo junto a un intervalo de confianza del 95 %. Sin embargo, no se aprecia apenas la sombra anaranjada que muestra el intervalo de confianza, a diferencia del primer ejemplo que se realizó en el apartado 6.1. Esto es debido a que en este caso hay muy poca incertidumbre, la varianza es muy pequeña por lo que el intervalo de confianza ni se ve en el gráfico. Veamos en la Tabla 6.8 las predicciones de algunos puntos, donde se aprecia que la varianza es prácticamente nula:

Tabla 6.8.: Predicción (media) y desviación típica de la predicción

Predicción media	Desviación típica
368.0041	0.2433
324.2277	0.2396
313.7959	0.2587
336.0746	0.2531
321.3553	0.2556
347.9167	0.2532
327.6705	0.2535
342.7717	0.2385
339.2668	0.2610
365.5379	0.2412

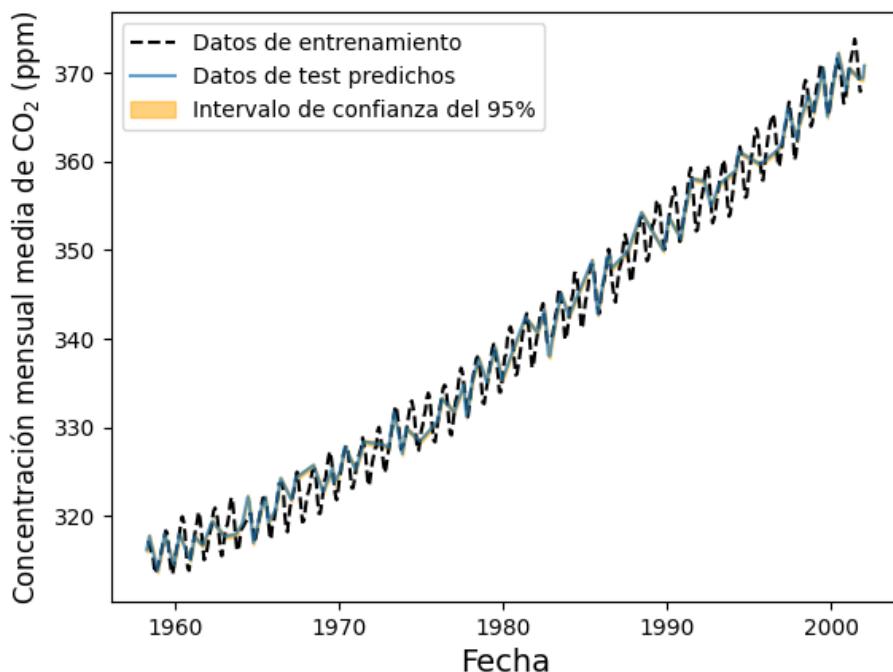


Figura 6.24.: Predicción media con intervalo de confianza del 95 %.

Veamos a continuación en la Tabla 6.9 los resultados de las métricas en test:

6. Experimentos

Modelo	MSE (test)	R^2 (test)
GaussianProcessRegressor	0.5697	0.9982
SVM	3.4021	0.9894
Gradient Boosting	1.079	0.9966

Tabla 6.9.: Resultados en el conjunto de test

Los modelos generalizan bien al conjunto de test, por lo que concluimos que a pesar de ajustarse de forma casi perfecta a los datos de entrenamiento, replican este comportamiento ante datos no vistos: no hay *overfitting*. SVM y Gradient Boosting incluso mejoran el MSE y el R^2 del entrenamiento.

Para realizar extrapolaciones más allá de 2001 realizo un nuevo conjunto de datos, que contenga 1000 instancias separadas proporcionalmente entre 1958 y la fecha de hoy. En la Figura 6.25 se ve el resultado de la predicción con el modelo *GaussianProcessRegressor*. Cabe destacar que a medida que se aleja del 2001 crece la incertidumbre y se empieza a ver la sombra naranja del intervalo de confianza del 95 %, cosa que no pasaba en los datos de entrenamiento como se comentó en la Figura 6.24.

