UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DEPARTAMENT DE MATEMÀTICA APLICADA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Matemática Aplicada

Clasificación de Series Temporales empleando Análisis Topológico de Datos

Trabajo Fin de Máster

Máster Universitario en Investigación Matemática

AUTOR/A: Jolin Rodrigo, Ignacio

Tutor/a: Conejero Casares, José Alberto

Cotutor/a externo: Falcó Montesinos, Antonio

CURSO ACADÉMICO: 2023/2024

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

VNIVERSITAT DE VALÈNCIA

Departamento de Matemática Aplicada

Trabajo Fin de Máster

Clasificación de Series Temporales empleando Análisis Topológico de Datos

Máster en investigación Matemática

Autor: Ignacio Jolín Rodrigo

Tutor: Conejero Casares, José Alberto

Cotutor externo: Antonio Falcó (UCH-CEU)

Curso 2023-2024

Contenido

Resumen

El análisis topológico de datos (TDA), se ha convertido en una de las ramas de matemáticas más activas en la era del análisis de datos en la que vivimos. En este trabajo estudiaremos los fundamentos de esta rama matemática e introduciremos los conceptos de imágenes y características persistentes, siendo este último una herramienta relativamente reciente que nos permite hacer uso del TDA de forma sencilla. En el último capítulo veremos como se puede detectar el Alzheimer con bastante precisión a partir de las características persistentes. Haremos uso además de varias herramientas de aprendizaje automático, como puede ser el algoritmo del bosque aleatorio o las redes neuronales convolucionales.

Abstract

Topological data analysis (TDA), has become one of the most active branches of mathematics in the age of data analysis in which we live. In this paper we will study the fundamentals of this mathematical topic and introduce the concepts of persistent images and persistent landscapes, tools that allows us to work with TDA in a simpler way. In the last chapter we will see how Alzheimer's can be detected with a high accuracy using persistent landscapes. We will also make use of various machine learning tools, such as the random forest algorithm or convolutional neural networks.

Introducción

La topología de datos es un campo emergente en la intersección de la topología algebraica y la ciencia de datos, que se ocupa de la forma, la estructura y la conectividad de los datos de alta dimensionalidad. Con el aumento masivo de datos generados en diversas disciplinas como la biología, la medicina, la física y las ciencias sociales, surge la necesidad de desarrollar métodos robustos para analizar datos complejos más allá de las técnicas estadísticas tradicionales. La topología de datos proporciona un marco matemático para capturar las características globales y locales de los datos, ofreciendo nuevas herramientas para la visualización, clasificación y comprensión de la estructura subyacente en los conjuntos de datos.

Este trabajo explora los principios fundamentales de la topología de datos, sus aplicaciones en diferentes áreas y cómo estas técnicas pueden ser utilizadas para resolver problemas complejos en el análisis de datos. Se discutirá la teoría básica de la homología persistente, la construcción de complejos simpliciales y las herramientas computacionales disponibles para aplicar estos conceptos a conjuntos de datos reales. A través de ejemplos y estudios de caso, se demostrará el poder de la topología de datos para extraer y comprender la información latente en estructuras de datos complejas, ofreciendo un enfoque innovador y potente para la ciencia de datos moderna.

La memoria se ha estructurado de tal forma que el primer capítulo introduzca los conceptos más fundamentales de álgebra, topología simplicial y homología simplicial, para poder definir más adelante los conceptos relacionados con la homología persistente de forma rigurosa. Además veremos como la homología es computable y la obtención de complejos simpliciales a partir de conjuntos de datos.

El segundo capítulo introduce al lector en el área de la homología persistente tras tener la base necesaria del capítulo anterior. Estudiaremos el concepto de filtración de series

temporales y como obtener su correspondiente código de barras. A través de esta última herramienta podremos definir las imágenes y características persistentes, donde veremos alguna de las propiedades de este último concepto.

A continuación, en el tercer capítulo, haremos una breve introducción del aprendizaje automático. Estudiaremos principalmente las redes neuronales convolucionales, con las cuales podemos clasificar imágenes, y el algoritmo de bosque aleatorio, herramienta que usaremos para la clasificación de vectores.

Finalmente en el cuarto capítulo, aplicaremos los conceptos estudiados en un problema real. Trataremos de desarrollar modelos capaces de detectar Alzheimer asociando a cada paciente una imagen o característica persistente, que habremos obtenido tras el análisis de las señales EEG. Tras desarrollar varios modelos obtenemos uno en concreto con gran precisión que hace uso de las características persistentes, resaltando la utilidad de esta herramienta.

Capítulo 1

Preliminares

En este capítulo introducimos los conceptos más fundamentales de álgebra, topología simplicial y homología simplicial, para poder definir más adelante los conceptos relacionados con la homología persistente de forma rigurosa. Además veremos como la homología es computable y la obtención de complejos simpliciales a partir de conjuntos de datos.

1.1 Álgebra

A continuación introduciremos algunos conceptos y resultados de álgebra de los que haremos uso más adelante.

Definición 1.1. Sea $\mathbb{F}$ un anillo conmutativo. Un $\mathbb{F}$-*módulo* es una dupla (M, μ), donde M es un grupo abeliano y μ es una aplicación de la forma

$$\mu : \mathbb{F} \times M \longrightarrow M,$$

que para $x, y \in M$ y $a, b \in \mathbb{F}$ satisface las siguientes propiedades:

1. $\mu(a, x + y) = \mu(a, x) + \mu(a, y)$.
2. $\mu(a + b, x) = \mu(a, x) + \mu(b, x)$.
3. $\mu(1, x) = x$.

A la aplicación $\mu(a, x)$ la denotaremos como $a \cdot x$.

Definición 1.2. Una *sucesión exacta* es un diagrama de $\mathbb{F}$-módulos y homomorfismos de la forma

$$\cdots \longrightarrow M_{n-1} \xrightarrow{f_{n-1}} M_n \xrightarrow{f_n} M_{n+1} \longrightarrow \cdots,$$

donde $\mathrm{Im}(f_{n-1}) = \mathrm{Ker}(f_n)$ para todo n.

Proposición 1.3. La sucesión de la forma

$$0 \longrightarrow M_1 \xrightarrow{f_1} M_2 \xrightarrow{f_2} M_3 \longrightarrow 0$$

es exacta si y solo sí, f_1 es inyectiva, f_2 sobreyectiva y $f_2 \circ f_1 = 0$.

A una sucesión que tenga esa forma se la llama *sucesión exacta corta*.

Definición 1.4. Una sucesión exacta corta

$$0 \longrightarrow M_1 \xrightarrow{f_1} M_2 \xrightarrow{f_2} M_3 \longrightarrow 0$$

se dice *escindible* cuando existe un homomorfismo $\rho : M_3 \longrightarrow M_2$ de tal forma que $f_2 \circ \rho = \mathrm{Id}_{M_3}$.

Proposición 1.5. Toda sucesión exacta corta

$$0 \longrightarrow M_1 \xrightarrow{f_1} M_2 \xrightarrow{f_2} M_3 \longrightarrow 0$$

donde M_3 es libre, es escindible.

Proposición 1.6. Sea

$$0 \longrightarrow M_1 \xrightarrow{f_1} M_2 \xrightarrow{f_2} M_3 \longrightarrow 0$$

una sucesión exacta corta escindible. Se tiene que $M_2 = f_1(M_1) \oplus M$, para cierto submódulo de M_2 isomorfo a M_3.

Demostración. Dado que es escindible tenemos el siguiente diagrama

$$M_3 \xrightarrow{\rho} M_2 \xrightarrow{f_2} M_3 \qquad (\mathrm{Id}: M_3 \to M_3)$$

Entonces si $x \in M_2$, se tiene que $x = \rho(f_2(x)) + (x - \rho(f_2(x)))$, por lo que $M_2 = \mathrm{Im}(\rho) \oplus \mathrm{Ker}(f_2)$. Como es una sucesión exacta, tenemos entonces que $M_2 = M \oplus f_1(M_1)$, donde $M = \mathrm{Im}(\rho)$. □

Teorema 1.7 (Primer teorema de isomorfía)**.** Sea $f : G \longrightarrow H$ un homomorfismo de grupos. Entonces

$$\frac{G}{\mathrm{Ker}(f)} \cong \mathrm{Im}(f).$$

Teorema 1.8 (Tercer teorema de isomorfía)**.** Sean N y H subgrupos normales de un grupo G con $N \subseteq H$. Entonces

$$\frac{G/N}{H/N} \cong \frac{G}{H}.$$

1.2 Topología simplicial

Definición 1.9. Sean $a_0, a_1, \ldots, a_n$ puntos de $\mathbb{R}^n$. Diremos que son *afínmente independientes* si los vectores $a_1 - a_0, a_2 - a_0, \ldots, a_n - a_0$ son linealmente independientes.

Definición 1.10. Dados $n+1$ puntos afínmente independientes $a_0, a_1, \ldots, a_n$, llamaremos *n-símplice* al conjunto convexo:

$$\sigma := \{x \in \mathbb{R}^n : x = \sum_{k=0}^{n} \lambda_k(x) a_k \text{ con } \sum_{k=0}^{n} \lambda_k(x) = 1 \text{ y } \lambda_k(x) \geq 0 \text{ para todo } k\}$$

Los puntos $a_0, \ldots, a_n$ serán los *vértices de* σ, que escribiremos como $V(\sigma) := \{a_0, \ldots, a_n\}$. Los símplices engendrados por subconjuntos no vacíos de $\{a_0, \ldots, a_n\}$ diremos que son *caras* de σ. Si τ es una cara de σ diferente de este, la denominaremos *cara propia* y lo denotaremos como $\tau < \sigma$. Los escalares $\lambda_k(x)$ están unívocamente determinados por el punto x, y se denominan *coordenadas baricéntricas de* x con respecto a los puntos a_k. En cuanto a la notación, los símplices los denotaremos por $\sigma :=< a_0, \ldots, a_n >$.

Definición 1.11. Sea σ un símplice. A la unión de las caras propias de σ la denominaremos la *frontera (geométrica)* de σ, que denotaremos por $\mathrm{Fr}^{\mathrm{g}}\,\sigma$. El *interior (geométrico)* de σ es el subconjunto de σ definido por la ecuación $\mathrm{Int}^{\mathrm{g}}\,\sigma := \sigma - \mathrm{Fr}^{\mathrm{g}}\,\sigma$.

Definición 1.12. Un *complejo simplicial* es un conjunto finito, $\mathcal{K}$, de símplices de $\mathbb{R}^n$ que verifican estas dos condiciones:

1. Si $\tau < \sigma$ y $\sigma \in \mathcal{K}$, entonces $\tau \in \mathcal{K}$.

2. La intersección de dos símplices de $\mathcal{K}$, o bien es vacía, o bien es una cara común de estos.

Definición 1.13. Sea $\mathcal{K}$ un complejo simplicial de $\mathbb{R}^n$. Denominaremos *poliedro de* $\mathcal{K}$ al subespacio de $\mathbb{R}^n$ formado por la unión de los símplices de $\mathcal{K}$, y lo denotaremos por $|\mathcal{K}|$, es decir,

$$|\mathcal{K}| := \bigcup_{\sigma \in \mathcal{K}} \sigma \subseteq \mathbb{R}^n.$$

Definición 1.14. Sean los conjuntos $\mathcal{K}$ y $\mathcal{L}$ complejos simpliciales. Diremos que una aplicación $\varphi : V(\mathcal{K}) \longrightarrow V(\mathcal{L})$ es una *aplicación simplicial* si satisface la siguiente propiedad: Si $\sigma :=< a_0, \ldots, a_n >$ es un símplice de $\mathcal{K}$, entonces $\{\varphi(a_0), \ldots, \varphi(a_n)\}$ es el conjunto de vértices de un símplice de $\mathcal{L}$.

La aplicación φ también induce una aplicación continua $|\mathcal{K}| \longrightarrow |\mathcal{L}|$, que seguiremos denotando con φ, de la siguiente forma: Si $x \in< a_0, \ldots, a_n >\in \mathcal{K}$ y $\lambda_0(x), \ldots, \lambda_n(x)$ son las coordenadas baricéntricas de x con respecto a los puntos $a_0, \ldots, a_n$, entonces

$$\varphi(x) = \varphi\left(\sum_{i=0}^{n} \lambda_i(x) \cdot a_i\right) = \sum_{i=0}^{n} \lambda_i(x) \cdot \varphi(a_i).$$

Podemos interpretar también la aplicación φ como una aplicación entre los complejos simpliciales $\mathcal{K}$ y $\mathcal{L}$, es decir, $\varphi(< a_0, \ldots, a_n >)$ es el símplice de $\mathcal{L}$ cuyos vértices son $\varphi(a_0), \ldots, \varphi(a_n)$ (obsérvese que algunos de los puntos $\varphi(a_0), \ldots, \varphi(a_n)$ pueden estar repetidos).

Definición 1.15. Sea V un conjunto. Un *complejo abstracto* $\mathcal{A}$ es una colección no vacía de partes finitas de V que satisface las siguientes condiciones:

1. $\mathcal{A}$ contiene todos los conjuntos unitarios de V.
2. Dado $\Sigma \in \mathcal{A}$, todo subconjunto de Σ pertenece a $\mathcal{A}$.

Ejemplo 1.16. Sea X un conjunto finito de puntos. Es fácil ver que $\mathcal{P}(X)$ es un complejo simplicial abstracto.

Proposición 1.17. Todo complejo simplicial geométrico tiene un complejo simplicial abstracto asociado, dado por una colección de subconjuntos de los vértices de $\mathcal{K}$.

Definición 1.18. Una *realización geométrica* de un complejo abstracto $\mathcal{A}$, es un complejo simplicial $\mathcal{K}$ cuyo correspondiente complejo abstracto es isomorfo a $\mathcal{A}$.

Teorema 1.19. Para todo complejo simplicial abstracto $\mathcal{A}$, existe una realización geométrica.

1.3 Homología simplicial

En esta sección vamos a introducir el concepto de homoogía, el cual es un invariante en topología que tiene gran importancia en la topología de datos.

Definición 1.20. Sea σ un símplice, con $\sigma :=< a_0, \ldots, a_n >$. Se define la siguiente relación de equivalencia sobre el conjunto de las ordenaciones de los vértices de σ:

$$< a_o, \ldots, a_n > \sim < a_{\pi(0)}, \ldots, a_{\pi(n)} >$$

si π es una permutación par. Esta relación da lugar a dos clases de equivalencia, cada una llamada *orientación* de σ. Cuando escojamos una orientación diremos que σ es un *símplice orientado*. La notación del símplice σ orientado será $[a_0, \ldots, a_n]$.

Definición 1.21. Se define el $\mathbb{F}$-*módulo de n-cadenas simpliciales orientadas* y se denota por $C_n(\mathcal{K}; \mathbb{F})$, al cociente del $\mathbb{F}$-módulo libre generado por todos los n-símplices σ del complejo $\mathcal{K}$, por el submódulo generado por $\sigma_1 + \sigma_2$, siendo estas las orientaciones de σ. A los elementos de $C_n(\mathcal{K}; \mathbb{F})$ los llamaremos *n-cadenas simpliciales orientadas.*

Tenemos entonces que los elementos de $C_n(\mathcal{K}; \mathbb{F})$ son combinaciones lineales de n-símplices orientados con coeficientes del anillo $\mathbb{F}$. Dado $\alpha \cdot [a_0, \ldots, a_n] \in C_n(\mathcal{K}; \mathbb{F})$ con $\alpha \in \mathbb{F}$, tenemos que si $[a_{\pi(0)}, \ldots, a_{\pi(n)}]$ es la otra orientación del símplice, entonces en $C_n(\mathcal{K}; \mathbb{F})$ se tiene la igualdad:

$$\alpha \cdot [a_0, \ldots, a_n] = \alpha \cdot (-[a_{\pi(0)}, \ldots, a_{\pi(n)}]).$$

Definición 1.22. Llamamos *operador borde de orden n* al homomorfismo de $\mathbb{F}$-módulos

$$\partial_n : C_n(\mathcal{K}; \mathbb{F}) \longrightarrow C_{n-1}(\mathcal{K}; \mathbb{F})$$

dado por la extensión lineal de

$$\partial_n([a_0, \ldots, a_n]) = \sum_{i=0}^{n} (-1)^i \cdot [a_0, \ldots, \hat{a}_i, \ldots, a_n],$$

donde $[a_0, \ldots, \hat{a}_i, \ldots, a_n]$ representa el $(n-1)$-símplice orientado obtenido al eliminar el vértice a_i de $[a_0, \ldots, a_n]$.

Proposición 1.23. El operador borde está bien definido, es decir, $\partial_n(-\sigma) = -\partial_n(\sigma)$.

Demostración: Vamos a comparar las expresiones

$$\partial_n([a_0,\ldots,a_j,a_{j+1},\ldots,a_n]) \quad \text{y} \quad \partial_n([a_0,\ldots,a_{j+1},a_j,\ldots,a_n]).$$

En primer lugar, si tomamos un término de la suma tal que el índice del vértice eliminado satisface que $i \notin \{j, j+1\}$, entonces los términos son iguales ya que

$$[a_0,\ldots,\hat{a}_i,\ldots,a_j,a_{j+1},\ldots,a_n] = -[a_0,\ldots,\hat{a}_i,\ldots,a_{j+1},a_j,\ldots,a_n].$$

En el caso en el que el índice de la suma es $i = j$ e $i = j+1$, obtenemos para la primera expresión

$$(-1)^j \cdot [a_0,\ldots,a_{j-1},a_{j+1},a_{j+2},\ldots,a_n] + (-1)^{j+1} \cdot [a_0,\ldots,a_{j-1},a_j,a_{j+2},\ldots,a_n]. \quad (1.1)$$

En el segundo tenemos la siguiente suma:

$$(-1)^j \cdot [a_0,\ldots,a_{j-1},a_j,a_{j+2},\ldots,a_n] + (-1)^{j+1} \cdot [a_0,\ldots,a_{j-1},a_{j+1},a_{j+2},\ldots,a_n]. \quad (1.2)$$

Obtenemos efectivamente que las expresiones 1.1 y 1.2 difieren en un signo, por lo que se cumple la igualdad del enunciado. □

Proposición 1.24. El operador borde verifica que $\partial_n \circ \partial_{n+1} = 0$ para todo n.

Demostración: Se tiene el siguiente cálculo:

$$\partial_n \circ \partial_{n+1} \cdot [a_0,\ldots,a_n,a_{n+1}]) = \partial_n \left(\sum_{i=0}^{n+1} (-1)^i [a_0,\ldots,\hat{a}_i,\ldots,a_n,a_{n+1}] \right) =$$

$$= \sum_{i=0}^{n+1} (-1)^i \left(\sum_{j=0}^{i-1} (-1)^j [a_0,\ldots,\hat{a}_j,\ldots,\hat{a}_i,\ldots,a_{n+1}] + \right.$$

$$\left. + \sum_{j=i+1}^{n+1} (-1)^{j-1} [a_0,\ldots,\hat{a}_j,\ldots,\hat{a}_i,\ldots,a_{n+1}] \right) = 0.$$

Esto es porque los términos de la suma se anulan por parejas. □

Definición 1.25. Una *complejo de cadenas* $\mathcal{C}$ es una secuencia

$$\cdots \xrightarrow{\partial_{n+2}} C_{n+1} \xrightarrow{\partial_{n+1}} C_n \xrightarrow{\partial_n} C_{p-1} \cdots,$$

donde C_n es un $\mathbb{F}$-módulo de n-cadenas y ∂_n el operador borde para todo $n \in \mathbb{N}$. Lo denotaremos por $\{C_n, \partial_n\}$.

