



UNIVERSIDAD DE GRANADA

Facultad de Ciencias y Escuela Técnica Superior de Ingeniería
Informática y Telecomunicaciones

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS

TRABAJO DE FIN DE GRADO

Análisis de series temporales mediante el uso de bases greedy

Presentado por:
Ángel Olmedo Navarro

Curso académico 2024-2025

Análisis de series temporales mediante el uso de bases greedy

Ángel Olmedo Navarro

Ángel Olmedo Navarro *Análisis de series temporales mediante el uso de bases greedy.*
Trabajo de fin de Grado. Curso académico 2024-2025.

**Responsable de
tutorización**

Javier Meri de la Maza
Análisis Matemático

José Manuel Benítez Sánchez
CCIA

Doble Grado en Ingeniería
Informática y Matemáticas

Facultad de Ciencias y
Escuela Técnica Superior
de Ingeniería Informática y
Telecomunicaciones

Universidad de Granada

DECLARACIÓN DE ORIGINALIDAD

D./Dña. Ángel Olmedo Navarro

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2024-2025, es original, entendido esto en el sentido de que no he utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 14 de julio de 2025

Fdo: Ángel Olmedo Navarro

Agradecimientos

Me gustaría comenzar expresando mi agradecimiento a mis tutores Javier y José Manuel por la confianza que han depositado en mí para desarrollar este trabajo, por su continua disponibilidad y por brindarme todos los recursos que he necesitado para finalizar este proyecto. Su experiencia y su dedicación han sido fundamentales para poder alcanzar todos los objetivos planteados.

También quiero agradecer a Luis, por su constante ayuda y por saber orientarme cuando las cosas no salían como esperaba.

Agradezco de corazón a mis padres, por su apoyo incondicional a lo largo de todos estos años, por confiar en mí en los momentos más difíciles y por enseñarme el valor del esfuerzo y la constancia.

Del mismo modo, quiero agradecer a Elsa por estar siempre a mi lado, por su paciencia infinita y por saber traer la calma cuando más falta hacía.

También me gustaría agradecer a Pedro y Juanma, por su amistad y por todos esos momentos que, sin pretenderlo, tantas veces me han ayudado a ver las cosas con más claridad.

Por último, quiero dar las gracias a mis compañeros de clase por todo lo que hemos recorrido juntos. Gracias por todos esos momentos inolvidables. Ha sido una experiencia que repetiría una y otra vez.

A todos ellos, gracias de corazón.

Summary

This bachelor's thesis presents the study of a new approach to the problem of time-series classification, combining mathematical tools with current machine-learning techniques. The goal has been to design, implement, and test an architecture that can serve as a starting point for future foundation models in this domain, rigorously integrating theoretical concepts with the practical needs inherent to sequential-data processing.

Because of the dual nature of this project, two well-differentiated parts can be distinguished. The first consists of a theoretical deepening in the subject of Functional Analysis. After revisiting the basic concepts of this field, unconditional bases are introduced, showing some of their most relevant properties and most useful characterisations. Subsequently, after a brief presentation of certain measure-theory concepts, the Haar system is described, acting as the nexus between the mathematical and computational sides of this work, and its most important properties are reviewed.

This leads to the concept of a *greedy* basis and its characterisation theorem, whose proof constitutes the main mathematical objective: to show that a basis is *greedy* if, and only if, it is unconditional and democratic. Once this proof is completed, it is used to show that the Haar system is a *greedy* basis and, therefore, can approximate a function with arbitrary precision using only a finite number of the most significant coefficients in its representation.

The mathematical section concludes with the introduction of the wavelet transform, in particular the Haar transform, highlighting its fundamental role in the development of the computational proposal.

The second part of the project focuses on research in the field of artificial intelligence, specifically Transformer models within machine learning. After a study of the theoretical foundations and a review of the state of the art, *HitsBERT* is proposed. A Transformer model based on the well-known BERT architecture, designed for language processing, and adapted in this work to time-series data. This is made possible by *HitsBE*, a module that segments time series and obtains a vector representation of them to embed in a Transformer-type model.

Alongside this model, a specific pre-training scheme is designed. An analogy of the *Masked Language Modeling* method introduced to train BERT is created, but in this case adapted to time series. Although running this pre-training indicates a slight improvement in model accuracy, it does not enhance the output probability distribution. Nevertheless, evaluation proceeds to obtain deeper conclusions.

The experimentation unfolds in two stages. First, the model's performance is measured by comparing it with the *tsfeatures* feature extractor. However, the developed model does not achieve good metrics even relative to that extractor. Therefore, a second stage follows, comparing several pre-trained and non-pre-trained versions of the model, in order to locate the proposal's error. After analysing the results in detail, it is concluded that, in the first instance, the problem lies in the pre-training phase. In general, the non-pre-trained models achieve better separation of some classes, showing clear superiority over the models that have been pre-trained.

Summary

This confirms that the main complication resides in the proposed pre-training. Even so, the approach provided by *HitsBE* for embedding time series proves promising, as it apparently offers a good approximation of them. The work concludes by outlining several future research directions aimed at perfecting the pre-training process and consolidating *HitsBERT* as a foundation model for time-series classification.

Índice general

Agradecimientos	III
Summary	V
Introducción	IX
1. Motivación	IX
2. Objetivos del proyecto	X
3. Estructura del documento	XI
I. Bases greedy en espacios de Banach	1
1. Algunas herramientas del Análisis Funcional	3
1.1. Espacios de Banach	3
1.2. Espacios clásicos del análisis funcional	4
1.2.1. Espacios de sucesiones	4
1.3. Espacios de funciones	5
1.4. Operadores lineales continuos	7
2. Bases Incondicionales	11
2.1. Definición y propiedades básicas	11
2.2. Convergencia incondicional	14
2.3. Bases incondicionales	16
3. El Sistema de Haar	19
3.1. Conceptos de teoría de la medida	19
3.2. Esperanza condicionada	20
3.3. Sistema de Haar	24
3.4. Sucesiones de Rademacher	31
4. Aproximando con Bases Greedy	35
4.1. Bases Greedy	35
4.2. Caracterización de las bases greedy	36
4.3. El caso de la base de Haar	40
5. La transformada de Haar	43
5.1. Transformada Wavelet	44
5.2. Wavelet de Haar	46

II. Clasificación de Series Temporales	51
6. Estado del arte	53
6.1. Clasificación de series temporales	53
6.2. Principales métodos	54
6.3. Aprendizaje automático.	58
6.3.1. Redes neuronales	62
6.3.2. El modelo <i>Transformer</i>	64
7. La propuesta	69
7.1. Vocabulario	69
7.2. Hitsbe	72
7.2.1. Tokenización	73
7.2.2. Indexación de Haar	74
7.2.3. Salida final	75
7.3. HitsBERT	76
7.4. Detalles de implementación	78
8. Evaluación experimental	85
8.1. Preentrenamiento del modelo	85
8.1.1. Enmascaramiento clásico	85
8.1.2. Esquema propuesto	86
8.1.3. Procesamiento de datos	87
8.1.4. Detalles técnicos	88
8.1.5. Resultados	89
8.2. Experimentación	91
9. Conclusiones	105
9.1. Objetivos alcanzados	105
9.2. Tiempo de desarrollo	105
9.3. Trabajo futuro	106
Bibliografía	109

Introducción

1. Motivación

Cada segundo, el mundo genera una cantidad ingente de información. Se estima que este 2025 se alcanzarán los 175 zettabytes de datos, muchos de los cuales provienen de sensores, registros financieros, dispositivos médicos o redes sociales. Una característica común para la mayoría de estos datos es que no llegan de forma aislada, sino organizados en el tiempo: cada observación viene acompañada de una marca temporal, lo que da lugar a las series temporales.

Estas forman parte esencial de numerosos procesos reales, y su presencia es cada vez más habitual en los datos que se generan a diario. Desde el consumo eléctrico hasta los mercados bursátiles, la climatología o los registros biométricos, incorporar la dependencia temporal resulta fundamental para estudiar estas representaciones.

En este contexto, es natural pensar en la clasificación de series temporales, un campo de investigación en pleno auge dentro de la inteligencia artificial. Se trata de un problema complejo, con grandes desafíos y numerosos avances aún por descubrir, lo que lo convierte en la oportunidad perfecta para explorar nuevas ideas y aportar una contribución significativa.

Desde el inicio, este trabajo se planteó como un reto para integrar los conocimientos adquiridos en el doble grado en Ingeniería Informática y Matemáticas. Más allá de cumplir con los requisitos académicos, la intención ha sido utilizar este proyecto como una oportunidad para explorar en profundidad un tema donde ambas disciplinas se complementan de manera natural. Se parte de la convicción de que uno de los principales pilares del doble grado reside precisamente en esa capacidad de aplicar fundamentos matemáticos sólidos al diseño y análisis de herramientas informáticas complejas.