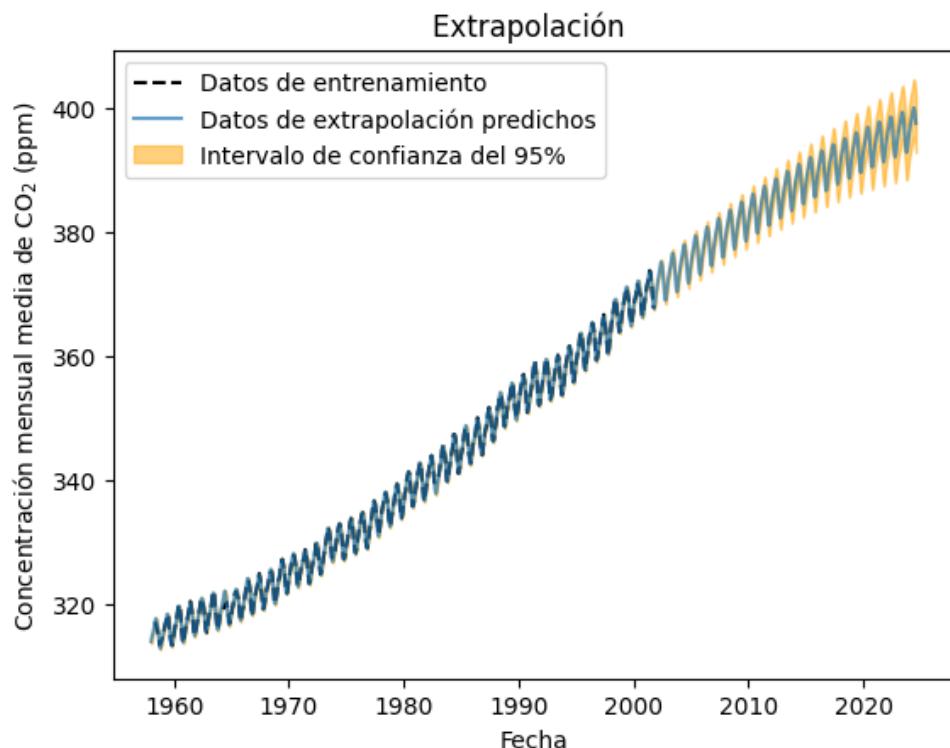


Figura 6.25.: Extrapolación hasta una fecha actual.

7. Conclusiones y Trabajos Futuros

En el presente trabajo se han estudiado y aplicado Procesos Gaussianos en Aprendizaje Automático Supervisado para distintos conjuntos de datos, tanto en problemas de regresión como de clasificación. En términos generales los Procesos Gaussianos han demostrado ser una buena herramienta con gran versatilidad, evidenciando una gran precisión. Se ha realizado una comparativa con diversos modelos (Gradient Boosting, SVM, Random Forest y KNN), observando que los Procesos Gaussianos no solo ofrecen resultados competitivos en cuanto a las métricas utilizadas se refiere, sino que también presentan una ventaja adicional al ser capaz de tratar la incertidumbre de forma explícita. Esta es una de las mayores fortalezas de los Procesos Gaussianos en Aprendizaje Automático Supervisado, y cuenta con una especial utilidad en contextos donde es crucial entender la confianza en los resultados obtenidos.

Sin embargo, también tienen sus inconvenientes. El principal problema al que se enfrenta es su limitada escalabilidad, como ya se ha podido ver en el apartado 6.2.1, donde con varias miles de instancias ya se han sufrido notables diferencias de tiempo de ejecución respecto a los otros modelos. Como se ha estudiado, los Procesos Gaussianos utilizados cuentan con una complejidad cúbica respecto al número de muestras de entrenamiento, por lo que desgraciadamente no es viable su uso con conjuntos de datos muy grandes. Para estas situaciones, existen los *Sparse Gaussian Processes*, que se mencionarán en la sección 7.2.

Se ha explorado en profundidad la Distribución Normal Multivariante y además se ha llevado a cabo una amplia investigación sobre la inferencia bayesiana. Este enfoque ha permitido ajustar las probabilidades a medida que se incorporan nuevas evidencias, mejorando así la capacidad para modelar y comprender fenómenos complejos en el contexto del Aprendizaje Automático.

Se ha evidenciado que la estrecha interacción entre la Ingeniería Informática y las Matemáticas produce resultados de gran interés y valor en el campo del Aprendizaje Automático. Estas disciplinas están íntimamente vinculadas y al unirse pueden enfrentar de manera más efectiva desafíos complejos.