Definición 1.26. Llamaremos *n-ésimo $\mathbb{F}$-módulo de ciclos* a $Z_n(\mathcal{K};\mathbb{F}) := \mathrm{Ker}(\partial_n)$, y *n-ésimo $\mathbb{F}$-módulo de bordes* a $B_n(\mathcal{K};\mathbb{F}) := \mathrm{Im}(\partial_{n+1})$.

Proposición 1.27. Para todo complejo simplicial $\mathcal{K}$ y para todo n se satisface que

$$B_n(\mathcal{K};\mathbb{F}) \subseteq Z_n(\mathcal{K};\mathbb{F}).$$

Definición 1.28. Llamaremos *n-ésimo grupo de homología simplicial del complejo* $\mathcal{K}$ al cociente:

$$H_n(\mathcal{K};\mathbb{F}) := \frac{Z_n(\mathcal{K};\mathbb{F})}{B_n(\mathcal{K};\mathbb{F})}.$$

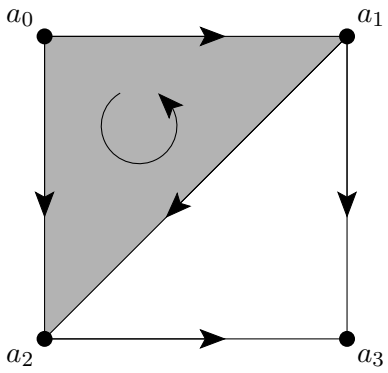

Figura 1.1: Realización geométrica de $\mathcal{K}$.

Ejemplo 1.29. Consideramos el complejo $\mathcal{K}$ de la figura 1.1. Vamos a calcular su primer grupo de homología, es decir, $H_1(\mathcal{K};\mathbb{Z})$. En primer lugar, es fácil ver que $C_0(\mathcal{K};\mathbb{Z})$ está generado por el conjunto de vértices $\{a_0, a_1, a_2, a_3\}$, $C_1(\mathcal{K};\mathbb{Z})$ tiene por base el conjunto $\{[a_0, a_1], [a_0, a_2], [a_1, a_2], [a_1, a_3], [a_2, a_3]\}$ y $C_2(\mathcal{K};\mathbb{Z})$ es generado por $\{[a_0, a_1, a_2]\}$.

A continuación hay que calcular $Z_1(\mathcal{K};\mathbb{Z})$, es decir, $\mathrm{Ker}(\partial_1)$. Para ello, resolvemos la siguiente ecuación

$$\partial_1(n_1 \cdot [a_0, a_1] + n_2 \cdot [a_0, a_2] + n_3 \cdot [a_1, a_2] + n_4 \cdot [a_1, a_3] + n_5 \cdot [a_2, a_3]) = 0,$$

obteniendo que una 1-cadena es un 1 ciclo si, y solo si, se tiene que $n_1 + n_2 = 0$, $n_4 + n_5 = 0$ y $n_1 = n_3 + n_4$. Tenemos por lo tanto que $Z_1(\mathcal{K};\mathbb{Z})$ es el grupo abeliano libre de grado 2 generado por $\{[a_0, a_1] + [a_1, a_2] - [a_0, a_2], [a_1, a_2] + [a_2, a_3] - [a_1, a_3]\}$.

Por otro lado, para calcular $B_1(\mathcal{K};\mathbb{Z})$ vamos a hallar $\mathrm{Im}(\partial_2)$, que en este caso es

$$\partial_2(n \cdot [a_0, a_1, a_2]) = n \cdot ([a_1, a_2] - [a_0, a_2] + [a_0, a_1]),$$

es decir, el grupo abeliano libre de grado 1 generado por $\{[a_1, a_2] - [a_0, a_2] + [a_0, a_1]\}$.

Finalmente tenemos que el primer grupo de homología de $\mathcal{K}$ es:

$$H_1(\mathcal{K}; \mathbb{Z}) = \frac{Z_1(\mathcal{K}; \mathbb{Z})}{B_1(\mathcal{K}; \mathbb{Z})} \cong \mathbb{Z}.$$

Proposición 1.30. Sean $\mathcal{K}$ y $\mathcal{L}$ complejos simpliciales y sea

$$\varphi : \mathcal{K} \longrightarrow \mathcal{L}$$

una aplicación simplicial. Para todo $n \geq 0$, la aplicación φ induce un homomorfismo de complejos de cadenas

$$C_n(\varphi) : C_n(\mathcal{K}; \mathbb{F}) \longrightarrow C_n(\mathcal{L}; \mathbb{F})$$

definido por la extensión lineal de

$$C_n(\varphi)([a_0, \ldots, a_n]) = \begin{cases} [\varphi(a_0), \ldots, \varphi(a_n)] & \text{si los vértices son distintos dos a dos,} \\ 0 & \text{en caso contrario.} \end{cases}$$

Corolario 1.31. Para todo $n \geq 0$, toda aplicación simplicial $\varphi : \mathcal{K} \longrightarrow \mathcal{L}$ induce un homomorfismo

$$H_n(\varphi) : H_n(\mathcal{K}; \mathbb{F}) \longrightarrow H_n(\mathcal{L}; \mathbb{F}).$$

1.4 Computabilidad de la homología simplicial

La homología simplicial es de gran utilidad ya que se pueden computar los grupos de homología en un tiempo de ejecución razonable. Para ello es necesario estudiar las matrices asociadas a los operadores bordes que posibilitan el desarrollo de un algoritmo para calcular los grupos de homología de cualquier grado.

En primer lugar, veamos algunos concepto y resultados de álgebra que utilizaremos más adelante.

Definición 1.32. Sean G y G' dos $\mathbb{F}$-módulos libres y finitamente generados que tienen como base $\{e_1, \ldots, e_n\}$ y $\{e'_1, \ldots, e'_m\}$, respectivamente. Si $f : G \longrightarrow G'$ es un homomorfismo de grupos, entonces

$$f(e_j) = \sum_{i=1}^{m} \lambda_{ij} e'_i,$$

para ciertos $\lambda_{ij} \in \mathbb{F}$, donde (λ_{ij}) es la *matriz asociada* a la aplicación f.

En el caso de los operadores bordes, si $\mathbb{F} = \mathbb{Z}$ entonces los coeficientes de la matriz asociada a ∂_n están en $\{-1, 1, 0\}$.

Teorema 1.33 (Teorema de diagonalización de Smith)**.** Sea $\mathbb{F}$ un dominio de ideales principales y sean G y G' dos $\mathbb{F}$-módulos libres de dimensión n y m, respectivamente. Para todo homomorfismo de grupos $f : G \longrightarrow G'$, si $M \in \mathfrak{M}_{m\times n}(\mathbb{F})$ es la matriz asociada a f, entonces M es equivalente a una matriz diagonal de la forma

$$D = \left[\begin{array}{ccc|ccc} \alpha_1 & & 0 & & & \\ & \ddots & & & 0 & \\ 0 & & \alpha_l & & & \\ \hline & & & 0 & & \\ & 0 & & & \ddots & \\ & & & & & 0 \end{array}\right]$$

donde $\alpha_i \geq 1$ y $\alpha_1|\alpha_2|\ldots|\alpha_l$, esto es, α_i divide a α_{i+1} para todo $0 \leq i < l$. A esta matriz la llamaremos *forma normal de Smith* de M.

Antes de la demostración, vamos a decir que las *operaciones elementales por filas* de una matriz con coeficientes en $\mathbb{F}$ son las siguientes:

1. Intercambiar la fila i por la fila k.
2. Multiplicar la fila i por -1
3. Sumar a la fila i la fila k multiplicada por un entero n, donde $i \neq k$.

De forma similar definimos las *operaciones elementales por columnas*, las cuales son las mismas operaciones pero con columnas en vez de con filas.

Demostración. Vamos a probar que M es equivalente a una matriz de la forma

$$\left[\begin{array}{c|c} d & 0 \\ \hline 0 & A \end{array}\right],$$

con $d \in \mathbb{F}$ y $A \in \mathfrak{M}_{(m-1)\times(n-1)}(\mathbb{F})$, donde d divide a todos los elementos de A.

Para obtener el resultado, llevamos a cabo el siguiente algoritmo:

1. En primer lugar, tomamos la primera columna y movemos el elemento con menor norma de esta a la primera fila, al intercambiar la primera fila por la de este. Lo denotaremos por d. Tras esto, hacemos la división euclídea a todos los elementos de la primera columna con d, obteniendo $a_{i1} = n_j \cdot d + r_i$, donde $|d| > |r_i| \geq 0$. Sumamos a la fila i la primera fila multiplicada por $-n_i$, obteniendo como pivote r_i. Si $r_i \neq 0$ permutamos esta fila con la primera, la cual tiene en la primera columna a r_i. Repetimos este proceso hasta que finalmente la primera columna son todo ceros menos el primer elemento.

2. En este paso, repetimos el mismo proceso pero con las columnas, sin embargo, si hacemos una permutación con la primera fila, tenemos que volver al primer paso hasta obtener nuevamente la primera columna con todo ceros menos el pivote. Repetimos este proceso hasta que la primera fila y primera columna tengan todos los elementos nulos menos el pivote d.

3. Por último, tenemos que comprobar que d divide a todos los elementos restantes. Si es así ya hemos terminado. En caso contrario, si existe un elemento a_{ij} al que no divide d, realizamos de nuevo la división euclídea con la que obtenemos $a_{ij} = n_{ij} \cdot d + r_{ij}$. Entonces sumamos la fila i a la primera, obteniendo la fila

$$[d, a_{i2}, \ldots, a_{ij}, \ldots, a_{jn}],$$

a la cual la volvemos a aplicar los primeros pasos del algoritmo. Repetimos este proceso hasta que d divida a todos los elementos de la matriz, obteniendo la forma deseada.

Tras esto, repetimos el algoritmo de forma inductiva a la submatriz A obtenida, hasta que finalmente obtenemos la forma normal de Smith. □

A continuación vamos a ver el teorema de estructura de módulos finitamente generados sobre dominios de ideales, que nos va a ser de gran utilidad para poder calcular el grupo de homología. Es consecuencia del teorema de diagonalización de Smith y dice lo siguiente:

Teorema 1.34. Sea $\mathbb{F}$ un dominio de ideales principales. Supongamos que G es un $\mathbb{F}$-módulo finitamente generado. Entonces

$$G \cong \mathbb{F}^r \oplus \left(\frac{\mathbb{F}}{(\alpha_1)}, \ldots, \frac{\mathbb{F}}{(\alpha_l)} \right),$$

donde los r primeros sumandos corresponden a la *parte libre* de G que denotaremos como L_G, mientras que a los l sumandos restantes serán la *parte de torsión* de G, que denotaremos como T_G. Además a los términos no nulos $\alpha_1, \ldots, \alpha_l \in \mathbb{F}$, que llamaremos *coeficientes de torsión* de G, satisfacen que $\alpha_1|\alpha_2|\ldots|\alpha_l$.

Demostración. Como G es finitamente generado, supongamos que el conjunto $\{x_1, \ldots, x_n\}$ genera a G. Sea $\{e_1, \ldots, e_n\}$ la base canónica de $\mathbb{F}^n$ con la cual podemos definir el siguiente homomorfismo:

$$\begin{aligned} \varphi : \mathbb{R}^n &\longrightarrow G \\ e_i &\longmapsto \varphi(e_i) = x_i. \end{aligned}$$

Es una aplicación sobreyectiva, luego por el primer teorema de isomorfía se tiene la equivalencia

$$\frac{\mathbb{F}^n}{\mathrm{Ker}(\varphi)} \cong \mathrm{Im}(\varphi) = G.$$

Dado que $\mathrm{Ker}(\varphi)$ es un submódulo de $\mathbb{F}^n$ de rango libre, entonces tiene un conjunto de generadosres $\{f_1, \ldots, f_m\}$ definidos en función de la base $\{e_1, \ldots, e_n\}$ de la siguiente forma:

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1n} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{m1} & \lambda_{m2} & \cdots & \lambda_{mn} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix},$$

donde $A := (\lambda_{ij}) \in \mathfrak{M}_{m\times n}(\mathbb{F})$. Dado que $\mathbb{F}$ es un dominio de ideales principales, podemos aplicar el teorema de diagonalización de Smith a la matriz A, es decir, obtenemos dos matrices invertibles $P \in \mathfrak{M}_{m\times m}(\mathbb{F})$ y $Q \in \mathfrak{M}_{n\times n}(\mathbb{F})$ tales que

$$D := \left[\begin{array}{ccc|ccc} \alpha_1 & & 0 & & & \\ & \ddots & & & 0 & \\ 0 & & \alpha_l & & & \\ \hline & & & 0 & & \\ & 0 & & & \ddots & \\ & & & & & 0 \end{array}\right] = P^{-1} \cdot A \cdot Q.$$

A continuación definimos los conjuntos

$$\begin{bmatrix} f'_1 \\ f'_2 \\ \vdots \\ f'_m \end{bmatrix} := P \cdot \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \text{ y } \begin{bmatrix} e'_1 \\ e'_2 \\ \vdots \\ e'_n \end{bmatrix} := Q \cdot \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix},$$

donde $\{f'_1, \ldots, f'_m\}$ genera $\mathrm{Ker}(\varphi)$ y $\{e'_1, \ldots, e'_n\}$ es una base de $\mathbb{F}^n$. Esto implica que podemos expresar el núcleo de φ de la siguiente forma

$$\mathrm{Ker}(\varphi) = (\alpha_1) \oplus \cdots \oplus (\alpha_l),$$

por lo que

$$\frac{\mathbb{F}^n}{\mathrm{Ker}(\varphi)} = \frac{\mathbb{F}^n}{(\alpha_1) \oplus \cdots \oplus (\alpha_l)} = \mathbb{F}^r \oplus \left(\frac{\mathbb{F}}{(\alpha_1)}, \ldots, \frac{\mathbb{F}}{(\alpha_l)} \right) \cong G.$$

□

Definición 1.35. Sea $C_n(\mathcal{K}; \mathbb{F})$ un $\mathbb{F}$-módulo de n-cadenas simpliciales ordenadas. Llamamos *n-ésimo $\mathbb{F}$-módulo de bordes débiles* al siguiente submódulo de $C_n(\mathcal{K}; \mathbb{F})$:

$$W_n := \{c_n \in C_n(\mathcal{K}; \mathbb{F}) : \exists \lambda \in \mathbb{F} \setminus \{0\}, \text{ tal que } \lambda \cdot c_n \in B_n\}.$$

Proposición 1.36. Sea $C_n(\mathcal{K}; \mathbb{F})$ un $\mathbb{F}$-módulo de n-cadenas simpliciales ordenadas. Entonces, se satisface la siguiente cadena de inclusiones para todo n:

$$B_n \subseteq W_n \subseteq Z_n \subseteq C_n.$$

Demostración. La primera inclusión es obvia, ya que basta tomar $\lambda = 1$.

En la segunda inclusión, dado que $C_n(\mathcal{K}; \mathbb{F})$ es un módulo libre, para $c_n \in W_n$ existe $c_{n+1} \in C_{n+1}(\mathcal{K}; \mathbb{F})$ tal que $\partial_{n+1}(c_{n+1}) = \lambda \cdot c_n$ con $\lambda \in \mathbb{F}$, por lo que

$$\partial_n(c_n) = \partial_n(\lambda^{-1} \cdot \partial_{n+1}(c_{n+1})) = \lambda^{-1} \cdot \partial_n(\partial_{n+1}(c_{n+1})) = 0.$$

La tercera inclusión es consecuencia directa de la definición de Z_n. □

Teorema 1.37. Sea $\{C_n, \partial_n\}$ un complejo de cadenas. Entonces para cada n existen submódulos $U_n, V_n \subseteq C_n$ tales que

$$Z_n = V_n \oplus W_n \quad \text{y} \quad C_n = U_n \oplus V_n \oplus W_n,$$

donde Z_n es el $\mathbb{F}$-módulo de ciclos y W_n es el $\mathbb{F}$-módulo de bordes débiles. Se tiene además que

$$L_{H_n} \cong \frac{Z_n}{W_n} \quad \text{y} \quad T_{H_n} \cong \frac{W_n}{B_n}.$$

Demostración. En primer lugar vamos a probar que $T_{H_n} \cong \dfrac{W_n}{B_n}$. Sea $c_n + B_n \in T_{H_n}$, entonces existe $\lambda \in \mathbb{F}$ tal que $\lambda \cdot c_n + B_n = 0$ en T_{H_n}. Esto es equivalente a que $c_n \in W_n$, luego $T_{H_n} \cong \dfrac{W_n}{B_n}$.

En cuanto a la equivalencia $L_{H_n} \cong \dfrac{Z_n}{W_n}$ se tiene lo siguiente:

$$L_{H_n} \cong \frac{H_n}{T_{H_n}} \cong \frac{Z_n/B_n}{W_n/B_n} \cong \frac{Z_n}{W_n},$$

donde la primera equivalencia es consecuencia del teorema de estructura 1.34 y la última se debe al tercer teorema de isomorfía.

Sea ahora $\{e_1 + W_n, \ldots, e_k + W_n\}$ una base de $\dfrac{Z_n}{W_n}$ y sea $\{d_1, \ldots, d_p\}$ una base de W_n. Por lo tanto $\{e_1, \ldots, e_k, d_1, \ldots, d_p\}$ es una base de Z_n. Si llamamos V_n al submódulo libre generado por $\{e_1, \ldots, e_k\}$, entonces obtenemos la suma directa

$$Z_n \cong V_n \oplus W_n.$$

Para demostrar la descomposición de C_n en suma directa, definimos la siguiente sucesión exacta corta

$$0 \longrightarrow Z_n \longrightarrow C_n \underset{\rho}{\longleftarrow} B_{n-1} \longrightarrow 0$$

Es escindible ya que B_{n-1} es libre. Por la proposición 1.6, obtenemos que si $U_n = \mathrm{Im}(\rho)$ entonces $C_n = U_n \oplus Z_n = U_n \oplus V_n \oplus W_n$. □

Gracias al teorema anterior obtenemos que

$$H_n \cong L_{H_n} \oplus T_{H_n} \cong \frac{Z_n}{W_n} \oplus \frac{W_n}{B_n},$$

por lo que el siguiente paso es desarrollar una forma de computar $\frac{Z_n}{W_n}$ y $\frac{W_n}{B_n}$. Sea $\mathcal{K}$ un complejo simplicial y sea $M \in \mathfrak{M}_{m\times p}$ la matriz asociada al operador borde $\partial_p : C_n \longrightarrow C_{n-1}$. Por el teorema 1.33 transformamos la matriz M a su forma normal de Smith

$$\left[\begin{array}{ccc|ccc} \alpha_1 & & 0 & & & \\ & \ddots & & & 0 & \\ 0 & & \alpha_l & & & \\ \hline & & & 0 & & \\ & 0 & & & \ddots & \\ & & & & & 0 \end{array}\right]$$

respecto a una base $\{e_1, \ldots, e_l, e_{l+1}, \ldots, e_p\}$ de C_n, y una base $\{g_1, \ldots, g_l, g_{l+1}, \ldots, g_m\}$ de C_{n-1}. Calculemos ahora bases para Z_n, W_{n-1} y B_{n-1}.