Por ello, la parte matemática se desarrolla en el campo del análisis matemático y funcional, dos ramas que ofrecen un marco riguroso para el estudio de series temporales. En concreto, este trabajo se basa en el estudio de las bases greedy, específicamente en la base de Haar. Herramienta con fuertes implicaciones en la informática por su eficacia a la hora de comprimir información de forma eficiente.

Por otro lado, desde el ámbito informático, se ha querido dar un paso más. Este trabajo representa también una primera aproximación a la investigación en modelos de aprendizaje profundo, proponiendo el diseño de una arquitectura novedosa basada en Transformers adaptados a series temporales. La investigación, el diseño y el desarrollo de este modelo se ha realizado con herramientas actuales, buscando no solo reproducir enfoques existentes, sino aportar una contribución original y abrir la puerta a futuras líneas de trabajo.

En definitiva, esta memoria se plantea como una forma de aplicar y conectar conocimientos adquiridos en ambas disciplinas del doble grado, combinando herramientas matemáticas rigurosas con técnicas actuales de aprendizaje automático. El objetivo ha sido abordar un problema actual explorando tanto su base teórica como su aplicación práctica.

2. Objetivos del proyecto

A partir de la motivación expuesta, los objetivos de este proyecto se dividen en dos vertientes claramente diferenciadas pero profundamente conectadas: una teórica, centrada en el desarrollo matemático, y otra aplicada, enfocada en la investigación informática y la inteligencia artificial.

Objetivo matemático

El objetivo principal desde el punto de vista matemático es profundizar en los fundamentos teóricos del análisis funcional, disciplina que ofrece un marco riguroso para comprender la estructura interna de las series temporales. En particular, se pretende estudiar el concepto de bases *greedy* dentro de los espacios funcionales, alcanzando la comprensión y demostración del teorema que las caracteriza.

El desarrollo matemático culmina con la demostración de que el sistema de Haar constituye una base *greedy*, lo que justifica su capacidad para representar una función mediante un número finito de coeficientes relevantes. Esta propiedad es de especial interés en el ámbito mencionado, ya que permite comprimir información de forma eficiente sin perder las características esenciales del fenómeno que se está representando.

El objetivo es, por tanto, consolidar y poner en práctica los conocimientos adquiridos a lo largo del grado, mostrando madurez matemática a través de la comprensión y exposición rigurosa de estos resultados, que a su vez serán la base para justificar la propuesta informática que se desarrolla en este trabajo.

Objetivo informático

En cuanto al ámbito informático, se pueden encontrar varios objetivos. El principal es buscar una conexión entre las bases de Haar y el problema de clasificación de series temporales, diseñando una arquitectura que integre ambas partes y verificando su correcto funcionamiento. En concreto, esta arquitectura estará basada en un modelo *Transformer*. La vía más directa de aplicación pasa por adaptar los modelos existentes, esto requiere analizar las limitaciones de estas arquitecturas al enfrentarse a datos secuenciales de naturaleza no lingüística y tratar de proponer modificaciones fundamentadas.

Dentro de este objetivo, se busca diseñar un esquema de entrada que permita representar series temporales de manera estructurada, conservando tanto la información local como las características relevantes de su evolución, siendo estas últimas aportadas por el sistema de Haar. Además, dicho diseño deberá ser desarrollado y presentar un correcto funcionamiento.

También se pretende, a partir de esta arquitectura, estudiar su viabilidad a la hora de clasificar series temporales y evaluar empíricamente el rendimiento de la misma con el fin de obtener conclusiones sólidas y detectar posibles líneas de mejora para trabajos futuros. Para ello, se pretende estudiar cómo aprenden estos modelos y diseñar un esquema de preentrenamiento específico que sea adecuado para su aplicación al ámbito del problema.

Finalmente, otro de los objetivos del proyecto es adquirir experiencia en el uso de herramientas modernas de desarrollo que sigan las buenas prácticas actuales en el campo del aprendizaje automático. Se busca familiarizarse con bibliotecas especializadas en *deep learning*, así como con tecnologías enfocadas a la gestión eficiente de proyectos, la organización

del código y la reproducibilidad de los experimentos. Entre estas prácticas se incluye el uso de gestores de dependencias, control de versiones y herramientas para la construcción y despliegue de entornos, con el fin de asegurar que el desarrollo siga los estándares actuales de calidad en ingeniería del software aplicado a la inteligencia artificial.

3. Estructura del documento

Este trabajo se desarrolla en dos bloques: un estudio teórico matemático, centrado en las bases greedy, prestando especial atención a la base de Haar e introduciendo la transformada relacionada a dicha base; y una propuesta aplicada en el ámbito del aprendizaje automático para la clasificación de series temporales mediante una arquitectura innovadora basada en *Transformers*.

El primer bloque, Bases greedy en espacios de Banach, desarrolla un marco teórico sobre el que se sustenta el enfoque propuesto. Se introducen los conceptos fundamentales del análisis funcional, comenzando con los espacios de Banach, la noción de dimensión infinita y el papel de los operadores y funcionales lineales. A continuación, se estudia el concepto de convergencia incondicional y las bases incondicionales, clave para la representación estable de funciones en contextos normados. La tercera sección se centra en la base de Haar, aportando previamente un marco teórico sobre teoría de la medida. La sección final de esta parte examina el uso de bases greedy para aproximación funcional, caracterizandolas y particularizando en el caso concreto de la base de Haar. Todo ello termina con una sección dedicada a la transformada de Haar, que será empleada posteriormente como herramienta computacional en el modelo propuesto.

El segundo bloque, Clasificación de Series Temporales, comienza definiendo el problema a resolver seguido de un estudio del estado del arte donde se recogen las principales técnicas de clasificación utilizadas hasta la fecha. Posteriormente, se realiza una introducción al aprendizaje automático, centrándose en el modelo *Transformer* y su estructura interna.

Sobre esta base, el siguiente capítulo recoge la propuesta original del trabajo. Se introduce un sistema de representación propio, denominado Hitsbe, que combina técnicas de tokenización funcional con una indexación basada en la transformada de Haar. Esta representación es empleada como entrada de un modelo cuya arquitectura se basa en los *Transformers*, denominada HitsBERT, diseñada específicamente para series temporales.

La sección de evaluación experimental recoge los procedimientos seguidos para preentrenar el modelo y validar su rendimiento, incluyendo los conjuntos de datos utilizados, el esquema de enmascaramiento y la metodología experimental. Se plantean varios experimentos que evalúan el modelo propuesto frente a métricas estándar de clasificación. Pese a no obtener los resultados esperados se realiza un análisis profundo para comprender los puntos fuertes, los puntos débiles del modelo y el potencial de este.

Finalmente, este trabajo concluye con una recapitulación de los objetivos alcanzados y una reflexión sobre posibles líneas futuras de investigación y mejora.

Parte I.

Bases greedy en espacios de Banach

1. Algunas herramientas del Análisis Funcional

Este capítulo consiste en un breve recordatorio de las herramientas básicas del análisis funcional necesarias para la comprensión del presente trabajo. El objetivo es repasar conceptos fundamentales ya asimilados durante el grado, poniendo énfasis en sus propiedades más relevantes.

Todos estos contenidos han sido adquiridos y consolidados a partir de los apuntes de *Análisis Matemático II* [Albb] y *Análisis Funcional* [Alba], estudiados en el transcurso del doble grado en Matemáticas e Ingeniería Informática en la *Universidad de Granada*.

1.1. Espacios de Banach

Los espacios vectoriales considerados estarán definidos sobre el cuerpo \mathbb{R} , salvo indicación expresa del uso de \mathbb{C} . Si X es un espacio vectorial. Una *norma* en X es una aplicación $\|\cdot\| : X \rightarrow \mathbb{R}_0^+$ que satisface:

- I. *Desigualdad triangular*: $\|x + y\| \leq \|x\| + \|y\|$, $\forall x, y \in X$.
- II. *Homogeneidad*: $\|\lambda x\| = |\lambda| \cdot \|x\|$, $\forall x \in X, \forall \lambda \in \mathbb{K}$.
- III. *No degeneración*: $\|x\| = 0 \Rightarrow x = 0$, $\forall x \in X$.

El par $(X, \|\cdot\|)$ se denomina *espacio normado*.