7.1. Objetivos Alcanzados

Durante el desarrollo de este trabajo, se ha logrado satisfactoriamente cumplir con los objetivos establecidos:

- Se ha profundizado en el estudio y aplicación de los Procesos Gaussianos para resolver problemas de regresión y clasificación en Aprendizaje Automático Supervisado.
- Se ha explorado el enfoque basado en la inferencia Bayesiana, área apenas tratada durante la carrera pero crucial en el ámbito del Aprendizaje Automático y la Estadística.

7. Conclusiones y Trabajos Futuros

- Se han abordado, utilizando la aproximación de Laplace, las complejidades que surgen en los problemas de clasificación debido a la incompatibilidad de la verosimilitud y la distribución Gaussiana.
- Se ha proporcionado a lo largo del trabajo una explicación exhaustiva de los conceptos matemáticos empleados, contribuyendo así a una comprensión más profunda y rigurosa de los fundamentos teóricos subyacentes.

7.2. Futuros Trabajos

Para mejorar la escalabilidad, una exitosa línea de investigación son los Procesos Gaussianos Dispersos (*Sparse Gaussian Processes*) [HMG15] [HMFG15] [AEA⁺20]. Usan técnicas que permiten trabajar con conjuntos de datos grandes de manera más eficiente, manteniendo la capacidad de capturar la incertidumbre y las correlaciones inherentes a los Procesos Gaussianos. Destacan la Aproximación de Nyström, el uso de *Subset of Regressors* (SoR) y el *Projected Process Approximation* (PPA). Estas mejoras podrían ser estudiadas en una continuación de este trabajo.

Otra posible continuación de este trabajo podría centrarse en nuevas aproximaciones para los problemas de clasificación, más allá de la Aproximación de Laplace. Entre ellas se destacan la *Expectation Propagation* (EP) [RW⁺06] y la *Variational Inference* (VI) [HMG15].

También es interesante la integración de Procesos Gaussianos con Redes Neuronales Profundas para crear *Deep Gaussian Processes* [DL13] [SD17]. Esta combinación podría aprovechar las ventajas de ambas metodologías, permitiendo modelar relaciones complejas en los datos mientras se mantiene una representación explícita de la incertidumbre.

En resumen, este trabajo ha demostrado la efectividad de los Procesos Gaussianos en diversas tareas de Aprendizaje Automático, descubriendo sus fortalezas y limitaciones. Las futuras investigaciones deben centrarse en mejorar la escalabilidad y eficiencia, así como en explorar nuevas aplicaciones y metodologías avanzadas.

A. Apéndices

A.1. Factorización de Cholesky

La Factorización de Cholesky [RW⁺o6], ya estudiada durante la carrera, es una técnica muy común en álgebra lineal que lleva el nombre del matemático francés André-Louis Cholesky. Dada una matriz simétrica y definida positiva A , consiste en descomponerla en el producto de una matriz triangular inferior L y su traspuesta L^T ,

$$A = LL^T,$$

donde L recibe el nombre de *Factor de Cholesky*. Para calcular L primero se inicializa como una matriz de ceros y para cada $i = 1, \dots, n$:

$$L_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i-1} L_{ik}^2}.$$

Posteriormente, para cada $j = i + 1, \dots, n$:

$$L_{ji} = \frac{A_{ji} - \sum_{k=1}^{i-1} L_{jk}L_{ik}}{L_{ii}}.$$

Es un método usado principalmente en la resolución de sistemas lineales $Ax = b$, donde A sea una matriz simétrica y definida positiva. Primero, se resuelve el sistema triangular $Ly = b$ mediante sustitución hacia adelante y luego el sistema triangular $L^T x = y$ mediante sustitución hacia atrás. Utilizando el operador '\', la solución se escribirá como $x = L^T \backslash (L \backslash b)$, donde la notación $A \backslash b$ es el vector x que verifica $Ax = b$. Suponiendo que la matriz A es de tamaño $n \times n$, tanto la sustitución hacia delante como la sustitución hacia atrás son de complejidad $\mathcal{O}(n^2/2)$. El cálculo del Factor de Cholesky, L , es numéricamente estable y es de complejidad $\mathcal{O}(n^3/6)$.

Además, como consecuencia, el determinante de una matriz simétrica y definida positiva A se puede determinar como

$$|A| = \prod_{i=1}^n L_{ii}^2,$$

donde L es el Factor de Cholesky de A .