Sea $c_n \in Z_n$, entonces lo podemos representar como

$$c_n = \sum_{i=1}^{p} \lambda_i \cdot e_i,$$

luego

$$\partial_n(c_n) = \sum_{i=1}^{p} \lambda_i \cdot \partial_n(e_i) = \sum_{i=1}^{l} \lambda_i \cdot \alpha_i \cdot g_i = 0.$$

Dado que los elementos g_i son linealmente independientes y los α_i son no nulos, tenemos entonces que c_n es un ciclo si, y solo si, $\lambda_i = 0$ para $i \in \{1, \ldots, l\}$, luego $\{e_{l+1}, \ldots, e_p\}$ es una base de Z_n.

Para calcular una base de W_{n-1}, hay que tener en cuenta que $g_i \in W_n$ para $i = 1, \ldots, l$, ya que $\alpha_i \cdot g_i = \partial_n(e_i)$. Vamos a probar por lo tanto que $\{g_1, \ldots, g_l\}$ es una base de W_n. Sea

$$c_{n-1} = \sum_{i=1}^{m} \mu_i \cdot g_i$$

una $(n-1)$-cadena tal que $c_{n-1} \in W_n$. Entonces existen $c_n \in C_n$ y $\gamma \neq 0$ tales que

$$\gamma \cdot c_{n-1} = \gamma \cdot \sum_{i=1}^{m} \mu_i \cdot g_i = \partial_n(c_n) = \sum_{i=1}^{l} \lambda_i \cdot \alpha_i \cdot g_i,$$

por lo que $\gamma \cdot \mu_i = 0$ para $i > l$. Luego $\{g_1, \ldots, g_l\}$ es una base de W_n.

Por último, calculemos una base para B_{n-1}. Sea $c_{n-1} \in B_{n-1}$, luego existe $c_n \in C_n$ tal que

$$c_{n-1} = \partial_n(c_n) = \sum_{i=1}^{l} \lambda_i \cdot \alpha_i \cdot g_i, \text{ donde los } \lambda_i \text{ son las coordenadas de } c_n.$$

Esto implica que el conjunto $\{\alpha_1 \cdot g_1, \ldots, \alpha_l \cdot g_l\}$ genera a B_{n-1}. Además es una base ya que son independientes.

Tenemos por lo tanto que dado un complejo simplicial $\mathcal{K}$ y un operador borde ∂_n, si llamamos D a la matriz asociada a ∂_n en su forma normal de Smith, es facil comprobar lo siguiente:

1. El rango de Z_n es el número de columnas nulas de D.

2. El rango de W_{n-1} es el número de filas no nulas de D.

A partir del teorema de estructura 1.34, podemos calcular el rango de H_n a partir de su parte libre. Hemos visto que L_{H_n} es isomorfo a $\frac{Z_n}{W_n}$ luego

$$\text{rango}(H_n) = \text{rango}(Z_n) - \text{rango}(W_n).$$

Definición 1.38. Llamaremos *n-ésimo número de Betti* de $\mathcal{K}$, denotado por β_n, al rango de H_n.

Hemos visto por lo tanto que podemos calcular H_n a partir de las matrices asociadas a los operadores borde ∂_n y ∂_{n+1}. Como consecuencia, hemos probado el siguiente teorema.

Teorema 1.39. Los módulos de homología de un complejo simplicial $\mathcal{K}$ sobre un dominio de ideales principales $\mathbb{F}$ son computables.

1.5 Complejos simpliciales asociados a conjuntos de datos. Filtraciones

En esta sección veremos distintas técnicas para asociar complejos simpliciales a conjuntos de datos.

Definición 1.40. Sea $X \subseteq \mathbb{R}^n$ un espacio finito con una distancia d y sea $\varepsilon > 0$. El *complejo de Čech*, que denotaremos por $\check{\text{C}}\text{ech}_\varepsilon(X, d)$, es el complejo simplicial abstracto formado por los símplices de la forma $< x_0, x_1, \ldots, x_k >$, donde $x_i \in X$ para $i = 0, \ldots k$ satisface que

$$\bigcap_{i=0}^{k} B_d(x_i, \varepsilon) \neq \emptyset.$$

Definición 1.41. Sea $X \subseteq \mathbb{R}^n$ un espacio finito con una distancia d y sea $\varepsilon > 0$. El *complejo de Vietoris-Rips*, que denotaremos por $\text{VR}_\varepsilon(X, d)$, es el complejo simplicial abstracto formado por los símplices de la forma $< x_0, x_1, \ldots, x_k >$, donde $x_i \in X$ para $i = 0, \ldots k$ satisface que

$$d(x_i, x_j) < 2\varepsilon, \quad \text{para todo} \quad 0 \leq i, j \leq k.$$

Proposición 1.42. Sea $X \subseteq \mathbb{R}^n$ un espacio finito con una distancia d. Para todo $\varepsilon > 0$ se tiene que

$$\check{\text{C}}\text{ech}_\varepsilon(X, d) \subseteq \text{VR}_\varepsilon(X, d).$$

Demostración. Sea $< x_0, x_1, \ldots, x_k > \in \check{\text{C}}\text{ech}_\varepsilon(X, d)$, luego se tiene que

$$\bigcap_{i=0}^{k} B_d(x_i, \varepsilon) \neq \emptyset.$$

Para todo $0 \leq i, j \leq k$ se satisface que $B_d(x_i, \varepsilon) \cap B_d(x_j, \varepsilon) \neq \emptyset$, luego $d(x_i, x_j) < 2\varepsilon$ para todos los vértices del símplice, por lo que $< x_0, x_1, \ldots, x_k > \in \text{VR}_\varepsilon(X, d)$. □

Definición 1.43. Una *filtración* de un complejo simplicial $\mathcal{K}$ es una sucesión de subcompjelos simpliciales encajados de la siguiente forma

$$\emptyset = \mathcal{K}_0 \subseteq \mathcal{K}_1 \subseteq \cdots \subseteq \mathcal{K}_n = \mathcal{K}.$$

Se puede inducir una filtración a partir de una aplicación monótona $f : \mathcal{K} \longrightarrow \mathbb{R}$, es decir, que si $\tau, \sigma \in \mathcal{K}$ y τ es una cara de σ, entonces $f(\tau) \leq f(\sigma)$. Consiste en formar la cadena de subcomplejos simpliciales

$$\emptyset = \mathcal{K}_0 \subseteq \mathcal{K}_1 \subseteq \cdots \subseteq \mathcal{K}_n = \mathcal{K},$$

donde $\mathcal{K}_i = f^{-1}((-\infty, a_i])$, para ciertos $-\infty = a_0 < a_1 < \cdots < a_n$.

Observación 1.44. Es fácil ver que el complejo de Čech y el complejo de Vietoris-Rips inducen una filtración cada uno de ellos.

1.5.1 Filtraciones a partir de series temporales

Para trabajar con series temporales las describiremos como funciones continuas a trozos cuyo dominio es un intervalo cerrado de tiempo. Es decir, podemos construir una serie temporal de la siguiente forma:

Dado un vector

$$X = (x_0, x_1, \ldots, x_{N-1}) \in \mathbb{R}^N,$$

podemos expresar X como una aplicación $X : \{0, 1, \ldots, N-1\} \longrightarrow \mathbb{R}$, donde $X(i) = x_i$ para $0 \leq i \leq N-1$. A partir de X, vamos a construir una función continua a trozos $f_X : \mathbb{R} \longrightarrow \mathbb{R}$ tal que $f_X(x_i) = X(i) = x_i$ para $0 \leq i \leq N-1$.

Para construir f_X, sea $\{\varphi_0, \varphi_1, \ldots, \varphi_{N-1}\}$ una base de funciones continuas $\varphi_i : \mathbb{R} \longrightarrow \mathbb{R}$ que tienen la siguiente forma:

$$\varphi_i(t) = \begin{cases} t - i + 1 & \text{si } i - 1 \leq t \leq i \\ i + 1 - t & \text{si } i \leq t \leq i + 1 \\ 0 & \text{si } i \notin [i-1, i+1] \end{cases}$$

para $i = 1, 2, \ldots, N-2$. Los otros casos son

$$\varphi_0(t) = \begin{cases} 1 - t & \text{si } 0 \leq t \leq i \\ 0 & \text{si } i \notin [0, 1] \end{cases}$$

y

$$\varphi_{N-1}(t) = \begin{cases} t - N + 2 & \text{si } N - 2 \leq t \leq N - 1 \\ 0 & \text{si } i \notin [N-2, N-1] \end{cases}$$

Es fácil ver que son funciones continuas y se tiene además que $f_X : \mathbb{R} \longrightarrow \mathbb{R}$ dada por

$$f_X(t) := \sum_{i=0}^{N-1} x_i \cdot \varphi_i(t)$$

es una función continua y satisface que $f_X(i) = x_i$ para todo $i = 0, 1, \ldots, N-1$.

Podemos dotar además a los conjuntos X de una norma de la siguiente forma

$$\|X\| := \|f_X\|_{L^2(\mathbb{R})} = \left(\int_{-\infty}^{+\infty} \|f_X\|^2 \right)^{1/2}.$$

Como consecuencia se tiene el siguiente resultado:

Proposición 1.45. La aplicación lineal

$$\Phi : (\mathbb{R}^N, \|\cdot\|) \longrightarrow (L^2(\mathbb{R}), \|\cdot\|_{L^2(\mathbb{R})})$$

donde $\Phi(X) = f_X$, es una isometría inyectiva entre espacios de Hilbert.

Demostración. Es isométrica por la propia definición de la norma de $\mathbb{R}$. La inyectividad se debe a que el conjunto de aplicaciones $\{\varphi_0, \varphi_1, \ldots, \varphi_{N-1}\}$ es linealmente independiente en $L^2(\mathbb{R})$. □

Dicho esto, veamos ahora que se puede asignar una filtración a toda serie temporal. Para $X \in \mathbb{R}^N$ sea f_X una serie temporal. Podemos definir entonces el siguiente complejo simplicial abstracto:

$$\mathcal{K}(X) := \{\{0\}, \{1\}, \ldots, \{N-1\}, [0,1], [1,2], \ldots, [N-2, N-1]\}.$$

Definición 1.46. Dado $\lambda \in \mathbb{R}$, llamaremos *subnivel* λ *de* f_X al conjunto

$$\mathcal{LS}_\lambda(f_X) := \{\sigma \in \mathcal{K}(X) : f_X(z) \leq \lambda \text{ para todo } z \in \sigma\}.$$

Sean ahora

$$\lambda_{\max} := \max_{t \in [0, N-1]} f_X(t) = \max_{0 \leq i \leq N-1} X(i)$$

y

$$\lambda_{\min} := \min_{t \in [0, N-1]} f_X(t) = \min_{0 \leq i \leq N-1} X(i).$$

Proposición 1.47. Dada una serie temporal f_X, sean $\lambda_0 < \lambda_{\min} \leq \lambda_1 \leq \lambda_2 \leq \lambda_{\max}$. Se tiene la siguiente sucesión de complejos simpliciales

$$\emptyset = \mathcal{LS}_{\lambda_0}(f_X) \subseteq \mathcal{LS}_{\lambda_1}(f_X) \subseteq \mathcal{LS}_{\lambda_2}(f_X) \subseteq \mathcal{LS}_{\lambda_{\max}}(f_X) = \mathcal{K}(X).$$

Por lo que los complejos de la forma $\mathcal{LS}_\lambda(f_X)$ inducen una filtración de $\mathcal{K}(X)$.

A continuación definimos las características de las funciones pertenecientes a $\Phi(\mathbb{R}^N)$ como

$$\mathfrak{C}(\Phi(\mathbb{R}^N)) := \{[i,j] \subseteq \mathbb{R} : 0 \leq i \leq j \leq N-1\}.$$

Es fácil ver que para todo $\lambda \in \mathbb{R}$ se tiene que $\mathcal{LS}_\lambda(f_X) \subseteq \mathfrak{C}(\Phi(\mathbb{R}^N))$, por lo que podemos construir las características asociadas al subnivel λ de f_X de la siguiente forma.

Definición 1.48. Sea $I \in \mathfrak{C}(\Phi(\mathbb{R}^n))$. Diremos que I pertenece a la *característica asociada al subnivel* λ *de* f_X, que denotaremos como $\mathfrak{C}(\mathcal{LS}_\lambda(f_X))$, si existen $I_1, \ldots, I_k \in \mathcal{LS}_\lambda(f_X)$ tales que $I = I_1 \cup \cdots \cup I_k$, es conexo y para todo $J \in \mathcal{LS}_\lambda(f_X)$ tal que $J \neq I_i$ para $1 \leq i \leq k$, se tiene que $J \cap \mathrm{Int}(I) = \emptyset$.

A raíz de esta definición, denominaremos *característica de* f_X al conjunto

$$\mathfrak{C}(f_X) := \bigcup_{\lambda \in \mathbb{R}} \mathfrak{C}(\mathcal{LS}_\lambda(f_X)).$$

En capítulos posteriores denotaremos la característica asociada al subnivel λ como $\mathfrak{C}_\lambda(f_X)$ para simplificar la notación.

Veamos estos nuevos términos con el siguiente ejemplo:

Ejemplo 1.49. Sea X la siguiente serie temporal

$$X = (4, 5, 7, 6, 4, 5, 1, 3, 2).$$

Podemos construir entonces la función f_X obteniendo los valores $\lambda_{\min} = 1$ y $\lambda_{\max} = 7$. Dicho esto, podemos calcular los subniveles λ de f_X:

$$\begin{aligned}
\mathcal{LS}_0(f_X) &= \emptyset \\
\mathcal{LS}_1(f_X) &= \{\{6\}\} \\
\mathcal{LS}_2(f_X) &= \mathcal{LS}_1(f_X) \cup \{\{8\}\} \\
\mathcal{LS}_3(f_X) &= \mathcal{LS}_2(f_X) \cup \{[6,7],[7,8],\{7\}\} \\
\mathcal{LS}_4(f_X) &= \mathcal{LS}_3(f_X) \cup \{\{0\},\{4\}\} \\
\mathcal{LS}_5(f_X) &= \mathcal{LS}_4(f_X) \cup \{\{1\},\{5\},[0,1],[4,5],[5,6]\} \\
\mathcal{LS}_6(f_X) &= \mathcal{LS}_5(f_X) \cup \{\{3\},[3,4]\} \\
\mathcal{LS}_7(f_X) &= \mathcal{LS}_6(f_X) \cup \{\{2\},[1,2]\}.
\end{aligned}$$

En cuanto a las características asociadas a los subniveles λ de f_X se tiene:

$$\begin{aligned}
\mathfrak{C}(\mathcal{LS}_0(f_X)) &= \emptyset \\
\mathfrak{C}(\mathcal{LS}_1(f_X)) &= \{\{6\}\} \\
\mathfrak{C}(\mathcal{LS}_2(f_X)) &= \{\{6\},\{8\}\} \\
\mathfrak{C}(\mathcal{LS}_3(f_X)) &= \{[6,8]\} \\
\mathfrak{C}(\mathcal{LS}_4(f_X)) &= \{\{0\},\{4\},[6,8]\} \\
\mathfrak{C}(\mathcal{LS}_5(f_X)) &= \{[0,1],[4,8]\} \\
\mathfrak{C}(\mathcal{LS}_6(f_X)) &= \{[0,1],[3,8]\} \\
\mathfrak{C}(\mathcal{LS}_7(f_X)) &= \{[0,8]\}.
\end{aligned}$$

Además, la característica de f_X será

$$\mathfrak{C}(f_X) = \{\{6\},\{8\},\{0\},\{4\},[6,8],[0,1],[4,8],[3,8],[0,8]\}.$$

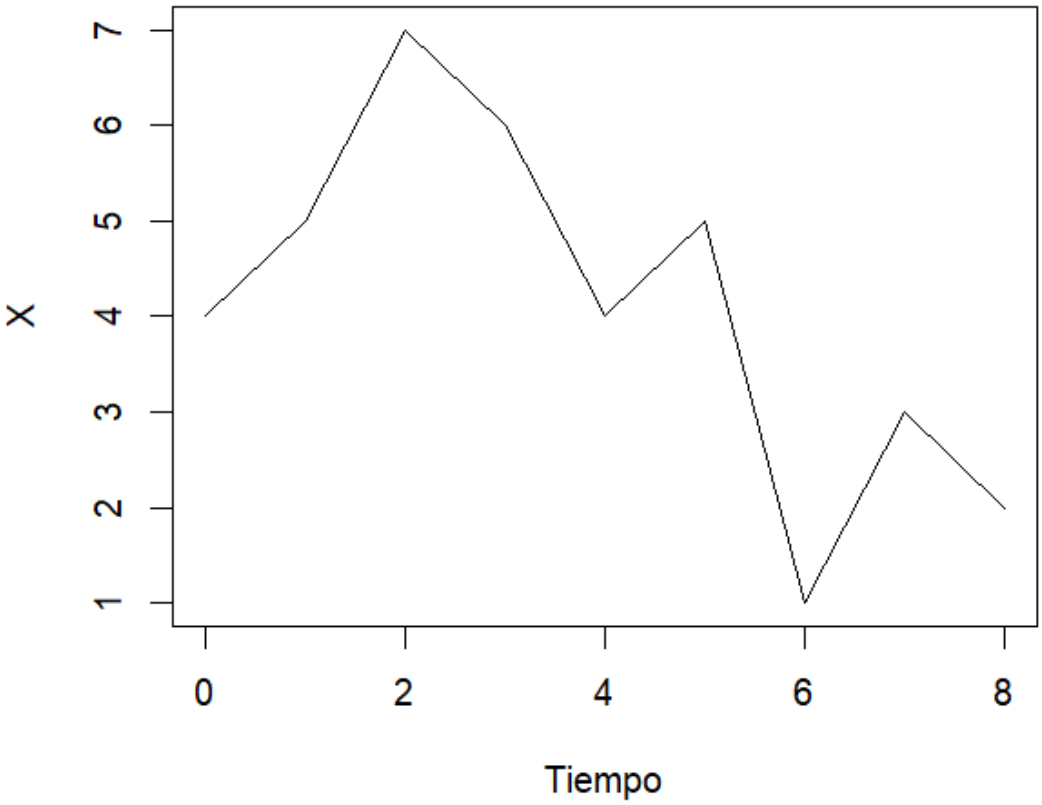

Figura 1.2: Representación gráfica de la serie temporal f_X.

Capítulo 2

Homología persistente

En este capítulo vamos a introducir el concepto de homología persistente, la cual es una poderosa herramienta que nos ayuda a analizar y clasificar distintos conjuntos de datos finitos. Consiste en formar complejos simpliciales a través de las técnicas que hemos visto anteriormente y formar filtraciones. Haremos uso además de los grupos de homología, gracias a los cuales definiremos los códigos de barras.