Fijada una dimensión $N \in \mathbb{N}$, se considera el espacio vectorial \mathbb{R}^N . Un elemento $x \in \mathbb{R}^N$ se representará como $x = (x(1), x(2), \dots, x(N))$, mientras que una sucesión en dicho espacio vendrá dada por $\{x_n\} \subset \mathbb{R}^N$. Sobre este espacio se pueden definir distintas normas. Para cada $1 \leq p < \infty$ se define la función

$$\|x\|_p = \left(\sum_{i=1}^N |x(i)|^p \right)^{\frac{1}{p}}, \quad x \in \mathbb{R}^N.$$

Esta cumple las tres propiedades anteriores y, por tanto, define una norma. Para el caso $p = \infty$ se define

$$\|x\|_\infty = \max\{|x(i)| : 1 \leq i \leq N\},$$

que también cumple las condiciones para ser norma. En consecuencia, el espacio $(\mathbb{R}^N, \|\cdot\|_p)$ es un espacio normado para todo $1 \leq p \leq \infty$.

A continuación se introduce el concepto de sucesión de Cauchy, necesario para definir la completitud de un espacio normado.

Definición 1.1. Una sucesión $\{x_n\} \subset X$ se dice *sucesión de Cauchy* si, para todo $\varepsilon > 0$, existe $N \in \mathbb{N}$ tal que

$$\|x_n - x_m\| < \varepsilon \quad \text{para todo } m, n > N.$$

1. Algunas herramientas del Análisis Funcional

Es decir, los términos de la sucesión se aproximan entre sí tanto como se desee a partir de cierto índice.

Con esta definición se puede considerar un nuevo tipo de espacios. Sea $(X, \|\cdot\|)$ un espacio normado, este se dice que es un *espacio de Banach* si toda sucesión de Cauchy en X converge a un elemento de X .

Otra propiedad importante para el estudio de los espacios normados es la *separabilidad*. Un espacio normado X se dice separable si contiene un subconjunto denso y numerable. La separabilidad constituye un *invariante topológico*, es decir, una propiedad que se preserva bajo homeomorfismos. En particular, permite distinguir entre espacios no isomorfos topológicamente.

1.2. Espacios clásicos del análisis funcional

A partir del concepto de espacio de Banach surgen numerosos ejemplos fundamentales que sirven como modelo y punto de partida en el desarrollo del análisis funcional. Podemos clasificar los espacios clásicos en dos grandes grupos: los espacios de sucesiones y los espacios de funciones.

El estudio de estos espacios se apoya de forma recurrente en desigualdades fundamentales como la de Hölder y Minkowski, que garantizan propiedades de normabilidad y completitud que serán utilizadas a lo largo del texto.

Desigualdad de Hölder

Sean $1 < p < \infty$, p^* su exponente conjugado (es decir, $1/p + 1/p^* = 1$), y $a_1, \dots, a_N, b_1, \dots, b_N \in \mathbb{R}_0^+$, entonces:

$$\sum_{k=1}^N a_k b_k \leq \left(\sum_{k=1}^N a_k^p \right)^{1/p} \left(\sum_{k=1}^N b_k^{p^*} \right)^{1/p^*}.$$

Desigualdad de Minkowski

Sean $1 \leq p < \infty$, $a_1, \dots, a_N, b_1, \dots, b_N \in \mathbb{R}_0^+$, entonces:

$$\left(\sum_{k=1}^N (a_k + b_k)^p \right)^{1/p} \leq \left(\sum_{k=1}^N a_k^p \right)^{1/p} + \left(\sum_{k=1}^N b_k^p \right)^{1/p}.$$

1.2.1. Espacios de sucesiones

Los espacios de sucesiones clásicos son los espacios ℓ_p , definidos para $1 \leq p < \infty$. Estos agrupan todas aquellas sucesiones $x = \{x(n)\}$ tales que la serie $\sum_{n=1}^{\infty} |x(n)|^p$ es convergente. Este conjunto se convierte en un espacio vectorial normado dotándolo con la norma

$$\|x\|_p = \left(\sum_{n=1}^{\infty} |x(n)|^p \right)^{1/p}.$$

Además, puede demostrarse que dicho espacio es completo respecto a dicha norma, por lo que se trata de un espacio de Banach.

El caso $p = \infty$ da lugar al espacio ℓ_∞ , que contiene todas las sucesiones acotadas de escalares. En este caso, la norma considerada es la del supremo:

$$\|x\|_\infty = \sup\{|x(n)| : n \in \mathbb{N}\},$$

y también se cumple que este espacio es de Banach respecto a la misma.

Dentro de ℓ_∞ aparecen dos subespacios particularmente relevantes. Por un lado, el espacio c_0 , formado por las sucesiones que convergen a cero. Este espacio tiene la propiedad de ser un subespacio cerrado de ℓ_∞ y, por tanto, también es un espacio de Banach con la norma del supremo.

Por otro lado, un subespacio aún más restringido es c_{00} , que contiene aquellas sucesiones con soporte finito, es decir, con un número finito de términos no nulos. Este espacio es denso en c_0 , lo que permite aproximar cualquier elemento de c_0 por sucesiones de soporte finito, sin embargo, c_{00} no es cerrado en ℓ_∞ .

Relación entre los espacios ℓ_p

Es posible establecer relaciones de inclusión entre los espacios ℓ_p para distintos valores de p . Si se cumple $1 \leq p < q < \infty$, entonces se verifica la inclusión

$$x \in \ell_p \Rightarrow x \in \ell_q \text{ y } \|x\|_q \leq \|x\|_p.$$

En consecuencia, cada espacio ℓ_p está contenido en ℓ_q cuando $p < q$, y dicha inclusión es estricta. La norma $\|\cdot\|_p$, sin embargo, no coincide con la norma inducida por ℓ_q , ya que la topología inducida por $\|\cdot\|_p$ es más fina que la inducida por $\|\cdot\|_q$.

Esta desigualdad permite afirmar que, aunque los elementos de ℓ_p también pertenecen a ℓ_q , las normas y topologías correspondientes no son equivalentes. Por tanto, los espacios ℓ_p para distintos valores de p son esencialmente distintos, incluso si uno está contenido en el otro como conjunto.

Además, cabe destacar que, para $1 \leq p < \infty$, toda sucesión de ℓ_p converge a cero, lo que implica que $\ell_p \subset c_0$, siendo esta inclusión estricta.

1.3. Espacios de funciones

En el caso de los espacios de funciones, para $1 \leq p \leq \infty$ se encuentran los espacios de Lebesgue L_p . Están definidos sobre un intervalo, normalmente $[0, 1]$, y dotado de la medida de Lebesgue. Estos espacios permiten trabajar con funciones medibles que pueden no ser continuas, siempre que su integral en valor absoluto elevado a p sea finita. Formalmente, para $1 \leq p < \infty$, se define

$$L_p = \left\{ f \text{ medible} : \int_0^1 |f(t)|^p dt < \infty \right\},$$

y se dota de la norma

1. Algunas herramientas del Análisis Funcional

$$\|f\|_p = \left(\int_0^1 |f(t)|^p dt \right)^{1/p}.$$

Como en el caso de las sucesiones, la validez de esta norma se justifica mediante desigualdades integrales, destacando las de Hölder y Minkowski, que generalizan las correspondientes para series. Además, se cumple que estos espacios, $(L_p, \|\cdot\|_p)$, son espacios de Banach al igual que ocurría con ℓ_p .

El espacio de funciones esencialmente acotadas L_∞ se obtiene como el caso límite de los espacios de Lebesgue cuando $p = \infty$. En este caso, se considera acotada una función si existe una constante M tal que $|f(t)| \leq M$ para casi todo $t \in [0, 1]$, es decir, salvo en un conjunto de medida nula. La menor constante con esta propiedad se denomina *supremo esencial*, y se define como

$$\text{ess sup } |f| = \inf \{M \in \mathbb{R}_0^+ \cup \{\infty\} : |f(t)| \leq M \text{ p.c.t. en } [0, 1]\}.$$

El conjunto de funciones medibles cuyo supremo esencial es finito forma el espacio

$$L_\infty = \{f \in L : \text{ess sup } |f| < \infty\},$$

y se dota de la norma

$$\|f\|_\infty = \text{ess sup } |f|.$$

Además, al igual que en el caso $p < \infty$, se puede demostrar que $(L_\infty, \|\cdot\|_\infty)$ es un espacio de Banach.

Relaciones entre los espacios de Lebesgue

Los espacios de Lebesgue L_p forman una familia de espacios de Banach dependientes del parámetro p , y cuya relación de inclusión se comporta de manera inversa a la observada en los espacios de sucesiones ℓ_p . En efecto, si $1 \leq p < q \leq \infty$, se verifica que $L_q \subset L_p$, con inclusión estricta. Además, para $f \in L_q$ se tiene que $\|f\|_p \leq \|f\|_q$, luego la norma decrece con el parámetro p . Esta inclusión es estricta, por lo que siempre puede encontrarse un elemento $f \in L_p$ de forma que $f \notin L_q$.