A.2. Método de Newton-Raphson

El método de Newton-Raphson [Ypm95], que ha sido estudiado durante la carrera, es una técnica iterativa para encontrar aproximaciones de las raíces de una función real $f(x)$. Es

A. Apéndices

uno de los métodos más utilizados en análisis numérico debido a su simplicidad y rapidez cuando se está cerca de la solución. Lleva el nombre de Isaac Newton y Joseph Raphson, quienes lo desarrollaron de forma independiente.

Dada una función $f : [a, b] \rightarrow \mathbb{R}$ diferenciable en $[a, b]$ y una estimación inicial x_0 de la raíz, el método de Newton-Raphson mejora esta estimación utilizando la siguiente fórmula iterativa:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},$$

donde $f'(x_n)$ es la derivada de f evaluada en x_n . La intuición detrás del método es usar la tangente a la curva $y = f(x)$ en el punto $(x_n, f(x_n))$ para aproximar la raíz. Es un proceso iterativo que continua hasta que la diferencia entre dos iteraciones sucesivas sea menor que un umbral de tolerancia predefinido, ϵ , es decir, $|x_{n+1} - x_n| < \epsilon$, o hasta que $|f(x_n)| < \epsilon$.

El método converge rápidamente cuando la estimación inicial x_0 está suficientemente cerca de la raíz verdadera y la función $f(x)$ es diferenciable y con una derivada no nula cerca de la raíz. Sin embargo, puede llegar a divergir cuando esto no se cumpla. En términos generales, la convergencia es cuadrática, lo que significa que el error se reduce aproximadamente al cuadrado en cada iteración.

Cada iteración del método de Newton-Raphson requiere calcular el valor de la función y su derivada en x_n . Si estas evaluaciones tienen una complejidad de $\mathcal{O}(1)$, entonces cada iteración también tiene una complejidad de $\mathcal{O}(1)$. Sin embargo, el número de iteraciones necesarias para alcanzar una precisión deseada depende de la función y de la estimación inicial, aunque típicamente es logarítmico en relación con la precisión requerida debido a la convergencia cuadrática del método.

A.3. Planificación Temporal

Debido a la alta carga de trabajo durante todo el curso (87 créditos), ha sido muy importante una correcta planificación desde septiembre para poder completar el trabajo sin contratiempos. Distinguiremos 4 etapas:

- Estudio inicial: Exploración general de las bibliografías aconsejadas para entender los conceptos básicos del tema y definir los objetivos del TFG.
- Investigación en profundidad: Análisis detallado de la literatura existente y desarrollo de un marco teórico sólido.
- Experimentación: Realización de experimentos y pruebas para la obtención de resultados y el entendimiento total del tema.
- Correcciones y desarrollo final: Revisión y corrección de la redacción y presentación del trabajo.

En la Figura A.1 se detalla la planificación seguida:

A.3. Planificación Temporal

	Septiembre	Octubre	Noviembre	Diciembre	Enero	Febrero	Marzo	Abril	Mayo	Junio	Julio
Estudio inicial											
Investigación en profundidad											
Experimentación											
Correcciones y Desarrollo Final											