En este trabajo lo definiremos para cualquier filtración, aunque posteriormente nos centraremos en el caso de las series temporales.

2.1 Código de barras de una filtración

Sea $\mathcal{K}$ un complejo simplicial y sea $f : \mathcal{K} \longrightarrow \mathbb{R}$ una aplicación monótona que induce una filtración

$$\emptyset = \mathcal{K}_0 \subseteq \mathcal{K}_1 \subseteq \cdots \subseteq \mathcal{K}_n = \mathcal{K}.$$

Para cada $0 \leq i \leq j \leq n$, sea $\varphi_p^{i,j} : H_p(\mathcal{K}_i) \longrightarrow H_p(\mathcal{K}_j)$ el homomorfismo de grupos que se induce de la aplicación de inclusión de $\mathcal{K}_i$ en $\mathcal{K}_j$.

Sea ahora $H_p^{i,j} = \mathrm{Im}(\varphi_p^{i,j})$ y $\beta_p^{i,j} = \mathrm{rango}(H_p^{i,j})$. Podemos saber entonces el número de clases de homología p-dimensionales que nacen en el complejo $\mathcal{K}_i$ y mueren en $\mathcal{K}_j$ de la siguiente forma

$$\mu_p^{i,j} = (\beta_p^{i,j-1} - \beta_p^{i,j}) - (\beta_p^{i-1,j-1} - \beta_p^{i-1,j})$$

para todo $i < j$ y toda dimensión p. Es fácil ver que el término $\beta_p^{i,j-1} - \beta_p^{i,j}$ cuenta el número de clases que nacen en $\mathcal{K}_i$ o antes y mueren en $\mathcal{K}_j$, mientras que $\beta_p^{i-1,j-1} - \beta_p^{i-1,j}$ cuenta las clases que nacen en $\mathcal{K}_{i-1}$ o antes y mueren en $\mathcal{K}_j$.

Sean ahora $-\infty = a_0 < a_1 < \cdots < a_n$ tales que $f^{-1}((-\infty, a_i]) = \mathcal{K}_i$ para $i = 0, 1, \ldots, n$.

Definición 2.1. Diremos que el *código de barras* de la filtración de un complejo $\mathcal{K}$ a partir de una función f es el multiconjunto de intervalos

$$\mathcal{B}_p(f, \mathcal{K}) := \{[a_i, a_j) \quad \text{tales que} \quad i < j \quad \text{y multiplicidad} \quad \mu_p^{i,j}\}.$$

Es decir, son los intervalos que indican el nacimiento y la muerte de las clases de homología en los $\mathcal{K}$.

Los códigos de barras se pueden interpretar de forma gráfica a través de un diagrama de persistencia, un gráfico que tiene como eje de coordenadas los nacimientos de las clases y como eje de abscisas las muertes. Los puntos que aparecen serán de la forma (a_i, a_j) y siempre estarán por encima de la diagonal ya que $i < j$. Otra forma es a través de intervalos donde cada uno representa a una clase, donde el inicio de este indica su nacimiento y el final su muerte.

Ejemplo 2.2. Sea $X \subseteq \mathbb{R}^2$ el conjunto dado por 100 uniformemente aleatorios de la circunferencia centrada en el origen y de radio unidad 2.1.

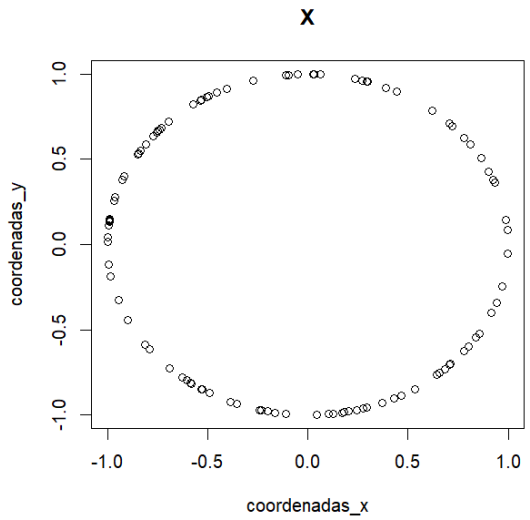

Figura 2.1: Conjunto X dado por 100 puntos aleatorios de la circunferencia unidad

A partir de X creamos una filtración de complejos de Vietoris-Rips. En la tabla 2.1, podemos ver un fragmento del código de barras. El primer intervalo indica que dicha clase muere cerca de 2, pero en realidad debería ser infinito. Esto se debe a que así es posible representar este valor en el diagrama de persistencia, como vemos en 2.2.

Table 2.1: Fragmento del código de barras de X

	Dimensión	Nacimientos	Muertes
[1,]	0	0.0000000	1.9999999793
[2,]	0	0.0000000	0.2085510183
⋮	⋮	⋮	⋮
[100,]	0	0.0000000	0.0007823978
[101,]	1	0.2232352	1.7339087370

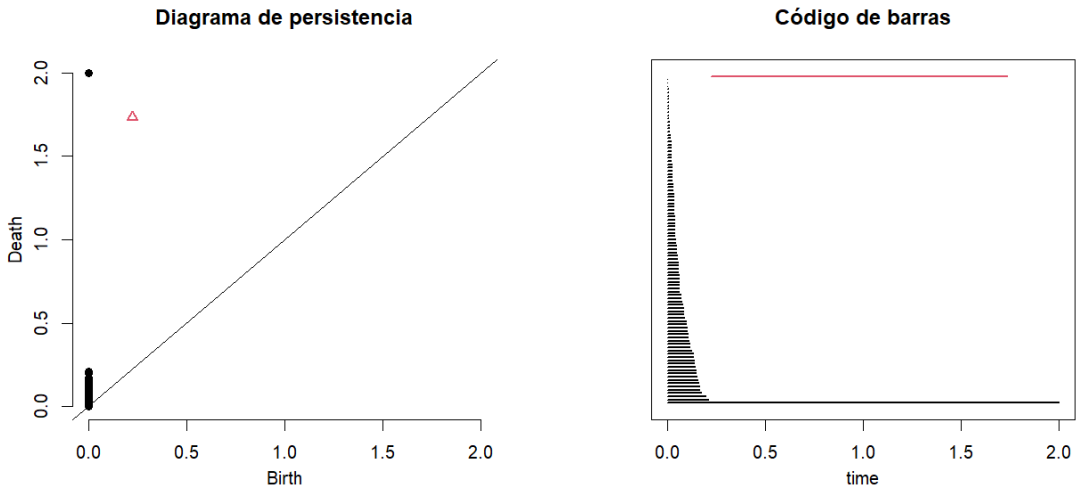

Figura 2.2: En la imagen de la izquierda se muestra el diagrama de persistencia de la filtración. Con los círculos negros muestra las clases de dimensión 0, mientras que con el triangulo rojo la clase de dimensión 1. En la imagen de la derecha representa el tiempo de vida de las clases a lo largo del tiempo, indicando su nacimiento y su muerte. Las clases de color negro son de dimensión 0 mientras que la de color rojo es de dimensión 1.

2.1.1 Códigos de barras de series temporales

Los complejos simpliciales que obtenemos de las series temporales están dados por símplices de dimensión 1 y vértices, además estos no hacen ciclos cerrados por lo que únicamente tiene sentido trabajar con los grupos de homología H_0, es decir, las componentes conexas de los complejos.

Lema 2.3. Sea $X \in \mathbb{R}^N$ y sea f_X la serie temporal asociada a X. Entonces existe una serie de valores reales positivos $\{\lambda_1, \ldots, \lambda_r\}$ ordenados de menor a mayor y diferentes entre sí, tales que $\mathfrak{C}_{\lambda_i}(f_X)$ y $\mathfrak{C}_{\lambda_j}(f_X)$ son distintos si $i \neq j$. En particular se tiene que para todo $i = 1, \ldots, r$, si $\lambda < \lambda_i$ entonces

$$\mathfrak{C}_{\lambda}(f_X) \neq \mathfrak{C}_{\lambda_i}(f_X).$$

Demostración. Sea $X = (x_1, x_2, \ldots, x_N) \in \mathbb{R}^N$ y sea f_X su serie temporal asociada. Si definimos $\lambda_{i_1} = f_X(1)$, tenemos entonces que si $\lambda < \lambda_{i_1}$, se satisface que $f_X(1) \notin \mathfrak{C}_{\lambda}(f_X)$ pero $f_X(1) \in \mathfrak{C}_{\lambda_1}(f_X)$. Tenemos por lo tanto que son complejos simpliciales diferentes.

Para obtener los valores $\lambda_1, \lambda_2, \ldots, \lambda_r$, basta ordenar los elementos del vector X de menor a mayor excluyendo los valores repetidos. □

Para todo $i = 1, \ldots, r$ sea $\mathcal{K}_i := \mathcal{LS}_{\lambda_i}(f_X)$. Podemos definir la función $F_X : \mathcal{K}(X) \longrightarrow \mathbb{R}$ donde

$$F_X(\sigma) = \min\{\lambda \in \mathbb{R} : \sigma \in \mathcal{LS}_{\lambda}(f_X)\}.$$

Obtenemos por lo tanto la filtración

$$\mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \cdots \subseteq \mathcal{K}_r,$$

a la cual la podemos aplicar la definición de código de barras del caso general.

Definición 2.4. Sea f_X una serie temporal y sean $\lambda_1 < \lambda_2 < \cdots < \lambda_r$ el conjunto de valores reales asociados a la f_X del lema anterior. Llamaremos *código de barras asociado a* f_X al multiconjunto de intervalos

$$\mathcal{B}(X) = \{[\lambda_i, \lambda_j) \quad \text{tales que} \quad i < j \quad \text{y multiplicidad} \quad \mu_0^{i,j}\}.$$

Esto quiere decir que para definir el código de barras solo nos hacen falta aquellos complejos de la filtración en los que su grupo de homología sufre una variación de rango.

Ejemplo 2.5. En el ejemplo 1.49, el código de barras correspondiente a la serie temporal sería

$$\mathcal{B}(X) = \{[1,7), [2,3), [4,5), [4,7)\}.$$

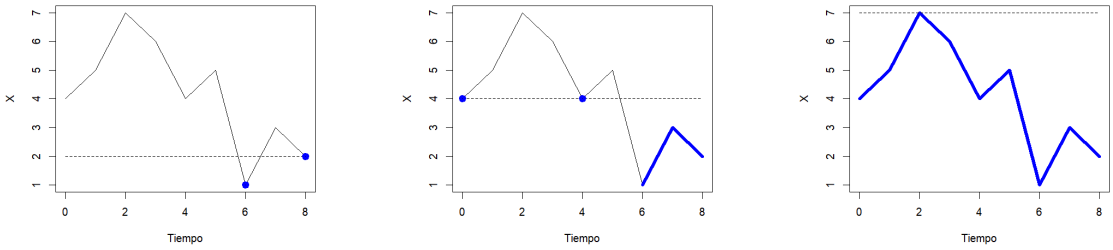

Figura 2.3: En estos gráficos se puede apreciar la evaluación de la serie temporal f_X de varios complejos. En la imagen de la izquierda se muestra la del complejo $F_X^{-1}((-\infty, 2])$, el del medio es la de $F_X^{-1}((-\infty, 4])$ y a la derecha se encuentra la del complejo simplicial $F_X^{-1}((-\infty, 7])$.

2.1.2 Imágenes persistentes de series temporales

A continuación, para determinar el grado de similaridad entre dos códigos de barras hacemos lo siguiente. Sea f_X una serie temporal y sea $\mathcal{B}(X)$ su código de barras correspondiente. Si los intervalos de $\mathcal{B}(X)$ son de la forma $[c_i, d_i)$, definimos el multiconjunto

$$\mathcal{TV}(f_X) := \{(c_1, d_1 - c_1), (c_2, d_2 - c_2), \cdots, (c_k, d_k - c_k)\},$$

para cierto $k \geq 1$ y tales que $d_1 - c_1 \leq d_2 - c_2 \leq \cdots \leq d_- c_k$. Definimos la función de pesos $\omega : \mathcal{TV}(f_X) \longrightarrow [0, 1]$ dada por

$$\omega(c_i, d_i - c_i) = \frac{d_i - c_i}{d_k - c_k} \quad \text{para} \quad 1 \leq i \leq k.$$

Fijamos ahora un número M natural y sea $g_u(x, y)$ la función de distribución normal multivariante de dos variable centrada en $u \in \mathcal{TV}(f_X)$ y con distribución

$$\sigma = \frac{1}{M} \cdot (c_k, d_k - c_k) \cdot \mathrm{Id}_2,$$

donde Id_2 es la matriz identidad de dimensión 2. Definimos como consecuencia, la función de núcleos de persistencia de una serie temporal f_X, dada por $\rho_X : \mathbb{R}^2 \longrightarrow \mathbb{R}$, donde

$$\rho_X(x, y) : \sum_{u \in \mathcal{TV}(f_X)} \omega(u) \cdot g_u(x, y).$$

Asociaremos entonces a una serie temporal $X \in \mathbb{R}^N$ una matriz $\Delta(X) \in \mathfrak{M}_M(\mathbb{R})$ de la siguiente forma. Sea $\varepsilon > 0$ lo suficientemente pequeño y sea

$$\Omega_{X,\varepsilon} := [u_1, v_1] \times [u_2, v_2] \subseteq \mathbb{R}^2$$

una región rectangular del plano real que satisface la desigualdad

$$\iint_{\Omega_{X,\varepsilon}} \rho_X(x, y)\ dx\ dy \geq 1 - \varepsilon.$$

A continuación hacemos una partición de la región equidistante, es decir, para

$$u_1 = p_0 < p_1 < \cdots < p_M = v_1 \quad \text{y} \quad u_2 = q_0 < q_1 < \cdots < q_M = v_2,$$

sea $P_{i,j} = [p_i, p_{i+1}] \times [q_j, q_{j+1}]$, donde $0 \leq i, j \leq M - 1$.

Definición 2.6. Llamaremos *imagen de persistencia de X asociada a la partición $\mathcal{P}$*, donde $\mathcal{P} = \{P_{i,j}\}_{i,j=0}^{M-1}$, a la matriz dada por

$$\mathcal{IP}(X, M, \mathcal{P}, \varepsilon) = \left(\int_{P_{i,j}} \rho_X(x, y)\ dx\ dy \right)_{i,j=0}^{i,j=M-1} \in \mathfrak{M}_M(\mathbb{R}).$$

Las imágenes de persistencia son de gran utilidad a la hora de clasificar los códigos de barras en categorías. Esto se debe a que se puede hacer uso de redes convolucionales, un tipo de arquitectura de redes neuronales que clasifica las imágenes con gran precisión, como veremos mas adelante.

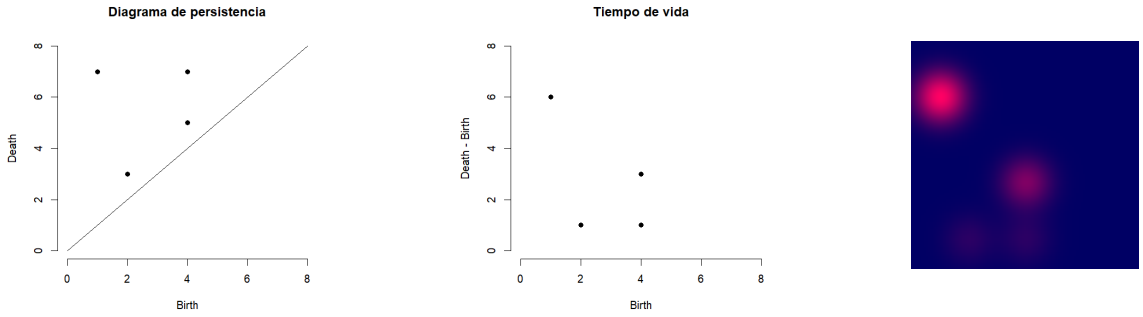

Figura 2.4: En estos gráficos se muestra a la izquierda el diagrama de persistencia del ejemplo de la serie temporal visto anteriormente, en el medio el tiempo de vida de las características y a la derecha la imagen de persistencia.

2.2 Características persistentes

Una vez obtenido el código de barras a partir de un conjunto de datos, vamos a definir el concepto de característica persistente. Se trata de asociar una función a cada conjunto de datos a partir de una filtración, la cual pertenece a un espacio de Banach, por lo que podremos usar herramientas de estadística y de machine learning.

Sea $\mathcal{K}$ un complejo simplicial y sea

$$\emptyset = \mathcal{K}_0 \subseteq \mathcal{K}_1 \subseteq \cdots \subseteq \mathcal{K}_n = \mathcal{K},$$

una filtración. Vamos a extenderla a una filtración continua, esto es, que si $t \in [i, i+1)$ con $i = 0, 1, \ldots, n-1$, entonces denotaremos $\mathcal{K}_t = \mathcal{K}_i$.

Tomando la notación de la sección anterior en la que denotamos por

$$\varphi_q^{i,j} : H_q(\mathcal{K}_i) \longrightarrow H_q(\mathcal{K}_j),$$

al homomorfismo inducido por la aplicación de inclusión de $\mathcal{K}_i$ en $\mathcal{K}_j$, tenemos el siguiente resultado.

Proposición 2.7. Sean $0 \leq i < j < k \leq n$. Se satisface que

$$\varphi_q^{i,k} = \varphi_q^{j,k} \circ \varphi_q^{i,j}.$$

Demostración. Esto se debe a que la aplicación de inclusión $\mathcal{K}_i \longrightarrow \mathcal{K}_k$ es igual a la composición de las inclusiones $\mathcal{K}_i \longrightarrow \mathcal{K}_j$ con $\mathcal{K}_j \longrightarrow \mathcal{K}_k$. □

Siendo $\beta_q^{i,j}$ la dimensión de la imagen de $\varphi_q^{i,j}$, obtenemos lo siguiente.

Proposición 2.8. Si $i \leq j \leq k \leq l$, entonces $\beta_q^{j,k} \leq \beta_q^{i,l}$.

Demostración. Por la proposición anterior tenemos que

$$\varphi_q^{i,l} = \varphi_q^{k,l} \circ \varphi_q^{j,k} \circ \varphi_q^{i,j}.$$

Como la dimensión de la imagen de un homomorfismo no puede ser mayor que la dimensión del espacio de salida, obtenemos la desigualdad. □

Corolario 2.9. Si $0 \leq h_1 \leq h_2$, para todo $t \in \mathbb{R}$ se tiene

$$\beta_q^{t-h_2,t+h_2} \leq \beta_q^{t-h_1,t+h_1}.$$

Demostración. Dado que $t-h_2 \leq t-h_1 \leq t+h_1 \leq t+h_2$, basta aplicar el lema anterior. □

Con estos resultados podemos definir las características persistentes

Definición 2.10. La *característica persistente* de la filtración de un complejo $\mathcal{K}$, es una función $\lambda : \mathbb{N} \times \mathbb{R} \longrightarrow \bar{\mathbb{R}}$, donde $\bar{\mathbb{R}}$ denota la recta extendida real, dada por

$$\lambda_q(k,t) = \sup\{m \geq 0 : \beta_q^{t-m,t+m} \geq k\}.$$

Otra forma alternativa de definir la característica persistente, si denotamos $\mathcal{B}_q(f,\mathcal{K})$ al código de barras de la filtración, es la siguiente:

$$\lambda_q(k,t) = \sup\{m \geq 0 : [t-m, t+m] \subseteq I_j \text{ para } k \text{ intervalos } I_j \text{ distintos de } \mathcal{B}_q(f,\mathcal{K})\}.$$

Observación 2.11. En el caso de las características persistentes asociadas a series temporales, como se tiene que $q = 0$, denotaremos $\lambda(k,t)$ en vez de $\lambda_0(k,t)$ para simplificar la notación.