Otro aspecto esencial en la comprensión de los espacios L_p es su relación con el espacio de funciones continuas $C[0, 1]$. Dado que las funciones continuas están acotadas en $[0, 1]$, se puede mostrar que $C[0, 1]$ es isométricamente isomorfo a un subespacio cerrado de L_∞ .

Por otro lado, aunque la norma $\|\cdot\|_p$ para $1 \leq p < \infty$ difiere de la norma del máximo usada en $C[0, 1]$, se puede demostrar que $C[0, 1]$ es denso en L_p . Esto implica que cualquier función en L_p puede aproximarse arbitrariamente, en norma $\|\cdot\|_p$, por funciones continuas.

Para finalizar esta sección cabe resaltar la relación que hay entre los espacios de sucesiones clásicos y los espacios de Lebesgue. Para todo $1 \leq p \leq \infty$, el espacio de sucesiones ℓ_p es isométricamente isomorfo a un subespacio del espacio L_p . En concreto, los espacios ℓ_p pueden interpretarse como subespacios lineales y cerrados de los espacios L_p , mediante una correspondencia basada en funciones por tramos.

En conjunto, los espacios de Lebesgue proporcionan un entorno flexible, robusto y completo para el análisis de espacios de funciones, siendo fundamentales en la formulación moderna

del análisis funcional y en numerosas aplicaciones tanto teóricas como prácticas.

1.4. Operadores lineales continuos

El estudio de operadores lineales continuos entre espacios normados constituye una de las herramientas centrales del análisis funcional, ya que permite analizar la acción de transformaciones lineales en contextos de dimensión infinita. Sean X y Y espacios vectoriales normados. Una aplicación $T : X \rightarrow Y$ se denomina *operador lineal* si satisface la propiedad de linealidad:

$$T(\lambda u + v) = \lambda T(u) + T(v), \quad \forall u, v \in X, \forall \lambda \in \mathbb{R}.$$

La continuidad de estos operadores admite una caracterización sencilla y de gran utilidad: un operador lineal T es continuo si, y solo si, es acotado, es decir, si existe $M \in \mathbb{R}_0^+$ tal que

$$\|T(x)\| \leq M\|x\|, \quad \forall x \in X.$$

Este resultado conecta directamente la estructura algebraica con la estructura topológica del espacio. Una de las propiedades más importantes de estos operadores es que basta con que T sea continuo en un único punto, el origen por ejemplo, para que lo sea en todo X .

Además, el hecho de que T sea acotado implica que se trata de una aplicación lipschitziana, y por tanto, continua de manera uniforme. En este contexto, todas las nociones usuales de continuidad (puntual, uniforme, lipschitziana) coinciden. Este fenómeno contrasta con el caso general de funciones no lineales, en el que dichas nociones pueden diferir notablemente.

Otra caracterización útil de la continuidad se da en términos de la preservación de conjuntos acotados. Un operador lineal $T : X \rightarrow Y$ es continuo si, y solo si, transforma conjuntos acotados de X en conjuntos acotados de Y . En otras palabras, la continuidad de T equivale a que este lleva acotados en acotados.

En resumen, la continuidad de los operadores lineales entre espacios normados se caracteriza de forma simple y robusta mediante la noción de acotación. Esta equivalencia entre continuidad y acotación será de gran utilidad en el desarrollo posterior de este trabajo.

Espacios de operadores

Una vez introducidos los operadores lineales continuos entre espacios normados, es natural considerar el conjunto de todos ellos como un espacio vectorial en sí mismo. En este sentido, dado un espacio normado X y otro Y , se denota por $L(X, Y)$ al conjunto de operadores lineales continuos de X en Y . Este conjunto admite una estructura de espacio vectorial y, además, puede dotarse de una norma.

Dado $T \in L(X, Y)$, su norma se define como el menor valor $M \in \mathbb{R}_0^+$ tal que

$$\|T(x)\| \leq M\|x\|, \quad \forall x \in X,$$

es decir,

$$\|T\| = \inf \{M \in \mathbb{R}_0^+ : \|T(x)\| \leq M\|x\| \forall x \in X\}.$$

1. Algunas herramientas del Análisis Funcional

Esta definición admite múltiples expresiones equivalentes, que resultan útiles en distintos contextos. Una de las formas más directas consiste en considerar el cociente entre la norma de la imagen y la norma del vector, lo que lleva a la expresión:

$$\|T\| = \sup \left\{ \frac{\|T(x)\|}{\|x\|} : x \in X \setminus \{0\} \right\}.$$

Dado que esta expresión involucra todos los vectores no nulos de X , es habitual restringirse a subconjuntos acotados que generen el mismo supremo. En particular, puede evaluarse la acción del operador sobre la esfera unidad $S = \{z \in X : \|z\| = 1\}$, de modo que:

$$\|T\| = \sup \{\|T(z)\| : z \in S\}.$$

Del mismo modo, se puede trabajar con la bola cerrada unidad $B = \{x \in X : \|x\| \leq 1\}$, obteniendo así,

$$\|T\| = \sup \{\|T(x)\| : x \in B\}.$$

Todas estas formas son equivalentes y su elección depende del contexto en el que se utilicen. Gracias a esta norma, el conjunto $L(X, Y)$ se convierte en un espacio normado. Más aún, si Y es un espacio de Banach, entonces $L(X, Y)$ también lo es. Por tanto, el espacio de operadores hereda la completitud de su espacio de llegada.

La convergencia en $L(X, Y)$ se interpreta como convergencia en norma de operador, lo que implica que una sucesión $\{T_n\} \subset L(X, Y)$ converge a T si

$$\lim_{n \rightarrow \infty} \|T_n - T\| = 0.$$

Funcionales lineales continuos

El caso particular en que el espacio de llegada es \mathbb{R} conduce al estudio de los *funcionales lineales continuos*, es decir, aplicaciones lineales $f : X \rightarrow \mathbb{R}$ definidas sobre un espacio normado X y que verifican la condición de continuidad. Como se ha mencionado, toda aplicación lineal continua es acotada, y viceversa, por lo que existe $M \in \mathbb{R}_0^+$ tal que

$$|f(x)| \leq M\|x\| \quad \text{para todo } x \in X.$$

El conjunto de todos estos funcionales se denota por X^* y recibe el nombre de *espacio dual* de X . Este conjunto, provisto de la *norma dual*

$$\|f\| = \sup \{|f(x)| : \|x\| \leq 1\},$$

se convierte en un espacio normado que, además, es siempre de Banach, incluso si X no lo es.

Dual de los espacios ℓ_p

Para $1 < p < \infty$, el dual del espacio ℓ_p se identifica isométricamente con ℓ_{p^*} , donde p^* es el exponente conjugado, es decir, el valor que satisface $1/p + 1/p^* = 1$. A cada sucesión $y \in \ell_{p^*}$ se le asocia un funcional $\hat{y} \in \ell_p^*$ definido por

$$\widehat{y}(x) = \sum_{n=1}^{\infty} x(n)y(n), \quad \forall x \in \ell_p.$$

Este funcional es lineal y continuo, y se verifica que $\|\widehat{y}\| = \|y\|_{p^*}$. Además, todo funcional lineal y continuo sobre ℓ_p se puede expresar de esta forma. En consecuencia, la aplicación

$$\Phi : \ell_{p^*} \rightarrow \ell_p^*, \quad y \mapsto \widehat{y}$$

define un isomorfismo lineal isométrico entre ℓ_{p^*} y el espacio dual ℓ_p^* . En particular, cuando $p = 2$, se tiene que ℓ_2 es autodual. Es decir, ℓ_2^* es isométricamente isomorfo a ℓ_2 .

El caso $p = 1$ presenta una situación particular. Dada una sucesión $y \in \ell_\infty$, se puede definir el funcional

$$\widehat{y}(x) = \sum_{n=1}^{\infty} x(n)y(n), \quad \forall x \in \ell_1,$$

lo cual determina una aplicación lineal continua sobre ℓ_1 . En consecuencia, se tiene que el espacio dual ℓ_1^* es isométricamente isomorfo a ℓ_∞ . Sin embargo, este resultado no es recíproco: ℓ_1^* no es el dual de ℓ_∞^* . De hecho, ℓ_1 es separable, mientras que ℓ_∞ no lo es.