Figura A.1.: Planificación temporal

Bibliografía

- [AAPV19] Stavros P Adam, Stamatios-Aggelos N Alexandropoulos, Panos M Pardalos, y Michael N Vrahatis. No free lunch theorem: A review. *Approximation and optimization: Algorithms, complexity and applications*, 2019.
- [AEA⁺20] Vincent Adam, Stefanos Eleftheriadis, Artem Artemev, Nicolas Durrande, y James Hensman. Doubly sparse variational gaussian processes. En *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020.
- [AMMIL12] Yaser S Abu-Mostafa, Malik Magdon-Ismail, y Hsuan-Tien Lin. *Learning from data*, volumen 4. AMLBook New York, 2012.
- [Ber18] Daniel Berrar. *Bayes' Theorem and Naive Bayes Classifier*. o1 2018.
- [BN06] Christopher M Bishop y Nasser M Nasrabadi. *Pattern recognition and machine learning*, volumen 4. Springer, 2006.
- [Ced23] Marcel Cedrez. Ordinary Kriging 5-Step Practical Guide. <https://discourse.online/>, 2023.
- [CM07] Vladimir Cherkassky y Filip M Mulier. *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007.
- [Dal93] Roger Daley. *Atmospheric data analysis*. Number 2. Cambridge university press, 1993.
- [DL13] Andreas Damianou y Neil D Lawrence. Deep gaussian processes. En *Artificial intelligence and statistics*. PMLR, 2013.
- [Fis36] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2), 1936.
- [Gal89] Francis Galton. *Natural inheritance*, volumen 42. Macmillan, 1889.
- [Gon09] Juan José González. Regresión a la media: Un fenómeno estadístico con historia y repercusión social. *Universidad de Las Palmas de Gran Canaria*, 10, 2009.
- [GS04] Iosif Illich Gikhman y Anatolii Vladimirovich Skorokhod. *The theory of stochastic processes I*. Springer Science & Business Media, 2004.
- [HMFG15] James Hensman, Alexander G Matthews, Maurizio Filippone, y Zoubin Ghahramani. Mcmc for variationally sparse gaussian processes. *Advances in neural information processing systems*, 28, 2015.
- [HMG15] James Hensman, Alexander Matthews, y Zoubin Ghahramani. Scalable variational gaussian process classification. En *Artificial Intelligence and Statistics*. PMLR, 2015.
- [HSSo8] Thomas Hofmann, Bernhard Schölkopf, y Alexander J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3), June 2008.
- [Kol41] Andrey Kolmogoroff. Interpolation und extrapolation von stationären zufälligen folgen. *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya*, 5(1), 1941.
- [Kri51] Daniel G Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6), 1951.
- [Mac92] David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3), 1992.
- [Mat60] B Matérn. Spatial variation, volume 49 of meddelanden från statens skogsforskningsinstitut. Stockholm: Statens Skogsforskningsinstitut, 1960.

Bibliografía

- [MGMB10] Mónica Martínez Gómez y Manuel Daniel Marí Benloch. La distribución normal. *Universitat Politècnica de València*, 2010.
- [Nea96] Radford M Neal. *Bayesian learning for neural networks*. Springer Science & Business Media, 1996.
- [O'H78] Anthony O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society: Series B (Methodological)*, 40(1):1–24, 1978.
- [OWoo] Manfred Opper y Ole Winther. Gaussian processes for classification: Mean-field algorithms. *Neural computation*, 12(11):2655–2684, 2000.
- [PDPF01] S Pértegas Díaz y S Pita Fernández. La distribución normal. *Cad Aten Primaria*, 8:268–274, 2001.
- [Pea20] Karl Pearson. Notes on the history of correlation. *Biometrika*, 13(1):25–45, 1920.
- [R⁺01] Irina Rish et al. An empirical study of the naive bayes classifier. En *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volumen 3. Seattle, WA, USA;, 2001.
- [Rin12] Luis Rincón. *Introducción a los procesos estocásticos*. UNAM, Facultad de Ciencias, 2012.
- [RW⁺06] Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian processes for machine learning*, volumen 1. Springer, 2006.
- [Sam59] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 1959.
- [SD17] Hugh Salimbeni y Marc Deisenroth. Doubly stochastic variational inference for deep gaussian processes. *Advances in neural information processing systems*, 30, 2017.
- [STCo4] John Shawe-Taylor y Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.
- [Ste65] M Abramowitz IA Stegun. Handbook of mathematical functions dover new york. 1965.
- [Ste99] ML Stein. Interpolation of spatial data. springer series in statistics, 1999.
- [SW89] WJ Sacks y TJ Welch. Mitchell, and hp wynn,“design and analysis of computer experiments,”. *Statistical Science*, 4(4):409–435, 1989.
- [SWX14] Alexander Y Sun, Dingbao Wang, y Xianli Xu. Monthly streamflow forecasting using gaussian process regression. *Journal of Hydrology*, 511, 2014.
- [TFPV92] Saul A Teukolsky, Brian P Flannery, WH Press, y W Vetterling. Numerical recipes in C. *SMR*, 693(1), 1992.
- [TT90] Yung Liang Tong y YL Tong. *Fundamental properties and sampling distributions of the multivariate normal distribution*. Springer, 1990.
- [UO30] George E Uhlenbeck y Leonard S Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.
- [Vap13] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.
- [VC⁺17] Carlos Villacampa Calvo et al. Procesos gaussianos para problemas de clasificación multiclase en conjuntos de datos grandes. Master's thesis, Universidad Autónoma de Madrid, 2017.
- [Wie49] Norbert Wiener. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. The MIT press, 1949.
- [Ypm95] Tjalling J Ypma. Historical development of the newton–raphson method. *SIAM review*, 37(4), 1995.