Proposición 2.12. Sea $\lambda_q(k,t) : \mathbb{N}\times\mathbb{R} \longrightarrow \mathbb{R}$ una característica persistente. Se satisfacen las siguientes propiedades:

1. $\lambda_q(k,t) \geq 0$.
2. $\lambda_q(k,t) \geq \lambda_q(k+1,t)$.
3. Fijado $k \in \mathbb{N}$, se tiene que $\lambda_q(k,t)$ es de 1-Lipschitz.

Demostración. Las dos primeras propiedades son consecuencia directa de la definición.

En cuanto a la tercera propiedad, veamos que $\lambda_q(k,t)$ es de Lipschitz, es decir,

$$|\lambda_q(k,t) - \lambda_q(k,s)| \leq |t-s| \quad \text{para todo } t,s \in \mathbb{R}.$$

En primer lugar, podemos asumir sin pérdida de generalidad que $\lambda_q(k,t) \geq \lambda_q(k,s) \geq 0$. Si $\lambda_q(k,t) \leq |t-s|$, entonces

$$\lambda_q(k,t) - \lambda_q(k,s) \leq \lambda_q(k,t) \leq |t-s|.$$

Supongamos ahora que $\lambda_q(k,t) > |t-s|$. Sea $\lambda_q(k,t) - |t-s| > h > 0$, luego se tienen las desigualdades $t - \lambda_q(k,t) < s - h < s + h < t + \lambda_q(k,t)$. Por el corolario 2.9, se tiene que $\beta_q^{t-\lambda_q(k,t),t+\lambda_q(k,t)} \leq \beta_q^{s-h,s+h}$, luego $\beta_q^{s-h,s+h} \geq k$. Por la propia definición de $\lambda_q(k,s)$ se tiene que $\lambda_q(k,s) \geq h$ para todo $\lambda_q(k,t) - |t-s| > h > 0$. Esto implica que $\lambda_q(k,s) \geq \lambda_q(k,t) - |t-s|$, luego

$$|t-s| \geq \lambda_q(k,t) - \lambda_q(k,s).$$

□

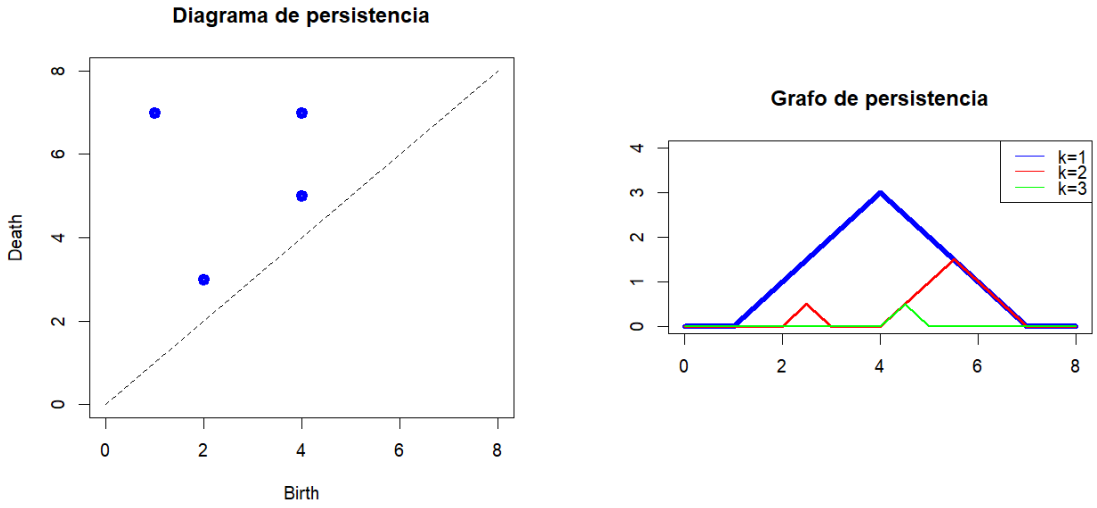

Figura 2.5: En la imagen de la izquierda se muestra el diagrama de persistencia de la serie temporal del ejemplo anterior. A la derecha se encuentra su correspondiente característica persistente.

2.2.1 El espacio de las características persistentes.

Recordamos que dado un espacio medible (E, σ, μ) y una función $f : E \longrightarrow \mathbb{R}$ medible μ-casi siempre, para $1 \leq p \leq +\infty$, podemos definir la norma

$$\|f\|_p = \left[\int |f|^p d\mu\right]^{\frac{1}{p}}.$$

En el caso en el que $p = +\infty$, se define la norma

$$\|f\|_\infty = \inf\{a : \mu(\{x \in E : |f(x) \ \geq a\}) = 0\}.$$

Definimos como consecuencia para $1 \leq p \leq \infty$ los espacios

$$\mathcal{L}^p(E) = \{f : E \longrightarrow \text{ tal que } \|f\|^p < +\infty\}.$$

Finalmente obtenemos los espacios $L^p(E) = \mathcal{L}^p(E)/\sim$, donde $f \sim g$ si $\|f - g\|^p = 0$.

Proposición 2.13. Sea ahora $\lambda_q(k,t) : \mathbb{N} \times \mathbb{R} \longrightarrow \mathbb{R}$ una característica persistente. La aplicación

$$\|\lambda_q\|_p = \sum_{k=1}^{\infty} \|\lambda_q(k,t)\|_p,$$

define una norma para las características persistentes. En otras palabras, las características persistentes tales que $\|\lambda_q\|_p < +\infty$, forman un espacio normado.

Demostración. Veamos que se cumplen las propiedades de las normas:

1. Por definición, se tiene siempre que $\|\lambda_q\|_p \geq 0$.

2. Supongamos que $\|\lambda_q\|_p = 0$, entonces $\|\lambda_q(k,t)\|_p = 0$ para todo $k = 1, 2, \ldots$ Luego $\lambda_q(k,t) = 0$ para todo k y como consecuencia $\lambda_q = 0$. Si $\lambda_q = 0$, es fácil ver que $\|\lambda_q\|_p = 0$.

3. Sea $\alpha \in \mathbb{R}$, luego

$$\|\alpha \cdot \lambda_q\| = \sum_{k=1}^{\infty} \|\alpha \cdot \lambda_q(k,t)\|_p = \sum_{k=1}^{\infty} |\alpha| \cdot \|\lambda_q(k,t)\|_p = |\alpha| \cdot \|\lambda_q\|.$$

4. Si tomamos otra característica persistente λ'_q, veamos que se cumple la desigualdad triangular

$$\|\lambda_q + \lambda'_q\|_p = \sum_{k=1}^{\infty} \|\lambda_q(k,t) + \lambda'_q(k,t)\|_p \leq \sum_{k=1}^{\infty} \left(\|\lambda_q(k,t)\|_p + \|\lambda'_q(k,t)\|_p\right) =$$

$$= \sum_{k=1}^{\infty} \|\lambda_q(k,t)\|_p + \sum_{k=1}^{\infty} \|\lambda'_q(k,t)\|_p = \|\lambda_q\|_p + \|\lambda'_q\|_p.$$

□

Proposición 2.14. El espacio de las características persistentes junto con la norma anterior, definen un espacio de Banach.

Demostración. Ya hemos visto que es un espacio normado, falta ver que es completo. Esto es, veamos que toda sucesión de Cauchy tiene límite.

Sea $\{\lambda_q^n\}_{n=0}^{\infty}$ una sucesión de Cauchy, es decir, para todo $\varepsilon > 0$, existe un número $N > 0$ tal que para todo $n, m > N$ se tiene que $\|\lambda_q^n - \lambda_q^m\|_p < \varepsilon$. Esto es equivalente a que

$$\sum_{k=1}^{\infty} \|\lambda_q^n(k,t) - \lambda_q^m(k,t)\|_p < \varepsilon,$$

luego $\|\lambda_q^n(k,t) - \lambda_q^m(k,t)\|_p < \varepsilon$ para todo $k \in \mathbb{N}$. La sucesión de funciones $\{\lambda_q^n(k,t)\}$, donde k está fijado, está contenida en el espacio de Banach$L^q(\mathbb{R})$ por lo que tiene límite al ser de Cauchy. Para todo $n \in \mathbb{N}$ sea

$$\lambda_q(k,t) = \lim_{n \longrightarrow \infty} \lambda_q^n(k,t).$$

Veamos por lo tanto que $\{\lambda_q^n\}_{n=0}^{\infty}$ converge a la característica persistente λ_q:

$$\lim_{n\to\infty} \|\lambda_q^n - \lambda_q\|_p = \lim_{n\to\infty} \sum_{k=1}^{\infty} \|\lambda_q^n(k,t) - \lambda_q^m(k,t)\|_p = \sum_{k=1}^{\infty} \lim_{n\to\infty} \|\lambda_q^n(k,t) - \lambda_q^m(k,t)\|_p = 0.$$

Luego el espacio de las características persistentes es completo, por lo que es un espacio de Banach. □

Capítulo 3

Aprendizaje automático. Métodos de clasificación

En este capítulo veremos dos técnicas de aprendizaje automático que usaremos para clasificar los datos obtenidos a partir de la homología persistente. En concreto, introduciremos los conceptos de redes neuronales convolucionales (CNN) y bosques aleatorios (RF) que nos serán útiles para clasificar las imágenes y las características persistentes, respectivamente.

3.1 Aprendizaje automático

El aprendizaje automático, o en inglés Machine Learning, es una rama de la inteligencia artificial que se centra en algoritmos que tratan de hacer predicciones o decisiones a partir de datos que ha recibido anteriormente como imput. Los algoritmos de aprendizaje automático son una gran herramienta para resolver problemas de los cuales no tengamos mucha información sobre su naturaleza, ya que a partir de datos es capaz de identificar patrones y relaciones entre dichos datos que nos ayudan a estimar unos parámetros con los que desarrollaremos un modelo de forma sencillo y con gran precisión. Una definición más formal es la siguiente:

Definición 3.1. Un algoritmo A se dice que es de *aprendizaje automático*, si aprende de una experiencia E respecto un conjunto de tareas T y con una eficiencia P, si al realizar una tarea de T aumenta la eficiencia P con una mayor experiencia.

Los sistemas de aprendizaje automático se pueden clasificar de diferentes formas, aunque la más común es la clasificación a partir de la supervisión humana:

1. *Aprendizaje supervisado*: Aquellos sistemas que están supervisados por personas, es decir, se asigna una etiqueta a los datos de entrenamiento para definir una clasificación.
2. *Aprendizaje no supervisado*: Estos sistemas no están supervisados por personas, es decir, los datos de entrenamiento no tienen una etiqueta, por lo que únicamente se sabe el número de grupos en los que ha dividido los datos y tras esto se asigna una etiqueta a cada grupo obtenido.
3. *Aprendizaje semisupervisado*: Sistemas que combinan datos etiquetados y no etiquetados.
4. *Aprendizaje por refuerzo*: Evolucionan a partir de refuerzos y penalizaciones dependiendo de las decisiones que realice.

En este trabajo nos centraremos en técnicas de aprendizaje supervisado para clasificar las imágenes de persistencia y las características persistentes.

En los algoritmos de aprendizaje supervisado, se suele dividir el conjunto de datos en dos grupos. El primer grupo es más grande que el segundo ya que suele tener en torno a un 75% del total de los datos. Estos se usan para entrenar el modelo. Para este conjunto de datos se usa la *función coste*, una función que el algoritmo tratará de minimizar para mejorar los parámetros del modelo. Esta mide, generalmente, la relación entre el valor predicho y el valor real de los datos de entrenamiento. La función coste que se suele utilizar más es el error cuadrático medio (MSR). El segundo grupo de datos contiene los restantes y se usa para evaluar la eficiencia del modelo, la cual es medida a partir de la *función de pérdida*. Esta función mide la discrepancia entre la predicción y el valor real de los datos de evaluación. También es común el uso del error cuadrático medio como función de pérdida.

Uno de los principales problemas del aprendizaje automático es el sobreaprendizaje, es decir, que el modelo toma demasiada información de los datos de entrenamiento y no es capaz de evaluar el caso general de forma precisa. Para solucionar esto se divide el conjunto de datos del test en otros dos grupos que suelen ser del mismo tamaño, siendo el primer conjunto los datos de validación y el segundo los datos de test. El proceso de

aprendizaje se detiene en el momento en el que el error de los datos de validación llegan a un mínimo. Si no es así el modelo minimizará el error de los datos de entrenamiento pero como consecuencia el error de los datos que no son de entrenamiento será mayor. Se tiene además que si tenemos una gran cantidad de datos, el error de los datos de validación será similar a los datos de test. Como consecuencia, se usan métodos numéricos que nos ayuden a minimizar la función del error de los datos de validación, como puede ser el método del gradiente.

3.2 Redes neuronales convolucionales

Las redes convolucionales son un tipo de red neuronal profunda que ha revolucionado el campo de la inteligencia artificial y la visión por computadora. Desde su introducción en la década de 1980, las redes convolucionales han evolucionado significativamente, impulsadas por avances en el poder computacional y el acceso a grandes cantidades de datos. En esta sección veremos en primer lugar una breve introducción a las redes neuronales, seguida de las redes neuronales profundas.

3.2.1 Redes neuronales de una capa

Una red neuronal es una función no lineal

$$f : \mathbb{R}^n \longrightarrow \mathbb{R},$$

que trata de predecir una respuesta y cuando toma como input un vector $x = (x_1, \ldots, x_n)$. De forma más específica se define como

$$f(x) = h\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right),$$

donde los w_i son los pesos correspondientes a cada entrada que irán variando, b es el umbral o sesgo y $h : \mathbb{R} \longrightarrow \mathbb{R}$ es la función de activación.

Hay muchos tipos de funciones de activación. Las más importantes son la función sigmoidea dada por

$$h(x) = \frac{e^x}{1 + e^x}$$

y la función de activación ReLU (rectified linear unit) dada por

$$h(x) = \max\{0, x\}.$$

En este trabajo usaremos la función ReLU ya que es más sencilla de computar que la función sigmoidea.

Dadas estas definiciones, nuestro objetivo es diseñar algoritmos de aprendizaje capaces de ajustar los pesos y el sesgo de la red neuronal. Para ellos existen varios algoritmos que obtienen unos pesos adecuados minimizando el error entre el valor predicho y el verdadero de los datos del conjunto de entrenamiento E. Este proceso inicia con unos pesos $w = (w_1, \ldots, w_n)$ que se han elegido de forma aleatoria. Para cada entrada $x = (x_1, \ldots, x_n)$ que tenga un valor esperado $\tilde{f}(x)$, tenemos que el valor predicho es

$$h\left(\sum_{i=1}^{n} w_i \cdot x_i + b\right).$$

Nuestro objetivo entonces es minimizar la función del error cuadrático medio

$$L(w, b) = \frac{1}{|E|} \cdot \sum_{x \in E} (\tilde{f}(x) - f(x))^2.$$

Para ello usaremos el método del descenso del gradiente. Este método consiste en iterar la función

$$(\tilde{w}, \tilde{b}) = (w, b) - \eta \cdot \nabla L(w, b),$$

donde $\tilde{w}$ y $\tilde{b}$ son los nuevos pesos y el nuevo sesgo, respectivamente. A la constante positiva η la llamaremos *tasa de aprendizaje*.

Este no es el único método que existe ya que en algunos casos puede ser computacionalmente muy costoso. Se puede realizar también con métodos estocásticos que a pesar de ser menos precisos es más rápido, o una combinación de ambos métodos, esto es dividir el conjunto de entrenamiento en subconjuntos más pequeños que llamaremos lotes, con los cuales podremos combinar dichos métodos. Con este último método es posible calibrar a nuestro criterio la precisión y la velocidad del algoritmo.

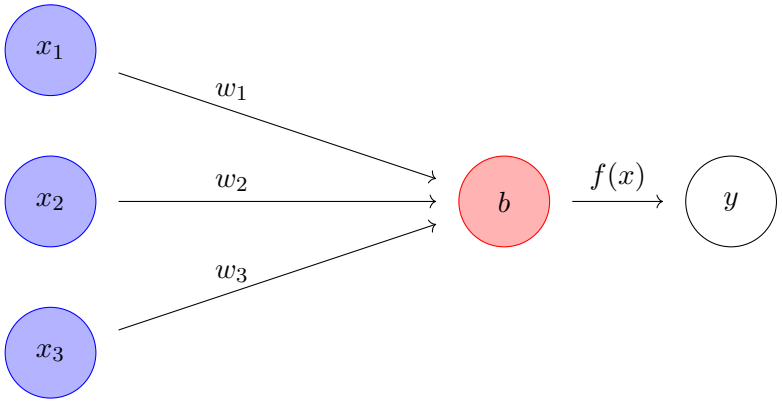

Figura 3.1: Diagrama de una red neuronal de una capa de la forma $f : \mathbb{R}^3 \longrightarrow \mathbb{R}$.

3.2.2 Redes neuronales multicapa. Aprendizaje profundo

Una de las principales limitaciones de las redes neuronales de una capa es la incapacidad para clasificar patrones que no son linealmente separables. Este problema se solucionó gracias a la introducción de capas intermedias en una red neuronal denominadas *capas ocultas*. Estas redes neuronales con una o varias capas ocultas reciben el nombre de *redes neuronales multicapa*.

Se definen como composición de redes neuronales simples, por lo que tienen la misma estructura, es decir,

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}.$$

Siguiendo el esquema de la figura 3.2, si tomamos por ejemplo la salida de la primera capa C_j^1, donde $1 \leq j \leq k_l$, se calcula como

$$C_j^1 = f_j^1(x) = h_j^1 \left(\sum_{i=1}^{n} w_{j,i}^1 \cdot x_i + b_j^1 \right),$$

donde f_j^1 es la función de dicha neurona, h_j^1 la función de activación, $w_{j,i}^1$ los pesos y b_j^1 el sesgo. En el caso de la segunda capa, las salidas se calculan con la fórmula

$$C_j^2 = f_j^2(x) = h_j^2 \left(\sum_{i=1}^{k_1} w_{j,i}^2 \cdot C_i^1 + b_j^1 \right).$$

En cuanto al entrenamiento de las redes neuronales multicapa, se lleva a cabo el mismo

proceso que con las simples, esto es, se trata de minimizar la función

$$L(w,b) = \frac{1}{|E|} \cdot \sum_{x \in E} (\tilde{f}(x) - f(x))^2,$$

donde w es el vector de todos los pesos de la red y b es el vector de los sesgos de la neurona. Para ello se emplea de forma similar el método del descenso del gradiente, es decir, iterar la función

$$(\tilde{w}, \tilde{b}) = (w,b) - \eta \cdot \nabla L(w,b)$$

hasta obtener un error lo suficientemente pequeño.