Sin embargo, ℓ_1 sí corresponde con el dual de c_0 , dada una sucesión $y \in \ell_1$, se define

$$\widehat{y}(x) = \sum_{n=1}^{\infty} x(n)y(n), \quad \forall x \in c_0,$$

que determina un funcional continuo. Toda forma lineal continua sobre c_0 se obtiene así, y se verifica que $\|\widehat{y}\| = \|y\|_{\ell_1}$. Por lo que el espacio dual c_0^* se identifica isométricamente con ℓ_1 .

Duales de los espacios de Lebesgue

El estudio del dual de los espacios L_p , con $1 \leq p < \infty$, sigue un paralelismo con el caso de los espacios ℓ_p . La desigualdad de Hölder, si $f \in L_p$ y $g \in L_{p^*}$, garantiza que $fg \in L_1$ y que:

$$\left| \int_0^1 f(t)g(t) dt \right| \leq \|f\|_p \|g\|_{p^*},$$

lo cual permite definir, para cada $g \in L_{p^*}$, un funcional lineal continuo sobre L_p mediante la fórmula:

$$\widehat{g}(f) = \int_0^1 f(t)g(t) dt, \quad \forall f \in L_p.$$

Dicho funcional cumple que $\|\widehat{g}\| \leq \|g\|_{p^*}$, y puede demostrarse que en realidad se cumple la igualdad. Así, la aplicación

$$\Phi : L_{p^*} \rightarrow (L_p)^*, \quad g \mapsto \widehat{g}$$

define un isomorfismo lineal isométrico. Luego todo funcional lineal continuo sobre L_p puede expresarse mediante una función en el espacio conjugado L_{p^*} , y esta correspondencia es isométrica.

1. Algunas herramientas del Análisis Funcional

En cambio, el caso $p = \infty$ presenta un comportamiento diferente pues la aplicación Φ no es sobreyectiva. De hecho, al igual que ocurre en el caso de los espacios ℓ_p , el espacio L_1 es separable, mientras que su dual $(L_\infty)^*$ no lo es. Por tanto, $(L_\infty)^*$ no puede identificarse con L_1 .

El principio de acotación uniforme

En el estudio de familias de funcionales o de operadores lineales acotados, es común encontrarse con situaciones donde se tiene información sobre su comportamiento puntual, pero se desea obtener conclusiones uniformes. En estos casos, uno de los resultados fundamentales que será de gran ayuda es el *Principio de acotación uniforme*, también conocido como teorema de Banach-Steinhaus.

Teorema 1.2. *Principio de acotación uniforme.* Sea X un espacio de Banach y $\{T_n\} \subset \mathcal{L}(X, Y)$ una sucesión de operadores lineales continuos. Si para cada $x \in X$ la sucesión $\{T_n(x)\}$ está acotada en Y , entonces se cumple:

$$\sup\{\|T_n\| : n \in \mathbb{N}\} < \infty.$$

Es decir, la familia $\{T_n\}$ está uniformemente acotada en norma.

Una consecuencia de este teorema que también será de gran utilidad en este trabajo es la siguiente.

Proposición 1.3. *Sea $\{T_n\}$ una sucesión de operadores lineales continuos de un espacio de Banach X en un espacio lineal normado Y tal que $\sup\{\|T_n\| : n \in \mathbb{N}\} < \infty$. Si $T : X \rightarrow Y$ es otro operador, entonces el subespacio*

$$\{x \in X : \|T_n(x) - T(x)\| \rightarrow 0\}$$

es cerrado en norma en X .

Se finaliza así este capítulo introductorio donde la intención era refrescar los conceptos básicos más relevantes sobre los que parte este trabajo.

2. Bases Incondicionales

En este capítulo se introducen las bases de Schauder como una herramienta esencial del análisis funcional, especialmente útil para trabajar con espacios de Banach de dimensión infinita. Se detallan sus definiciones fundamentales y propiedades clave, como los funcionales biortogonales, la constante de la base y los operadores de suma parcial.

A continuación, se profundiza en el concepto de convergencia incondicional, una generalización significativa de la convergencia tradicional, que no depende del orden específico en el que se suman los términos. Esta convergencia dará lugar a un nuevo tipo de bases, las bases incondicionales, que desempeñarán un papel fundamental en este trabajo.

2.1. Definición y propiedades básicas

Definición 2.1. Una sucesión de elementos $\{e_n\}$ de un espacio de Banach X de dimensión infinita, se dice que es una *base de Schauder de X* si, para cada elemento $x \in X$, existe una única sucesión de escalares $\{a_n\}$ tal que

$$x = \sum_{k=1}^{\infty} a_k e_k.$$

Es decir, la sucesión $\{\sum_{k=1}^n a_k e_k\}$ converge a x en la norma de X . Supóngase ahora que existe una sucesión $\{e_n^*\} \subset X^*$ de forma que:

I. $e_k^*(e_j) = 1$ si $j = k$ y $e_k^*(e_j) = 0$ en otro caso, con $k, j \in \mathbb{N}$.

II. $x = \sum_{k=1}^{\infty} e_k^*(x) e_k$ para todo $x \in X$.

En tal caso $\{e_n^*\}$ son los *funcionales biortogonales* asociados a la base de Schauder $\{e_n\}$.

En espacios de dimensión infinita, el concepto de *base* suele usarse para referirse a una base de Schauder. El uso de base algebraica, basada en combinaciones lineales finitas, suele estar bastante limitado en análisis funcional. Este trabajo se centrará principalmente en bases de Schauder, dejando clara la distinción en caso de ser necesario.

Un caso particularmente importante dentro de los espacios de Banach es el de los espacios con estructura de producto interno. En estos, no solo se puede hablar de convergencia y norma, sino también de ángulo y ortogonalidad entre vectores, lo cual enriquece significativamente el análisis.

Este tipo de espacios, que generalizan el espacio euclídeo a dimensión infinita, reciben el nombre de *espacios de Hilbert*. Su relevancia es fundamental en teoría de bases, ya que permiten definir y trabajar con bases ortonormales, en las que la convergencia de las series asociadas es especialmente sencilla de analizar.

2. Bases Incondicionales

Definición 2.2. Un *espacio de Hilbert* es un espacio vectorial real dotado de un producto interno $\langle \cdot, \cdot \rangle$ que es completo respecto a la norma inducida por dicho producto:

$$\|x\| = \sqrt{\langle x, x \rangle}.$$

En este contexto, se pueden definir bases ortonormales, que generalizan las bases canónicas de espacios como \mathbb{R}^N y presentan propiedades de gran utilidad.

Se definen los operadores suma parcial, $S_n : X \rightarrow X$, de la siguiente manera. Si $n = 0$, se toma $S_0 = 0$ y, para $n \geq 1$,

$$S_n(x) = S_n \left(\sum_{k=1}^{\infty} e_k^*(x) e_k \right) = \sum_{k=1}^n e_k^*(x) e_k.$$

La aplicación S_n es un operador lineal continuo, luego acotado, puesto que cada funcional e_k^* es continuo. Obsérvese además que S_k está asociado a la base $\{e_n\}$ para todo $k \in \mathbb{N}$.

Proposición 2.3. Sea $\{e_n\}$ una base de Schauder de un espacio de Banach X y $\{S_n\}$ la sucesión de sumas parciales asociada a la base. Entonces,

$$\sup\{\|S_n\| : n \in \mathbb{N}\} < \infty.$$

Demostración. La sucesión de operadores lineales y continuos es puntualmente acotada en X porque $\{S_n(x)\}$ converge para cada x en X . Luego por el *Principio de acotación uniforme*, se concluye que $\sup\{\|S_n\| : n \in \mathbb{N}\} < \infty$. ■

Definición 2.4. Si $\{e_n\}$ es una base de Schauder de un espacio de Banach X , entonces $K = \sup\{\|S_n\|\}$ es llamada la *constante de la base*. En el caso $K = 1$, la base $\{e_n\}$ es *monótona*.

Se presenta a continuación un método inductivo para construir una base de Schauder en un espacio de Banach X .

Proposición 2.5. Sea $S_n : X \rightarrow X$, con $n \in \mathbb{N}$, una sucesión de proyecciones lineales acotadas en un espacio de Banach X que cumplen las siguientes propiedades:

- I. $\dim S_n(X) = n$ para todo $n \in \mathbb{N}$.
- II. $S_n S_m = S_m S_n = S_{\min\{n,m\}}$ para cualquier m, n .
- III. $\{S_n(x)\} \rightarrow x$ para todo $x \in X$.

Entonces, cualquier sucesión de vectores $\{e_n\}$ en X elegida de forma que $e_n \in S_n(X) \cap S_{n-1}^{-1}(0)$ y $\|e_n\| = 1$, es una base de Schauder de X con proyecciones dadas por la sucesión $\{S_n\}$.

Demostración. Sea $e_1 \in S_1(X)$ el primer vector no nulo tal que $\|e_1\| = 1$ y se define el funcional $e_1^* : X \rightarrow \mathbb{R}$ de forma que $e_1^*(x)e_1 = S_1(x)$. Para el siguiente vector, se toma $e_2 \in S_2(X) \cap S_1^{-1}(0)$ no nulo, es decir, que esté en la imagen de S_2 y sea anulado por S_1 , y tal que $\|e_2\| = 1$. Se define entonces $e_2^*(x) = S_2(x) - S_1(x)$.

Generalizando, pueden construirse todos los funcionales por inducción. Para cada $n \in \mathbb{N}$, se toma $e_n \in S_n(X) \cap S_{n-1}^{-1}(0)$, no nulo y normalizado, y se define $e_n^* : X \rightarrow \mathbb{R}$ de modo que

$$e_n^*(x)e_n = S_n(x) - S_{n-1}(x).$$

Obsérvese que el funcional es lineal y está acotado,

$$\begin{aligned} |e_n^*(x)| &= \frac{\|S_n(x) - S_{n-1}(x)\|}{\|e_n\|} = \|S_n(x) - S_{n-1}(x)\| \\ &\leq \|S_n\| \|x\| + \|S_{n-1}\| \|x\| \leq 2 \sup \{\|S_n\| : n \in \mathbb{N}\} \|x\|, \end{aligned}$$

luego $e_n^* \in X^*$. Además, para cada $k, j \in \mathbb{N}$, si $k = j$, se cumple que

$$|e_k^*(e_j)| = \|S_k(e_j) - S_{k-1}(e_j)\| = \|e_j\| = 1,$$

mientras que para $k \neq j$,

$$|e_k^*(e_j)| = \|S_k(e_j) - S_{k-1}(e_j)\| = \|0\| = 0.$$

Finalmente, si se toma $S_0(x) = 0$ para todo $x \in X$, entonces

$$S_n(x) = \sum_{k=1}^n (S_k(x) - S_{k-1}(x)) = \sum_{k=1}^n e_k^*(x)e_k,$$

y como $\{S_n(x)\} \rightarrow x$, se tiene finalmente que

$$x = \sum_{k=1}^{\infty} e_k^*(x)e_k \quad \text{para todo } x \in X.$$

■

Definición 2.6. Una sucesión $\{e_n\}$ en un espacio de Banach X es llamada *sucesión básica* si es una base para $[\{e_n\}] = \overline{\text{Lin}\{e_n\}}$. Siendo $\text{Lin}\{e_n\}$ el subespacio engendrado por $\{e_n\}$, es decir, el mínimo subespacio de X que contiene a $\{e_n\}$.

Se presenta a continuación una caracterización de las sucesiones básicas, también conocida como *Criterio de Grumblum*.

Proposición 2.7 (Criterio de Grumblum). *Una sucesión $\{e_n\}$ de elementos distintos de cero en un espacio de Banach X es básica si, y solo si, existe una constante positiva $K > 0$ tal que,*

$$\left\| \sum_{k=1}^M a_k e_k \right\| \leq K \left\| \sum_{k=1}^N a_k e_k \right\| \tag{2.1}$$

para cualquier sucesión de escalares $\{a_n\}$ y cualesquiera naturales $M \leq N$.

Demostración. Sea $\{e_n\}$ una sucesión básica, y sean $S_m : [\{e_n\}] \rightarrow [\{e_n\}]$ las proyecciones de sumas parciales. Entonces, si $M, N \in \mathbb{N}$ son dos naturales tales que $M \leq N$, se cumple

$$\left\| \sum_{k=1}^M a_k e_k \right\| = \left\| S_M \left(\sum_{k=1}^N a_k e_k \right) \right\| \leq \sup \{\|S_m\| : m \in \mathbb{N}\} \left\| \sum_{k=1}^N a_k e_k \right\|.$$

Por tanto, la desigualdad (2.1) se cumple con $K = \sup \{\|S_m\| : m \in \mathbb{N}\}$.

2. Bases Incondicionales

Para la otra implicación, sea $E = \text{Lin}\{e_n\}$ el subespacio generado por $\{e_n\}$. Se define el operador de rango finito $s_m : E \rightarrow [\{e_1, \dots, e_m\}]$ por

$$s_m \left(\sum_{k=1}^N a_k e_j \right) = \sum_{k=1}^{\min\{m, N\}} a_k e_k, \quad m, N \in \mathbb{N}.$$

Como E es denso en $[\{e_n\}]$ y s_m es un operador lineal y acotado $\|s_m\| \leq K$, s_m se extiende a $S_m : [\{e_n\}] \rightarrow [\{e_1, \dots, e_m\}]$ con $\|S_m\| = \|s_m\| \leq K$. Además, para todo $x \in E$ se tiene que

$$S_n S_m(x) = S_m S_n(x) = S_{\min\{m, n\}}(x),$$

y esta ecuación se cumple para todo $x \in [\{e_n\}]$. Finalmente, por la [Proposición 1.3](#), el conjunto $\{x \in [\{e_n\}] : S_m(x) \rightarrow x\}$ es cerrado y contiene a E , que es denso en $[\{e_n\}]$, por lo tanto $\{S_n(x)\}$ converge a x para todo $x \in [\{e_n\}]$. Por la [Proposición 2.5](#), se concluye que $\{e_k\}$ es una base de $[\{e_n\}]$ con proyecciones de suma parcial $\{S_m\}$. ■

2.2. Convergencia incondicional

Definición 2.8. Sea $\{x_n\}$ una sucesión en un espacio de Banach X . Se dice que la serie $\sum_{n=1}^{\infty} x_n$ converge *incondicionalmente* en X si, para cualquier permutación σ de \mathbb{N} , la serie $\sum_{n=1}^{\infty} x_{\sigma(n)}$ converge en X .

A continuación se presenta una serie de caracterizaciones de series incondicionalmente convergentes cuya demostración puede encontrarse en [\[Heio4\]](#). Estas caracterizaciones serán de utilidad como herramienta para futuras demostraciones además de aportar intuición al concepto de base de incondicional.

Lema 2.9. Sea $\sum_{n=1}^{\infty} x_n$ una serie en un espacio de Banach X . Entonces son equivalentes:

- I. $\sum_{n=1}^{\infty} x_n$ es incondicionalmente convergente.
- II. $\sum_{n=1}^{\infty} x_{n_k}$ converge para cualquier sucesión parcial creciente $\{n_k\}$.
- III. $\sum_{n=1}^{\infty} \epsilon_n x_n$ converge para cualquier sucesión de signos $\{\epsilon_n\}$.
- IV. Para cada $\varepsilon > 0$ existe un número natural $K \in \mathbb{N}$ tal que para cualquier subconjunto finito F de $\{K+1, K+2, \dots\}$ se cumple

$$\left\| \sum_{k \in F} x_k \right\| < \varepsilon.$$

Demostración. I \Rightarrow IV. Supóngase que IV es falso, entonces existe un $\varepsilon > 0$ con el que para cualquier entero $K \in \mathbb{N}$ se puede encontrar un conjunto finito $F_n \subset \{K+1, K+2, \dots\}$ tal que,

$$\left\| \sum_{k \in F_K} x_k \right\| \geq \varepsilon.$$

El objetivo es construir una permutación σ que haga a la serie $\sum_{n=1}^{\infty} x_{\sigma(n)}$ divergente. Para $n_1 = 1$ se toman $A_1 = F_{n_1}$, $n_2 = \max A_1$ y $B_1 = \{n_1 + 1, \dots, n_2\} \setminus A_1$. De forma iterativa,

se eligen $A_2 = F_{n_2}$, $n_3 = \max A_2$ y $B_2 = \{n_2 + 1, \dots, n_3\} \setminus A_2$. Obteniendo así una sucesión $\{n_k\}$ y una partición de $\{n_k + 1, \dots, n_{k+1}\} = A_k \cup B_k$ para todo $k \in \mathbb{N}$.

Se define entonces la permutación que reordena cada uno de tales conjuntos de forma que los elementos de A_k precedan a todos los elementos de B_k . Se tiene de esta forma que la serie $\sum_{n=1}^{\infty} x_{\sigma(n)}$ diverge, pues no es de Cauchy. Para cualquier $\varepsilon > 0$ y cualquier $N \in \mathbb{N}$ que se elija, existirá algún $k \in \mathbb{N}$ de forma que $n_k + 1 > N$ y,

$$\left\| \sum_{j \in A_k} x_j \right\| \geq \varepsilon.$$

$IV \Rightarrow II$. Se considera $\sigma : \mathbb{N} \rightarrow \mathbb{N}$ una aplicación creciente y supóngase que IV es cierto. Sea $N \in \mathbb{N}$ un entero para el que $\sigma(n) > K + 1$ para todo $n > N$, siendo K el valor definido en la hipótesis IV . Sean $L > M \geq N$, entonces

$$\min\{\sigma(M + 1), \dots, \sigma(L)\} > N$$

y, por hipótesis, $\left\| \sum_{k=K+1}^L x_{\sigma(k)} \right\| < \varepsilon$, luego la serie es de Cauchy y converge en X .