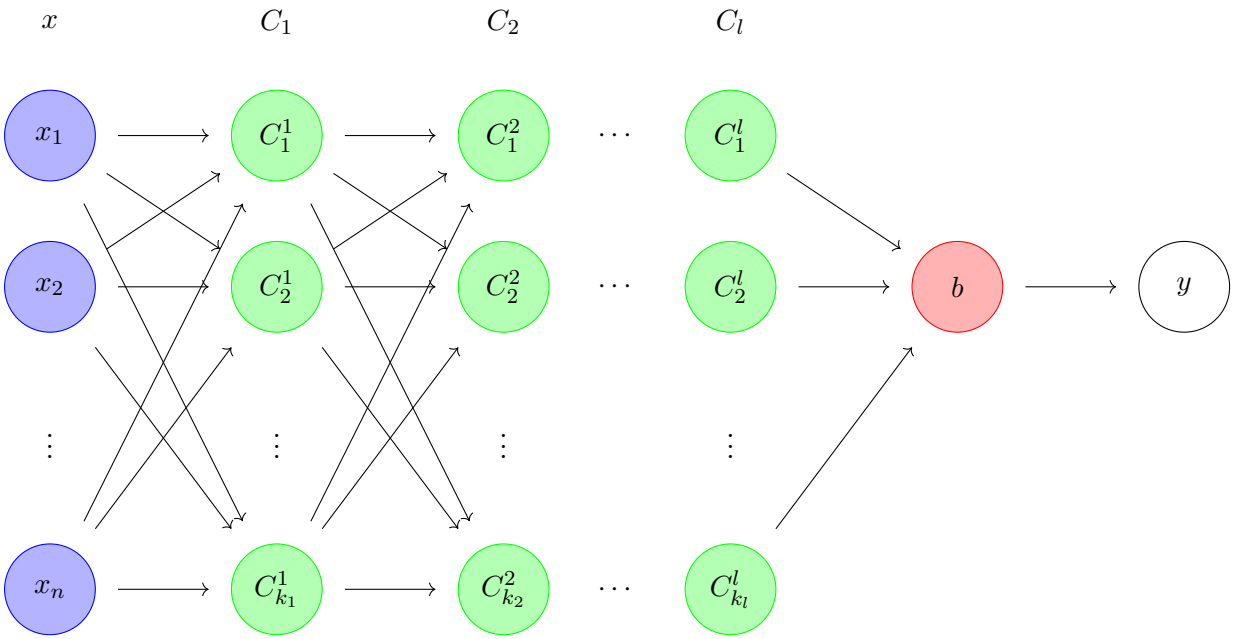

Figura 3.2: Diagrama de una red neuronal multicapa de la forma $f : \mathbb{R}^n \longrightarrow \mathbb{R}$. Esta compuesta de l capas ocultas.

3.2.3 Redes neuronales convolucionales

Las redes neuronales convolucionales son un tipo de de red neuronal que está diseñada para procesar y analizar datos con una estructura cuadrícula, como por ejemplo imágenes o vídeos. A día de hoy es una de las herramientas más eficientes para esta tarea.

La estructura consta de dos partes principales:

1. *Base:* Se encarga de extraer las características de una imagen. Para ello hace uso de capas convolucionales y capas de agrupamiento que explicaremos más adelante. Es común usar bases ya preentrenadas pues reduce el tiempo de entrenamiento y tiene buenos resultados.

2. *Cabeza:* Clasifica las características extraídas anteriormente. Se entrena con las imágenes que queremos clasificar.

Una *capa convolucional* esta compuesta de varios *filtros convolucionales*. Cada uno de estos determina si se presenta una particular característica en la imagen. Un filtro convolucional consiste en una multiplicación de una matriz a distintos conjuntos de pixeles de una imagen y luego la suma de estos. Por ejemplo, si tenemos una imagen dada por los píxeles

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix},$$

al aplicar el filtro

$$\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix},$$

obtenemos la imagen

$$\begin{bmatrix} \alpha a_{1,1} + \beta a_{1,2} + \gamma a_{2,1} + \delta a_{2,2} & \cdots & \alpha a_{1,n-1} + \beta a_{1,n} + \gamma a_{2,n-1} + \delta a_{2,n} \\ \alpha a_{2,1} + \beta a_{2,2} + \gamma a_{3,1} + \delta a_{3,2} & \cdots & \alpha a_{2,n-1} + \beta a_{2,n} + \gamma a_{3,n-1} + \delta a_{3,n} \\ \vdots & \ddots & \vdots \\ \alpha a_{m-1,1} + \beta a_{m-1,2} + \gamma a_{m,1} + \delta a_{m,2} & \cdots & \alpha a_{m-1,n-1} + \beta a_{m-1,n} + \gamma a_{m,n-1} + \delta a_{m,n} \end{bmatrix}.$$

Este ejemplo no es la única forma de aplicar el filtro, ya que puede variar la dimensión de la matriz de filtro o el tamaño del "salto" de un grupo de pixeles a otro. Tras ser filtrada la imagen, esta será evaluada por una función de activación, siendo la más común la función ReLU vista anteriormente. En otras palabra, sustituye los valores negativos asociados a los pixeles por el 0.

Un ejemplo de filtro podría ser la matriz

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix},$$

el cual filtrará las componentes horizontales de la imagen.

El siguiente paso es condensar la información que hemos extraído en una imagen más pequeña, es decir, reducir su dimensión. Este es el objetivo de las capas de agrupamiento, las cuales dividen la imagen filtrada en celdas, que suelen tener un tamaño de 2×2, y se realiza una operación sobre esta.

Una operación de agrupamiento muy utilizada es la del máximo, la cual toma el máximo valor de la celda seleccionada. Las principales ventajas de este agrupamiento es que conserva las características más importante y es robusta ante las traslaciones y al ruido. Su principal desventaja es que pierde información que en algunos casos tiene gran importancia.

Otra operación bastante usada es la media, cuya función es tomar el valor medio de los píxeles de la celda. Se suele utilizar si se desea conservar una información general sobre la región o si deseamos suavizar alguna característica. El problema de esta operación es que diluye las características más relevantes y es sensible al ruido.

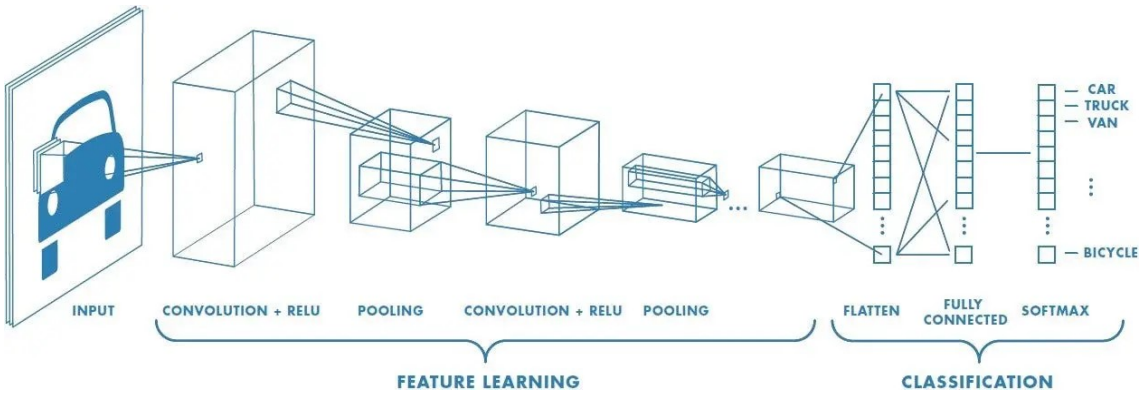

Figura 3.3: Imagen tomada de [SS]. En este ejemplo se puede apreciar la estructura de una red neuronal convolucional. La primera parte alterna capas de convolución con capas de agrupamiento reduciendo la dimensionalidad de la imagen pero extrayendo múltiples características. Tras un "aplanamiento", la imagen se ha transformado en un vector que es la entrada de una red neuronal multicapa. Finalmente tiene lugar una clasificación a partir de las características.

Resumiendo, la base de una red neuronal convolucional consiste en una serie de capas de convolución y agrupamiento que toma como entrada una imagen y devuelve un vector. En algunos casos la base tiene como última operación un "aplanamiento", el cual

consiste transformar un conjunto de matrices en un vector. Como hemos mencionado anteriormente, la base de una red neuronal convolucional se incorporar ya preentrenada a la red completa, es decir, los coeficientes de las capas de convolución se han calculado anteriormente.

La segunda parte de la red neuronal convolucional es la cabeza. Esta toma como entrada el vector salida de la base y devuelve otro vector. Esta compuesta de varias redes neuronales multicapa las cuales se entrenan a partir de un conjunto de entrenamiento.

3.3 Bosques aleatorios

Los bosques aleatorios (Random Forests) son un tipo de modelo de aprendizaje automático basado en el uso de múltiples árboles de decisión para realizar tareas de clasificación, regresión u otras tareas en aprendizaje supervisado. Es de gran utilidad a la hora de clasificar vectores de dimensión finita ya que es muy rápido computacionalmente y suele tener muy buenos resultados. Antes de definir este concepto, estudiemos el muestreo con remplazo y los árboles de decisión.

3.3.1 Muestreo con remplazo

El *muestreo con remplazo* (bootstrap), es un método de inferencia que se usa para estimar alguna característica sobre los datos a partir de una muestra, como puede ser la media, mediana, varianza... Es un método que consiste en generar múltiples muestras simuladas a partir de una muestra original. Cada muestra obtenida se denominará *muestra bootstrap*.

Este algoritmo comienza tomando nuestra muestra de datos original X que tenemos disponible de tamaño n. A partir de esta, se genera una muestra bootstrap tomando n elementos del conjunto X con remplazo que denotaremos $\tilde{X}_i$, es decir, puede haber elementos de X que se repitan y otros que a su vez no pertenezcan a $\tilde{X}_i$. Se realiza este proceso en B ocasiones obteniendo B muestras bootstrap, siendo en general un número alto. A partir de cada $\tilde{X}_i$ se puede calcular la característica que nos interese y estimar la del conjunto total.

Las principales ventajas del muestreo con remplazo es su sencillez de implementación y que no depende de ningún parámetro. Los problemas que puede conllevar es que puede ser

computacionalmente costoso y dependemos de que la muestra original sea representativa de la población total.

Ejemplo 3.2. Supongamos que tenemos una muestra de datos dada por

$$X = \{3.25, 2.47, 2.64, 3.12, 4.89\}.$$

Podemos tomar, por ejemplo las siguientes muestras bootstrap:

$$\tilde{X}_1 = \{2.47, 3.25, 3.12, 3.25, 4.89\}$$

$$\tilde{X}_2 = \{2.64, 3.25, 2.64, 3.12, 3.25\}$$

$$\tilde{X}_3 = \{4.89, 2.47, 4.89, 2.64, 3.12\}.$$

A partir de ellas podemos tratar de estimar alguna característica de la población total, aunque en este caso no será muy preciso ya que el número de muestras bootstrap es muy pequeño.

3.3.2 Árboles de decisión

Los árboles de decisión es un tipo de modelo de aprendizaje automático utilizado para tareas de clasificación y de regresión de vectores de dimensión finita. Su funcionamiento consiste en dividir de forma repetida el conjunto datos utilizando reglas definidas a partir de las características de los datos. Tras todas las divisiones, a los subconjuntos obtenidos se les asigna una predicción.

Los elementos que definen un árbol de decisión son los siguientes:

1. *Raíz*: Es el nodo inicial del árbol que contiene el conjunto total de datos de los que disponemos. Este nodo será dividido según la característica que mejor separe los datos.

2. *Nodos internos*: Representan las decisiones que se toman para dividir los datos. Se realizan a partir de una desigualdad respecto a cierta característica de los vectores.

3. *Ramas*: Conecta los nodos entre sí y representa los resultados de las decisiones.

4. *Hojas*: Son los nodos finales tras sucesivas divisiones. Cada hoja representa un conjunto de datos, el cual tendrá asignado un grupo o un valor predicho.

El siguiente paso es entrenar el árbol de decisión, es decir, elegir las características y los discriminantes para cada ramificación. El principal problema es saber si una ramificación está bien hecha, para ello haremos uso de la impureza de Gini.

Definición 3.3. La *impureza de Gini* es una medida de cuan puro es un subconjunto obtenido tras una decisión. Se calcula mediante la fórmula:

$$Gini(S) = 1 - \sum_{i \in G} p_i^2,$$

donde G es el conjunto de grupos de clasificación y p_i es el porcentaje que pertenece a dicho grupo en el subconjunto S.

Dicho esto, el entrenamiento comienza con el conjunto de datos en el nodo inicial, el cual lo dividirá en dos subconjuntos minimizando la impureza de Gini de cada uno y seleccionando distintas características y valores para las desigualdades. Se repetirá este proceso con cada ramificación de forma iterada hasta que se alcance una impureza muy pequeña o no se pueda mejorar, convirtiendo estos nodos en hojas.

A este entrenamiento se pueden añadir previamente unos parámetros, siendo los principales la impureza mínima para llevar a cabo la ramificación o la profundidad máxima de las ramas para que no se lleve a cabo un sobreajuste. Tras el entrenamiento se puede realizar una poda, la cual consiste en eliminar aquellos nodos que no aporten mucha mejora en la precisión, reduciendo el sobreajuste y disminuyendo la complejidad del modelo.

3.3.3 Bosques aleatorios

Finalmente, podemos definir el algoritmo de aprendizaje automático de bosque aleatorio o randomo forest. Dado un conjunto de datos X de n elementos, este método consiste en tomar B muestras bootstrap de tamañao n siendo B un número grande. Tras esto, con cada subconjunto de vectores $\tilde{X}_i$ elegimos de forma aleatoria unas características concretas con las cuales entrenaremos un árbol de decisión. Finalmente obtendremos B árboles de decisión que difieren entre sí debido a la aleatoriedad de las características elegidas y de los elementos que lo entrenan. En el caso de que sea un modelo de clasificación se elegirá aquella etiqueta que sea mayoritariamente predicha por los árboles, mientras que en un modelo de regresión será el promedio.

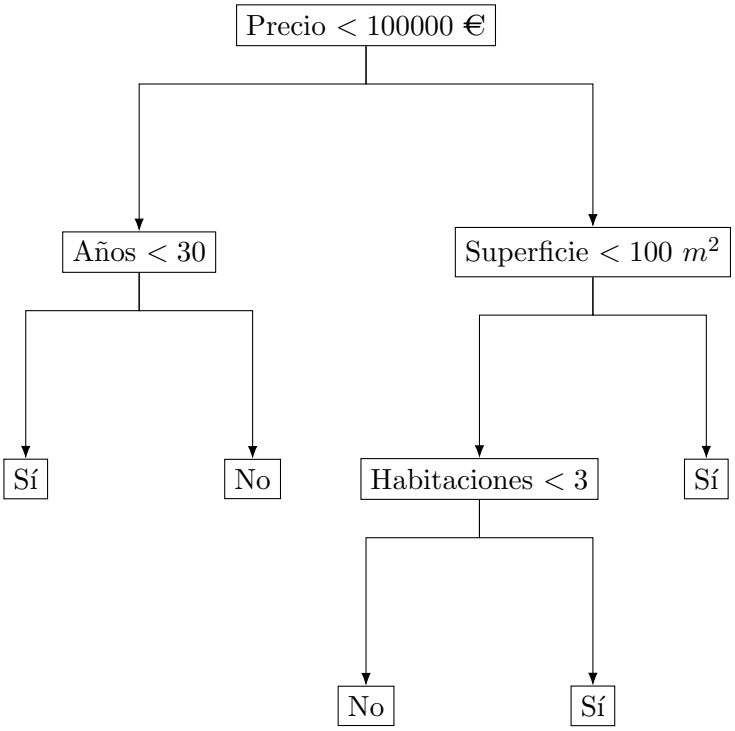

Figura 3.4: Ejemplo de árbol de decisión. En este caso podemos ver algunos atributos a la hora de comprar una vivienda que son importantes para tomar la decisión de comprarla o no. Los atributos en cuestión son el precio, la antigüedad de la vivienda, su superficie y el número de habitaciones.

Las principales ventajas de este modelo es su gran robustez, ya que al combinar las predicciones de varios árboles es muy difícil que tengamos sobreajuste. También es una herramienta muy buena para trabajar con datos con alta dimensionalidad. La principal desventaja de este modelo es que requiere gran cantidad de recursos en cuanto a memoria y tiempo, ya que hace uso de gran cantidad de árboles de decisión simultáneamente.

Capítulo 4

Detección de Alzheimer a partir de señales EEG

En este capítulo estudiaremos un caso real en el que podemos hacer uso de las herramientas definidas a lo largo de la memoria en los capítulos anteriores. Analizaremos distintos electroencefalogramas (señales EEG) para tratar de distinguir si un paciente padece Alzheimer o no. Para ello usaremos dos técnicas distintas haciendo uso de la topología de datos. La primera consiste en calcular las características persistentes de dichas señales. Tras esto, entrenaremos un modelo para que clasifique las señales de la forma más precisa posible haciendo uso del bosque aleatorio definido en el capítulo anterior. La segunda técnica consiste en entrenar una red neuronal convolucional a parir de las imágenes persistentes obtenidas de las señales EEG. La entrenaremos dividiendo los casos en si el paciente padece Alzheimer o no, evaluando finalmente un conjunto de test para probar la eficacia del método. Finalmente compararemos ambos modelos para averiguar que técnica es más efectiva. Para el análisis de estas señales hemos hecho uso del paquete mne de Python.

4.1 Obtención y filtro de señales EEG

La obtención de señales EEG, consiste en un cuidadoso proceso para captar la actividad eléctrica del cerebro. El primer paso consiste en limpiar el cuero cabelludo del paciente eliminando aceites, células muertas y demás sustancias que puedan interferir en la conductividad eléctrica. Tras esto se coloca el casco de electrodos sobre el cuero cabelludo del

paciente. En el conjunto de datos que vamos a utilizar se coloca según el sistema internacional $10-20$, que es un estándar para colocar los electrodos en posiciones específicas que corresponden a distintas regiones del cerebro.

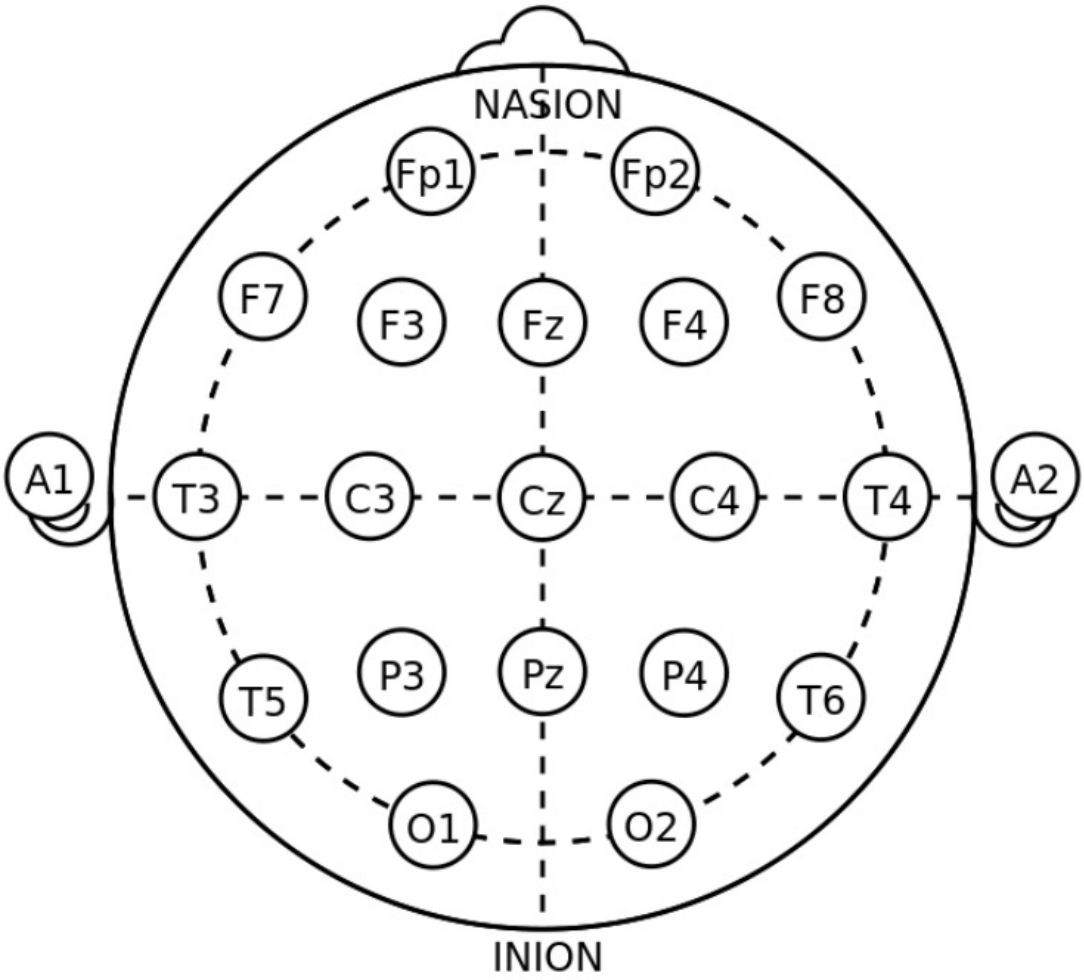

Figura 4.1: Localización de los electrodos en el cuero cabelludo para la detección de señales eléctricas según el sistema internacional $10-20$.

Los electrodos $A1$ y $A2$ no están presentes en el conjunto de datos ya que se usan como referencia. Como consecuencia, cada paciente tiene asociado 19 electrodos ordenados de la siguiente forma:

$$[Fp1, Fp2, F3, F4, C3, C4, P3, P4, O1, O2, F7, F8, T3, T4, T5, T6, Fz, Cz, Pz].$$

Al estar ordenados de esta forma es más fácil acceder a cada electrodo.

El conjunto de datos usado en este trabajo proviene de [AKT]. Contiene un total de 88 pacientes, los cuales se dividen en tres categorías: pacientes que padecen deficiencia frontotemporal, alzheimer y pacientes sanos. En nuestro caso hemos hecho uso únicamente los últimos dos grupos, descartado además aquellos datos que no se han procesado bien.

Cada paciente tendrá asociado una señal EEG, su genero, su edad, la enfermedad que padece y una puntuación del 0 al 30 del test internacional Mini-Mental State Examination (MMSE). Este examen indica que cuanto menor sea la puntuación, mayor será el deterioro cognitivo del paciente. Todas las señales tienen una duración mínima de $13,5$ minutos.

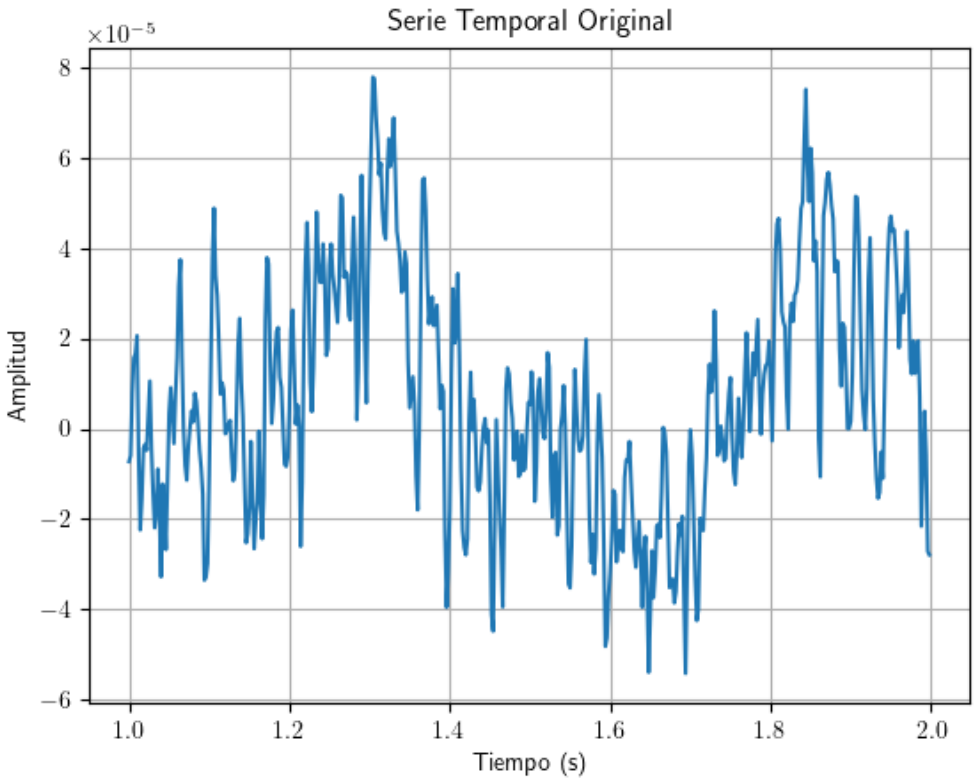

Figura 4.2: Ejemplo de la señal EEG detectada en el paciente 43 del conjunto de datos durante 1 segundo. En este caso se trata del electrodo $T3$.

El problema de este tipo de señales, es que para detectar la actividad cerebral los electrodos son muy sensibles a las señales eléctricas y por lo tanto a cualquier perturbación. Es por eso que es necesario eliminar el ruido de la señal tomando aquellas frecuencias que

nos interesan. Para ello usamos la transformada discreta de Fourier (DFT).

Dada una serie temporal $X = (x_0, x_1, \ldots, x_{N-1}) \in \mathbb{R}^N$, supongamos que queremos filtrar X y eliminar las frecuencias que no se encuentren en un rango entre f_{min} y f_{max}. En primer lugar calculamos los coeficientes complejos de Fourier a partir de la siguiente fórmula:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{2\pi i}{N} kn}, \quad k = 0, 1, \ldots, N-1.$$

Sea ahora f_{serie} la frecuencia por segundo de la serie temporal, siendo la frecuencia del conjunto de datos que hemos tomado de 500. Calculamos por lo tanto

$$k_{min} = \left\lceil \frac{f_{min} \cdot N}{f_{serie}} \right\rceil$$

y

$$k_{max} = \left\lfloor \frac{f_{max} \cdot N}{f_{serie}} \right\rfloor.$$

A continuación, definimos los nuevos coeficientes de Fourier filtrando aquellos que nos interesan, es decir,

$$X'_k = \begin{cases} X_k & \text{si } k \in [k_{min}, k_{max}], \\ 0 & \text{en caso contrario.} \end{cases}$$

Por último, se calcula la transformada inversa discreta de Fourier (IDFT) a partir de los nuevos coeficientes obtenidos de la siguiente forma:

$$x'_k = \frac{1}{N} \cdot \sum_{n=0}^{N-1} X'_n \cdot e^{\frac{2\pi i}{N} kn}, \quad k = 0, 1, \ldots, N-1.$$

Tenemos por lo tanto, que la serie temporal X filtrada está dada por

$$X' = (x'_0, x'_1, \cdots, x'_{N-1}).$$

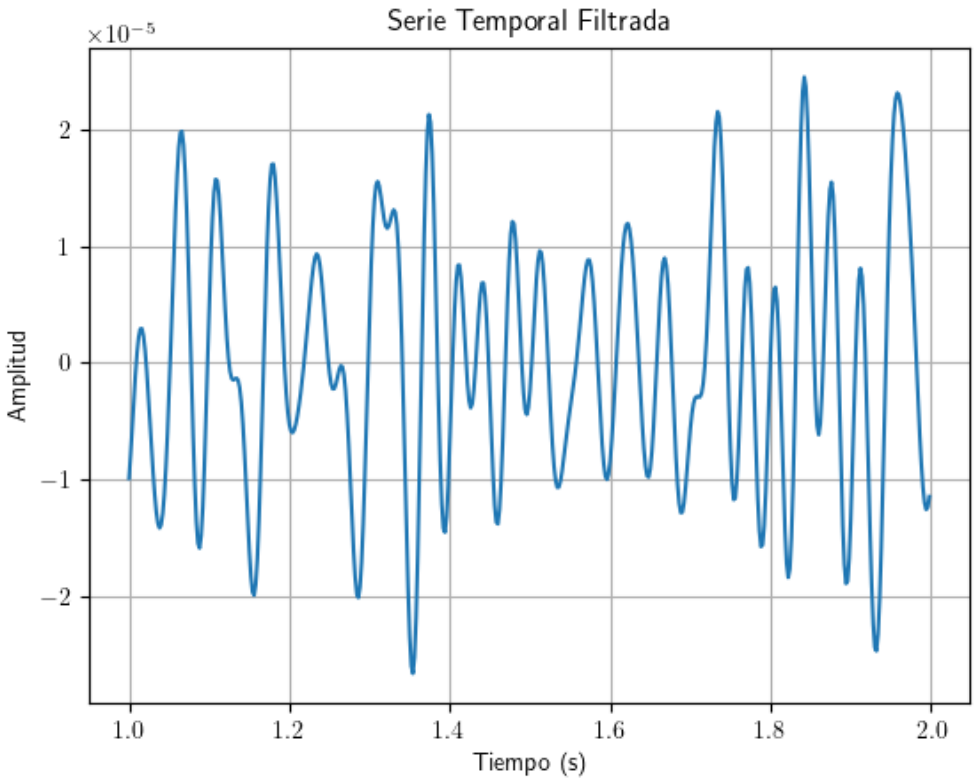

Figura 4.3: Ejemplo de la señal EEG 4.2 filtrada en el rango de frecuencias $[8, 30]$.

4.2 Modelos de detección de Alzheimer

En esta sección vamos a probar distintos métodos para la predicción de Alzheimer a partir de las señales EEG haciendo uso de los conceptos definidos anteriormente. Se harán pruebas con distintas frecuencias y sensores hasta obtener un modelo con una alta precisión.

Para entrenar y evaluar los modelos hacemos uso de la validación cruzada, ya que disponemos de pocos pacientes (52), y esto nos ayuda a mejorar el modelo. La validación cruzada consiste en dividir el conjunto de datos en k subconjuntos del mismo tamaño, en nuestro caso $k = 5$, e usar $k - 1$ subconjuntos para entrenar el modelo y el restante para su evaluación. Al repetir este proceso k veces usando cada vez un subconjunto distinto para la evaluación, obtenemos k modelos con los cuales formaremos el modelo final para

nuevos datos.

4.2.1 Evaluación de modelos de clasificación binaria

Al tener un modelo entrenado, si lo evaluamos sobre el conjunto test, cada elemento de este pertenecerá a una de las siguientes categorías:

1. *Verdadero negativo (TN)*: El modelo predice que un paciente del grupo de control pertenece a dicho grupo.
2. *Verdadero positivo (TP)*: Predice que el paciente que padece Alzheimer pertenece a dicho grupo.
3. *Falso negativo (FN)*: Asigna a un paciente que padece Alzheimer el grupo de control, fallando en la predicción.
4. *Falso positivo (FN)*: El modelo predice que un paciente del grupo de control pertenece al grupo de Alzheimer, errando como consecuencia.

Definidas estas categorías, vamos a usar los siguientes índices para evaluar la eficacia de un modelo:

$$\text{Exactitud } = \frac{TP + TN}{TP + TN + FP + FN}$$

Exactitud: Peor valor = 0. Mejor valor = 1.

$$\text{Índice F1 } = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

Índice F1: Peor valor = 0. Mejor valor = 1.

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

Sensibilidad: Peor valor = 0. Mejor valor = 1.

$$\text{Especificidad } = \frac{TN}{TN + FP}$$

Especificidad: Peor valor = 0. Mejor valor = 1.

$$\text{Valor predictivo positivo (PPV)} = \frac{TP}{TP + FP}$$

PPV: Peor valor = 0. Mejor valor = 1.

$$\text{Valor predictivo negativo (PPN)} = \frac{TN}{TN + FN}$$

PPN: Peor valor = 0. Mejor valor = 1.

El problema de estos índices es que si hay una gran desigualdad en el número de datos de cada clase, es difícil apreciar la precisión del modelo a la hora de predecir elementos de todas las clases. Por ello, introducimos los conceptos de *coeficiente de correlación de Matthews* (MCC) y el *kappa de Cohen de una matriz de confusión de clasificación binaria* (κ):

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}}$$

MCC: Peor valor = -1. Mejor valor = 1.

$$\kappa = \frac{2 \cdot (TP \cdot TN - FN \cdot FP)}{(TP + FP) \cdot (FP + TN) + (TP + FN) \cdot (FN + TN)}$$

κ: Peor valor = -1. Mejor valor = 1.

4.2.2 Detección de Alzheimer con características persistentes

A continuación explicaremos los modelos que hemos entrenado para detectar Alzheimer a partir de las características persistentes, definidas en la sección 2.2.

El primer paso ha sido la extracción de las características persistentes de los paciente. Para ello, hemos tomado de los canales $T5$, $T6$ y Pz de las señales EEG un intervalo de 5 minutos de cada paciente. Después, a partir de la transformada discreta de Fourier, hemos filtrado las señales en el intervalo de frecuencias $[8, 12]$, también denominadas ondas alfa. Tras dicho análisis hemos obtenido las características persistentes de las ondas a partir de una función de filtración, proceso que computacionalmente es algo extenso en cuanto a tiempo ya que ha tomado unos 15 minutos para las 3 ondas de cada paciente.

En cuanto a la obtención de códigos de barras a partir de la función de filtración hemos usado dos métodos. En el primero, el inicio de los intervalos tomaban siempre como nacimiento el 0 mientras que el final lo indicaba el tiempo de vida. En el segundo método los intervalos indicaban el nacimiento y el final de la característica. Aunque finalmente nos hemos decantado por el segundo método, es verdad que con algunas frecuencias y canales concretos funciona mejor el primero, pero no es el caso del modelo definitivo.

Ya obtenidas las características persistentes, hemos probado varios modelos. Uno de ellos ha sido el modelo de *centro de masa*. En nuestro caso hemos seleccionado el canal $T5$ y a partir del conjunto de entrenamiento tomamos los subconjuntos de pacientes que padecen Alzheimer y los de control, dados por Λ_A y Λ_C, respectivamente. Definimos entonces las siguientes características persistentes

$$\lambda_A := \frac{1}{|\Lambda_A|} \cdot \sum_{\lambda \in \Lambda_A} \lambda \quad \text{y} \quad \lambda_C := \frac{1}{|\Lambda_C|} \cdot \sum_{\lambda \in \Lambda_C} \lambda.$$

Tras esto, dada una característica persistente $\tilde{\lambda}$ del conjunto de test, haremos la predicción de que dicho paciente padece Alzheimer si

$$\|\lambda_A - \tilde{\lambda}\|_2 < \|\lambda_C - \tilde{\lambda}\|_2.$$

En caso contrario la predicción será que no lo padece.

Con este modelo se ha obtenido la matriz de confusión 4.4 y los siguientes coeficientes:

	Centro de masa
Exactitud	0.6730
Índice F1	0.7017
Sensibilidad	0.7692
Especificidad	0.5769
PPV	0.6452
PPN	0.7143
MCC	0.3527
κ	0.3462

Table 4.1: Coeficientes del modelo de centro de masa.

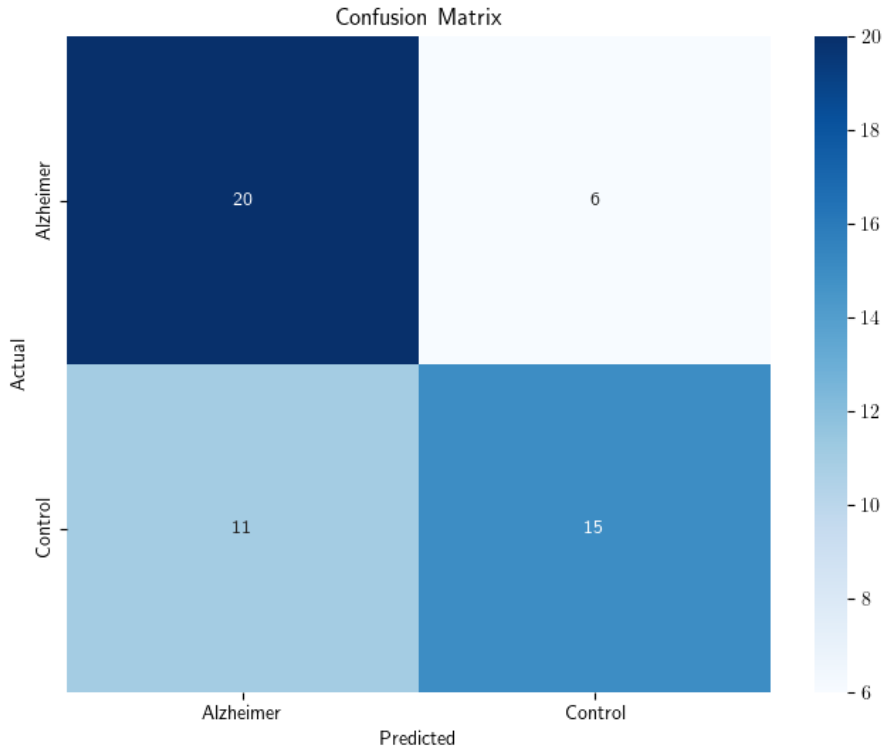

Figura 4.4: Matriz de confusión del modelo de centro de masas en el canal $T5$ y frecuencias $[8, 12]$.

Como podemos apreciar, hay una diferenciación entre las dos clases posibles, pero la precisión no es lo suficientemente alta para tomar como útil este modelo ya que tiene una alta tasa de error.