$II \Rightarrow III$. Sea $\{\epsilon_n\}$ una sucesión de signos, se define entonces la partición

$$F^+ = \{n \in \mathbb{N} : \epsilon_n = 1\}, \quad F^- = \{n \in \mathbb{N} : \epsilon_n = -1\},$$

y se denotan por $\{n_k^+\}$ y $\{n_k^-\}$ a las sucesiones de los elementos de F^+ y F^- respectivamente en orden creciente. Por hipótesis tanto $\sum_{k=1}^{\infty} x_{n_k^+}$ como $\sum_{k=1}^{\infty} x_{n_k^-}$ convergen y, por lo tanto,

$$\sum_{n=1}^{\infty} \epsilon_n x_n = \sum_{k=1}^{\infty} x_{n_k^+} - \sum_{k=1}^{\infty} x_{n_k^-}$$

converge también.

$III \Rightarrow II$. Sea $\sigma : \mathbb{N} \rightarrow \mathbb{N}$ una aplicación creciente. Se definen dos sucesiones, por un lado $\{\epsilon_n\}$, la sucesión constante con $\epsilon_n = 1$ para todo $n \in \mathbb{N}$ y, por otro lado $\{\kappa_n\}$, que se define como

$$\kappa_n = \begin{cases} 1, & \text{si } n = \sigma(m) \text{ para algún } m \in \mathbb{N}, \\ -1, & \text{si } n \neq \sigma(m) \text{ para todo } m \in \mathbb{N}. \end{cases}$$

Por hipótesis, la serie $\sum_{n=1}^{\infty} \epsilon_n x_n$ y $\sum_{n=1}^{\infty} \kappa_n x_n$ convergen, luego

$$\sum_{n=1}^{\infty} x_{\sigma(n)} = \frac{1}{2} \left(\sum_{n=1}^{\infty} \epsilon_n x_n + \sum_{n=1}^{\infty} \kappa_n x_n \right)$$

converge también.

$II \Rightarrow IV$. Supóngase que IV es falso. Con el mismo razonamiento usado en $I \Rightarrow IV$, se definen entonces los conjuntos $\{n_k + 1, \dots, n_{k+1}\}$ con sus respectivas particiones $A_k \cup B_k$. Se toma la aplicación $\sigma : \mathbb{N} \rightarrow \cup_{k \in \mathbb{N}} A_k$ que lista los elementos de $\cup_{k \in \mathbb{N}} A_k$ en orden creciente. Se tiene así que $\sum_{n=1}^{\infty} x_{\sigma(n)}$ no es de Cauchy, luego no converge.

$IV \Rightarrow I$. Se parte de la hipótesis de que, elegido un umbral $\varepsilon > 0$, a partir de cierto índice K de la serie, cualquier selección de elementos pesará menos que dicho umbral. El objetivo es ver que $\sum_{n=1}^{\infty} x_{\sigma(n)}$ es de Cauchy para cualquier permutación σ .

2. Bases Incondicionales

Sea $\varepsilon > 0$ y K el valor correspondiente que confirma la hipótesis, se toma entonces $N = \max\{\sigma^{-1}(1), \dots, \sigma^{-1}(K)\}$, de forma que los K primeros índices de la sucesión se encuentran en $\{\sigma(1), \dots, \sigma(N)\}$. Obsérvese que cualquier índice posterior a N , una vez permutado, será mayor que K . Se toman $L > M \geq N$ y se considera el segmento de la serie permutada $\sum_{k=M+1}^N x_{\sigma(k)}$, se define entonces $F = \{\sigma(M+1), \dots, \sigma(L)\} \subset \{K+1, K+2, \dots\}$. Por hipótesis,

$$\left\| \sum_{k=M+1}^L x_{\sigma(k)} \right\| = \left\| \sum_{k \in F} x_k \right\| < \varepsilon.$$

Luego la serie $\sum_{n=1}^{\infty} x_{\sigma(n)}$ es de Cauchy para cualquier permutación. ■

Para finalizar esta sección, se mostrará otra caracterización de la convergencia incondicional que será de utilidad para demostraciones posteriores y que puede probarse fácilmente usando el último lema.

Proposición 2.10. *Sea $\sum_{n=1}^{\infty} x_n$ una serie en un espacio de Banach X . Entonces la serie es incondicionalmente convergente si, y solo si, la serie $\sum_{n=1}^{\infty} t_n x_n$ converge incondicionalmente para cualquier sucesión $\{t_n\} \in l_{\infty}$.*

Demostración. Supóngase que $\sum_{n=1}^{\infty} x_n$ es incondicional. Sea $\{t_n\} \in l_{\infty}$ una sucesión acotada con $\|\{t_n\}\|_{\infty} = M < \infty$, se fija entonces $\varepsilon > 0$ y usando la caracterización IV del [Lema 2.9](#), existe un número natural K tal que, para todo subconjunto finito $F \subset \{K+1, K+2, \dots\}$, se cumple que

$$\left\| \sum_{k \in F} x_k \right\| < \frac{\varepsilon}{M}.$$

Entonces se puede acotar,

$$\left\| \sum_{k \in F} t_k x_k \right\| \leq \|\{t_k\}\|_{\infty} \left\| \sum_{k \in F} x_k \right\| < M \frac{\varepsilon}{M} = \varepsilon.$$

Luego la serie $\sum_{n=1}^{\infty} t_n x_n$ converge incondicionalmente.

Se considera ahora una sucesión de signos $\{\varepsilon_n\}$, que se sabe está acotada, $\|\{\varepsilon_n\}\|_{\infty} = 1$, luego $\{\varepsilon_n\} \in l_{\infty}$. Por hipótesis, $\sum_{n=1}^{\infty} \varepsilon_n x_n$ converge y, por la caracterización III del [Lema 2.9](#), la serie original converge incondicionalmente. ■

2.3. Bases incondicionales

Finalmente, se amplía el estudio de la convergencia incondicional en el contexto particular de las bases de Schauder, introduciendo la noción de base incondicional y una caracterización que facilitará el camino más adelante.

Definición 2.11. Una base $\{e_n\}$ de un espacio de Banach X se dice *incondicional* si, para cada elemento $x \in X$, la serie $\sum_{n=1}^{\infty} e_n^*(x) e_n$ converge incondicionalmente.

Proposición 2.12. Una base $\{e_n\}$ de un espacio de Banach X es incondicional si, y solo si, existe una constante $K \geq 1$ tal que para cada $N \in \mathbb{N}$ y para cualesquiera $a_1, \dots, a_N, b_1, \dots, b_N$ escalares cumpliendo $|a_k| \leq |b_k|$ con $k = 1, \dots, N$ se tiene que

$$\left\| \sum_{k=1}^N a_k e_k \right\| \leq K \left\| \sum_{k=1}^N b_k e_k \right\|. \quad (2.2)$$

Demostración. Supóngase que $\{e_n\}$ es incondicional. Si $\sum_{n=1}^{\infty} a_n e_n$ es convergente, por la [Proposición 2.10](#) se sabe que $\sum_{n=1}^{\infty} t_n a_n e_n$ es convergente también para cualquier sucesión $\{t_n\} \in \ell_{\infty}$. Se considera la aplicación lineal y continua,

$$T_{\{t_n\}} : X \rightarrow X, \quad T_{\{t_n\}} \left(\sum_{n=1}^{\infty} a_n e_n \right) = \sum_{n=1}^{\infty} t_n a_n e_n.$$

La familia de operadores $\{T_{\{t_n\}} : \|t_n\|_{\infty} \leq 1\}$ está puntualmente acotada, entonces, por el *Principio de acotación uniforme*, se deduce que $\sup\{\|T_{\{t_n\}}\| : \|t_n\|_{\infty} \leq 1\} \leq K$ con $K \geq 1$. Sean $a_1, \dots, a_N, b_1, \dots, b_N$ escalares con $|a_k| \leq |b_k|$ para todo $1 \leq k \leq N$, se toma entonces la sucesión $\{t_n\}$ dada por

$$t_k = \begin{cases} \frac{a_k}{b_k} & \text{si } b_k \neq 0, \\ 0 & \text{si } b_k = 0, \end{cases}$$

para $1 \leq k \leq N$ y $t_k = 0$ para $k > N$. Esta sucesión cumple que $|t_k| \leq 1$ para todo k y, por lo tanto, $\|\{t_n\}\|_{\infty} \leq 1$. Se puede entonces aplicar el operador $T_{\{t_n\}}$ para obtener

$$\sum_{k=1}^N a_k e_k = \sum_{k=1}^N t_k b_k e_k = T_{\{t_n\}} \left(\sum_{k=1}^{\infty} b_k e_k \right),$$

luego,

$$\left\| \sum_{k=1}^N a_k e_k \right\| = \left\| T_{\{t_n\}} \left(\sum_{k=1}^{\infty} b_k e_k \right) \right\| \leq \|T_{\{t_n\}}\|_{\infty} \left\| \sum_{k=1}^{\infty} b_k e_k \right\| \leq K \left\| \sum_{k=1}^{\infty} b_k e_k \right\|.$$

Encontrando así la constante K que satisface (2.2).