El siguiente modelo que hemos probado está basado en la técnica de bosques aleatorios vista en la sección del capítulo anterior 3.3. El principal problema que hemos encontrado es que las características persistentes son vectores de dimensión infinita, mientras que el algoritmo de bosques aleatorios solo se puede usar en vectores de dimensión finita por su propia definición. Para ello, hemos asociado una matriz a cada característica persistente, donde las columnas representan una discretización de la función a partir de una malla de 500 elementos, mientras que las columnas los niveles de persistencia no nulos, es decir, los $\lambda(k, t)$ para $k \in \mathbb{N}$.

Otro problema es que no todas las matrices tienen el mismo tamaño debido a que no todas las características persistentes tienen el mismo número de niveles de persistencia no nulos. Por ello se ha tomado aquella matriz con mayor número de filas y se han añadido filas de ceros a las demás matrices hasta llegar a este número. Tras este proceso, se han concatenado las filas de la matriz obteniendo un vector.

En cuanto al mejor modelo obtenido mediante este método se ha realizado lo siguiente. En primer lugar, para cada paciente, se han calculado las matrices asociadas a las características persistente de los sensores $T5$, $T6$ y Pz. Tras obtener todas las matrices del mismo tamaño $M(T5)$, $M(T6)$ y $M(Pz)$, hemos sumado de forma ponderada las matrices asociadas a los sensores mencionados de la siguiente forma:

$$3 \cdot M(T5) + 3 \cdot M(T6) + 2 \cdot M(Pz).$$

Asociamos de esta forma el vector obtenido al concatenar las filas de dicha matriz a su paciente correspondiente, pudiendo así aplicar el método de bosque aleatorio. Como parámetros hemos usado 500 árboles de decisión ya que así tarda menos de un minuto y tiene el mismo resultado que un número superior.

Usando este modelo, se ha generado la matriz de confusión 4.5 y se han calculado los siguientes coeficientes:

	Bosque aleatorio
Exactitud	0.8654
Índice F1	0.8679
Sensibilidad	0.8846
Especificidad	0.8462
PPV	0.8519
PPN	0.8800
MCC	0.7313
κ	0.7308

Table 4.2: Coeficientes del modelo de bosque aleatorio.

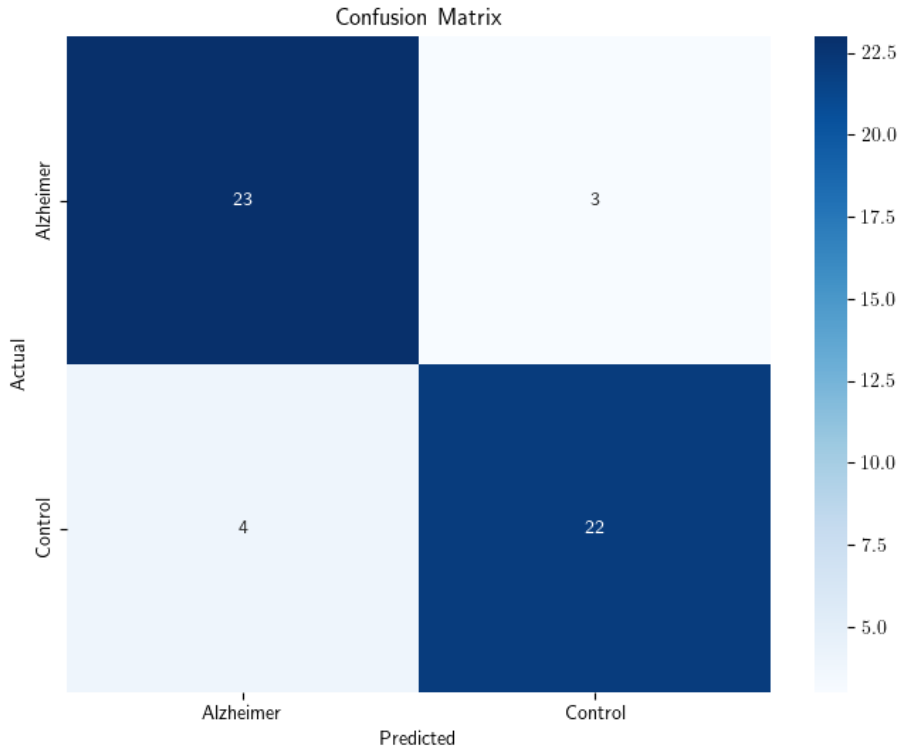

Figura 4.5: Matriz de confusión del modelo de bosque aleatorio. Se hace uso de los sensores $T5$, $T6$ y Pz en el rango de frecuencias $[8, 12]$.

El principal problema de este modelo es que los vectores que usa son de un tamaño mayor a 1 millón de elementos, luego contienen demasiada información que no necesitamos. Para intentar solucionar este problema reducimos el tamaño de las matrices usando análisis de componentes principales, pero no mejoramos los resultados.

4.2.3 Detección de Alzheimer con imágenes persistentes

Otro tipo de modelos que hemos utilizado para tratar de detectar Alzheimer los hemos basado en las imágenes persistentes estudiadas en el capítulo 2. Estos modelos consisten en clasificar las imágenes persistentes asociadas al canal $T5$ de las señales EEG extraídas de los pacientes en el rango de frecuencias $[8, 12]$, al igual que los demás modelos.

Tras varias pruebas con distintas estructuras de redes neuronales, no hemos encontrado ninguna red que clasifique de forma optima las imágenes. Esto se debe a la dificultad de encontrar patrones distintivos de cada clase en este problema de clasificación. Además, al modificar el factor aleatorio que determina el modelo, las predicciones son totalmente distintas.

A continuación, podemos apreciar la matriz de confusión y los coeficientes obtenidos al evaluar un modelo de este tipo:

	CNN
Exactitud	0.5577
Índice F1	0.4888
Sensibilidad	0.3333
Especificidad	0.5455
PPV	0.5789
PPN	0.5455
MCC	0.1198
κ	0.1154

Table 4.3: Coeficientes del modelo de redes neuronales convolucionales.

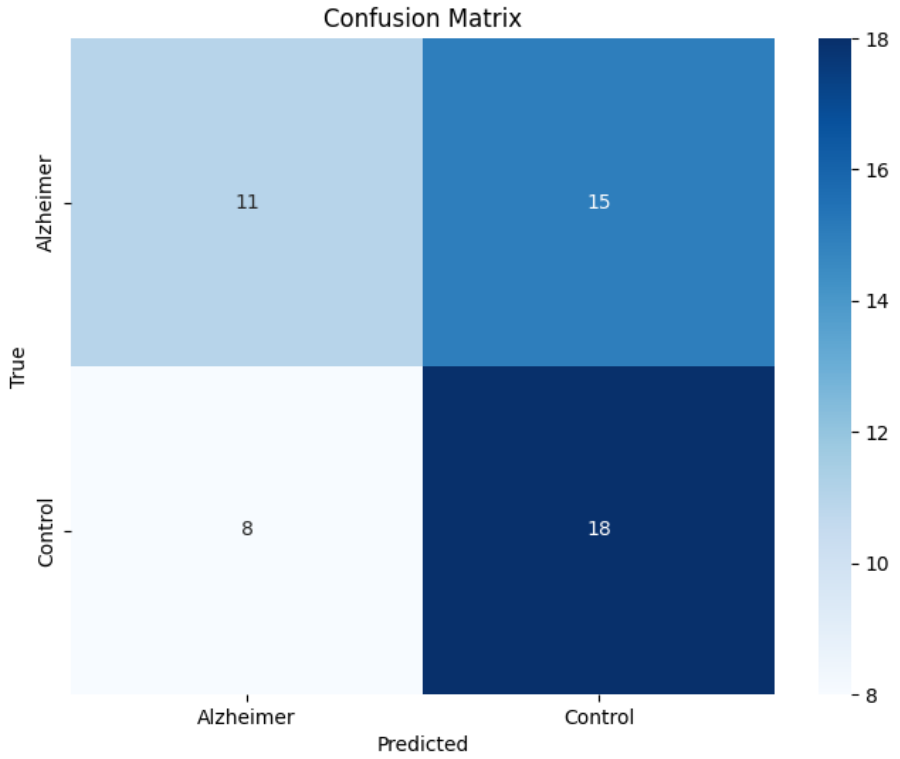

Figura 4.6: Matriz de confusión de un modelo de redes neuronales convolucionales. Se hace uso del sensor $T5$ en el rango de frecuencias $[8, 12]$. Como podemos apreciar, este modelo no es capaz de realizar una buena clasificación.

4.3 Conclusiones y trabajo futuro

En esta memoria hemos introducido los conceptos de características persistentes e imágenes persistentes. Tras esto, hemos desarrollado varios modelos relacionados con estas herramientas para la clasificación de series temporales.

Si comparamos los tres modelos mencionados en el trabajo, tenemos los siguientes coeficientes:

	Centro de masa	**Bosque aleatorio**	**CNN**
Exactitud	0.6730	**0.8654**	0.5577
Índice F1	0.7017	**0.8679**	0.4888
Sensibilidad	0.7692	**0.8846**	0.3333
Especificidad	0.5769	**0.8462**	0.5455
PPV	0.6452	**0.8519**	0.5789
PPN	0.7143	**0.8800**	0.5455
MCC	0.3527	**0.7313**	0.1198
κ	0.3462	**0.7308**	0.1154

Table 4.4: Comparativa de los coeficientes de los modelos incluidos en esta memoria para la predicción de Alzheimer a partir de señales EEG.

Se puede apreciar que los modelos que hacen uso de características persistentes son bastante más precisos que el basado en imágenes persistentes. Esto se puede deber a que la información que aporta una característica persistente es bastante superior a la que contienen las imágenes. Lo podemos apreciar en el tamaño de los archivos ya que la característica persistente asociada a un canal de un paciente tiene un tamaño de más de 10000 kB, mientras que las imágenes tienen un tamaño de poco más de 1 kB. Esto no implica que las imágenes persistentes no sean útiles, ya que se pude hacer uso de estas en otro tipo de modelos que tengan patrones más claros y conjuntos de datos más grandes.

En cuanto al futuro trabajo, creo que se pude tratar de mejorar más aún el modelo basado de bosque aleatorio. Se trata al final de un modelo entrenado únicamente a partir de los datos de 52 pacientes, por lo que si se aumenta este conjunto se podría aumentar la precisión. Otra forma para intentar mejorarlo sería tratar de hacer uso de otra frecuencias de las señales EEG, ya que hay muchos intervalos posibles. Por último, otro camino a seguir para intentar mejorar el modelo sería probar las distintas combinación de las características persistentes asociadas a los canales de los pacientes. Hay un total de 19 canales distintos, luego al combinarlos mediante ponderaciones se obtienen muchos modelos distintos.

Tras esto daremos por terminada la memoria, en los apéndices podemos encontrar el código de las funciones utilizada para el desarrollo de los modelos mostrados.

Appendix A

Funciones utilizadas en python para el desarrollo de la memoria

A continuación, adjunto un repositorio Github con el código de los programas usados en el siguiente enlace:

`https://github.com/jo12n/TFM-caracteristicas-persistentes`

A.1 Función de filtración

```
def barcode2(serie):
        tam_serie = len(serie)
        ind_ord = np.argsort(serie)
        m = ind_ord[0]
        barcode = [[serie[m],serie[m]]]
        ind_bar = {m:0}
                for i in ind_ord[1:]:
                ind_car = []
                ven_izq = i-1
                ven_der= i+1
                while ven_izq >= 0 and serie[ven_izq] < serie[i
                    ]:
```

```
                ven_izq = ven_izq-1
            if ven_izq < 0:
                ven_izq = 0
            while ven_der < tam_serie and serie[ven_der] <
                serie[i]:
                ven_der = ven_der +1
            if ven_der >= tam_serie-1:
                ven_der = tam_serie
            for intervalo in list(set(range(ven_izq,ven_der)
                ).intersection(set(ind_bar.keys()))):
                if max(serie[min([i,intervalo]):max([i,
                    intervalo])]) <= serie[i]:
                        ind_car.append(intervalo)
            if len(ind_car) == 0:
                barcode.append([serie[i],serie[i]])
                ind_bar.update({i:len(ind_bar)})

            elif len(ind_car) == 1:
                continue
            else:
                p = min([serie[ind_bar[w]] for w in
                    ind_car])
                for w in ind_car:
                    if barcode[ind_bar[w]][0] ==
                        barcode[ind_bar[w]][1] and
                        serie[ind_bar[w]] > p:
                            barcode[ind_bar[w]][1] =
                                serie[i]
    barcode[0][1] = max(serie)
    return np.array(barcode)
```

A.2 Función de extracción de características persistentes

```
# Leer el archivo .set usando MNE
```

```
raw = mne.io.read_raw_eeglab(archivo_set, preload=True)

# Indice del canal
idx_canal = 18

# Extraer los datos del canal
datos_canal, tiempos = raw[idx_canal, :]

# Guardar los datos del canal en un vector

vector_datos_canal = datos_canal[0,50000:200000]

N=150000
fs = 500
T = 1/fs
x= tiempos[50000:200000]
#x = np.linspace(0.0, N*T, N, endpoint=False)
yf = np.fft.fft(vector_datos_canal)
xf = np.fft.fftfreq(N, T)

frec_inf = 8  # Umbral de frecuencia en Hz
frec_sup = 12
mask_inf = np.abs(xf) >= frec_inf
mask_sup = np.abs(xf) <= frec_sup

yf_filtered = np.copy(yf)
yf_filtered = np.where(mask_inf, yf, 0)
yf_filtered = np.where(mask_sup, yf_filtered, 0)

# Reconstruir la senal temporal a partir de la transformada de
    Fourier filtrada

vector_datos_canal = np.fft.ifft(yf_filtered).real
barcode = barcode2(vector_datos_canal)
barcode = sorted(barcode, key=lambda x: x[1]-x[0], reverse=True)
```

```
barcode = barcode[50:]

ple = PersLandscapeApprox(dgms=[np.array(barcode)],hom_deg=0)

        with open(filename, "wb") as file:
pickle.dump(ple, file)
```

A.3 Función de extracción de imágenes persistentes

```
def imagen_pers(barcode,resolucion,guardado):
        for i in range(len(barcode)):
                barcode[i][1] = barcode[i][1] - barcode[i][0]
        max_nac = max([v[0] for v in barcode])
        min_nac = min([v[0] for v in barcode])
        max_TV = max([v[1] for v in barcode])
        min_TV = min([v[1] for v in barcode])
        rango_foto = max(max_nac-min_nac,max_TV-min_TV)
        imagen = Image.new('RGB', (resolucion,resolucion), '
           white')
        pixeles = imagen.load()
        matriz = np.zeros((resolucion, resolucion))
        k=0
        l=0
        paso = (1.2*rango_foto)/(resolucion-1)
        for i in np.arange(min_nac-0.1*rango_foto, min_nac+1.1*
           rango_foto, paso):
                for j in np.arange(min_TV-0.1*rango_foto, min_TV
                   +1.1*rango_foto, paso):
                        matriz[k,l] = crear_imagen(barcode,i,i+
                           paso,j,j+paso,rango_foto)
                        k=k+1
                k=0
                l=l+1
        maximal = np.amax(matriz)
```

```
        for i in range(resolucion):
                for j in range(resolucion):
                        pixeles[j,i] = (round(255*matriz[i,j]/
                            maximal),0,100)
        imagen = imagen.transpose(Image.FLIP_TOP_BOTTOM)
        imagen.save(guardado)
        return
```

Bibliografía

[AAM] H. Altaheri, G. Muhammad, M. Alsulaiman, S. Umar Amin, G. Ali Altuwaijri, W. Abdul, Mohamed A. Bencherif and M. Faisal *Deep learning techniques for classification of electroencephalogram (EEG) motor imagery (MI) signals: a review*, Neural Computing and Applications, 2023.

[ADQ] R. Ayala, E. Domínguez y A. Quintero, *Elementos de la teoría de homología clásica*, Universidad de Sevilla, 2002.

[AKT] Andreas Miltiadous and Katerina D. Tzimourta and Theodora Afrantou and Panagiotis Ioannidis and Nikolaos Grigoriadis and Dimitrios G. Tsalikakis and Pantelis Angelidis and Markos G. Tsipouras and Evripidis Glavas and Nikolaos Giannakeas and Alexandros T. Tzallas. *A dataset of EEG recordings from: Alzheimer's disease, Frontotemporal dementia and Healthy subjects*, OpenNeuro, 2023.

`https://openneuro.org/datasets/ds004504/versions/1.0.4`

[BCG] R. Belton, B. Cummins and T. Gedeon, *Extremal event graphs: A (stable) tool for analyzing noisy time series data*, Montana State University, 2022.

[BNG] N. Boutry, L. Najman y T. Geraud, *Some equivalence relation between persistent homology and morphological dynamics*, Noname manuscript, 2022.

[Bub1] P. Bubenik, *Statistical Topological Data Analysis using Persistence Landscapes*, Journal of Machine Learning Research 16 , 2015.

[Bub2] P.Bubenik, *The persistence landscape and some of its properties*, University of Florida, 2019.

[Chi] D. Chicco, *Ten quick tips for machine learning in computational biology*, BioDataMining, 2017.

[Ear] R. Earl, *Rings and Modules*, Hilary Term, 2018.

[EH] H. Edelsbrunner and John L. Harer, *Computational Topology. An introduction*, American Mathematical Society, 1972.

[Fra] T. Frahi, *Analyse Topologique des Donnees dans la Mecanique Numerique*, Arts et Metiers , 2021.

[KKS] Kavin Kumar D., V. Koliyat, R. Seenivasagan, S. Subbaiyan y S. Matheshwaran *TDA layer: Impact of persistent homology on the performance of Convolutional Neural Networks*, International Journal of Advance Research, Ideas and Innovations in Technology , 2021.

[Kwe] E. Kwessi, *Topological comparison of some dimension reduction methods using persistent homology on EEG data*, Eddy Kwessi, 2023.

[Lan] S. Lang, *Algebra*, Springer, 1993.

[Mun1] J. R. Munkres, *Elements of Algebric Topology*, Benjamin, 1984.

[Mun2] J. R. Munkres, *Topology*, Prentice Hall, 1975.

[Pen] G. Penide Calvo, *Homología persistente de redes complejas*, Universidade de Santiago de Compostela, 2016.

[Que] A. Quentin, *Analyse Topologique des Données pour l'étude de la maladie d'Alzheimer*, Centrale Nantes, 2023.

[Rav] N. Ravishanker, *An introduction to persistent homology for time series*, Wiley, 2020.

[RB] R. Rabadán y Andrew J. Blumberg, *Topological data analysis for genomics and evolution*, Cambrigde University Press, 2020.

[SCN] A. Som, H. Choi, K. Natesan Ramamurthy, Matthew P. Buman y P. Turaga, *PI-Net: A Deep Learning Approach to Extract Topological Persistence Images*, Conference on Computer Vision and Pattern Recognition Workshops, 2020.

[SS] Sumit Saha, *A guide to convolutional neural Networks. The ELI5 way*, 2018.

`https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/`

[WOC] Y. Wang, H. Ombao and Moo K.Chung, *Topological data analysis of single-trial electroencephalographic signals*, Springer, 1993.