Por otro lado, sea $\sum_{n=1}^{\infty} a_n e_n$ una serie convergente en X . Vamos a probar que la subserie $\sum_{k=1}^{\infty} a_{n_k} e_{n_k}$ es convergente para cualquier sucesión parcial creciente $\{n_k\} \subset \mathbb{N}$. Por el [Lemma 2.9](#), para cada $\varepsilon > 0$ existe un número natural $N \in \mathbb{N}$ tal que, si $m_2 > m_1 \geq N$, entonces

$$\left\| \sum_{n=m_1+1}^{m_2} a_n e_n \right\| < \frac{\varepsilon}{K}.$$

Por hipótesis, se tiene que, si $N \leq n_k \leq \dots \leq n_{k+l}$, entonces

$$\left\| \sum_{j=k+1}^{k+l} a_{n_j} e_{n_j} \right\| \leq K \left\| \sum_{j=n_k+1}^{n_{k+l}} a_j e_j \right\| < \varepsilon,$$

por lo que la serie $\sum_{k=1}^{\infty} a_k e_k$ es de Cauchy. ■

2. *Bases Incondicionales*

Definición 2.13. Sea $\{e_n\}$ una base incondicional de un espacio de Banach X . La constante incondicional K_u de $\{e_n\}$ es la menor constante que cumple

$$\left\| \sum_{n=1}^N a_n e_n \right\| \leq K_u \left\| \sum_{n=1}^N b_n e_n \right\|.$$

Entonces decimos que $\{e_n\}$ es K_u -incondicional.

Sea $\{e_n\}$ una base incondicional de un espacio de Banach X .

1. **Constante de incondicionalidad universal.** Para cada sucesión de signos $\{\epsilon_n\}$ con $\epsilon_k \in \{-1, 1\}$ para todo $k \in \mathbb{N}$, se define el operador lineal

$$T_{\{\epsilon_n\}} : X \rightarrow X, \quad T_{\{\epsilon_n\}} \left(\sum_{n=1}^{\infty} a_n e_n \right) = \sum_{n=1}^{\infty} \epsilon_n a_n e_n.$$

La constante de incondicionalidad K_u de la base se define como

$$K_u = \sup \left\{ \|T_{\{\epsilon_n\}}\| : \{\epsilon_n\} \subset \{-1, 1\} \right\}.$$

Esta constante controla cuánto puede crecer la norma de un vector al cambiar arbitrariamente los signos de los coeficientes en su desarrollo en la base.

2. **Constante de supresión.** Para cada subconjunto $A \subset \mathbb{N}$, se considera la proyección natural

$$P_A(x) = \sum_{k \in A} e_k^*(x) e_k.$$

La familia de proyecciones naturales $\mathcal{P} = \{P_A : A \subset \mathbb{N}\}$ está puntualmente acotada, y, por el *Principio de acotación uniforme*, existe una constante

$$K_s = \sup \{ \|P_A\| : P_A \in \mathcal{P} \} < \infty,$$

llamada *constante de supresión* de la base. Esta constante cuantifica el crecimiento posible de la norma al suprimir arbitrariamente coordenadas en el desarrollo de un vector.

3. **Relación entre constantes.** Siempre se cumple la desigualdad,

$$1 \leq K_s \leq K_u \leq 2K_s.$$

Esta desigualdad muestra que la constante de incondicionalidad universal K_u puede controlarse a partir de la constante de supresión K_s .

3. El Sistema de Haar

Este capítulo aborda el sistema de Haar, una herramienta propia del análisis funcional cuyo papel es fundamental en este trabajo. Para motivar su desarrollo, se inicia con una revisión concisa de los conceptos esenciales de teoría de la medida.

Con estos conceptos introducidos, se profundiza sobre la esperanza condicionada. Abordando esta función como una proyección en un espacio funcional y desarrollando algunas propiedades interesantes que enriquecen dicha noción. Posteriormente se llega al concepto más importante del capítulo, el sistema de Haar. El cual permite una representación eficiente y ligera de funciones medibles.

Finalmente se trata el concepto de sucesiones de Rademacher, cuyo comportamiento probabilístico resulta clave para analizar la incondicionalidad y otras propiedades del sistema de Haar.

3.1. Conceptos de teoría de la medida

Recuérdese que para un conjunto X , si se considera el conjunto de sus partes, $\mathcal{P}(X)$, un subconjunto $\Sigma \subset \mathcal{P}(X)$ se dice que es una σ -álgebra si satisface las siguientes propiedades:

- I. $X \in \Sigma$.
- II. Si $A \in \Sigma$ entonces su complementario también, $X \setminus A \in \Sigma$.
- III. Σ es cerrado bajo uniones contables, es decir, si $\{A_k\}$ son elementos de Σ son $k \in \Lambda \subset \mathbb{N}$, entonces $\bigcup_{k \in \Lambda} A_k$ está también en Σ .

Dentro de una σ -álgebra dada Σ , un subconjunto $\Sigma' \subset \Sigma$ se dice que es una *sub- σ -álgebra* si cumple también las propiedades anteriores. Es decir, Σ' es una σ -álgebra contenida en otra sobre el mismo conjunto X . Representa una estructura que permite trabajar con una información parcial del espacio original.

Por otro lado, cuando se trabaja con espacios topológicos, si se toma la colección de todos los conjuntos abiertos de un espacio topológico como base, la σ -álgebra generada por ellos recibe el nombre de *σ -álgebra de Borel*.

Ahora, si se toma un conjunto, X , junto con una σ -álgebra, Σ , entonces (X, Σ) es un espacio medible y los elementos de Σ son conjuntos medibles. Sobre estos espacios se pueden definir funciones.

Sean (X, Σ) y (Y, Ξ) espacios medibles. Una función $g : X \rightarrow Y$ es Σ - Ξ -medible si, para cada $E \in \Xi$, la preimagen de E bajo f está en Σ . Es decir, para cada $E \in \Xi$,

$$f^{-1}(E) = \{x \in X : f(x) \in E\} \in \Sigma.$$

En tal caso, simplemente puede decirse que la función $g : (X, \Sigma) \rightarrow (Y, \Xi)$ es *medible*. Obsérvese que esta propiedad garantiza que la función respete la estructura de tales espacios.

3. El Sistema de Haar

En este caso, se considerará $Y = \mathbb{R}$ y $\Sigma = \mathcal{B}(\mathbb{R})$, siendo esta la σ -álgebra de Borel sobre \mathbb{R} , entonces simplemente se dice que $g : (X, \Sigma) \rightarrow (Y, \Sigma)$ es Σ -medible.

Sea (X, Σ) un espacio medible, una función μ definida sobre Σ y con imagen la linea real extendida, es decir, $\mathbb{R} \cup \{+\infty, -\infty\}$, se dice que es una *medida* si cumple las siguientes tres propiedades:

- I. Para cada $E \in \Sigma$, $\mu(E) \geq 0$.
- II. $\mu(\emptyset) = 0$.
- III. Para cualquier sucesión de elementos $\{E_n\}$ en Σ , disjuntos dos a dos,

$$\mu \left(\bigcup_{k=1}^{\infty} E_k \right) = \sum_{k=1}^{\infty} \mu(E_k).$$

Un espacio medible (X, Σ) junto con una medida μ se llama *espacio de medida* (X, Σ, μ) . Un caso particular es el *espacio de probabilidad*, que es un espacio de medida donde la medida representa una distribución de probabilidad sobre los conjuntos medibles de X .

Si μ y ν son dos medidas sobre un mismo espacio medible (X, Σ) , se dice que ν es *absolutamente continua* con respecto a μ si $\nu(E) = 0$ para cada conjunto $E \in \Sigma$ para el que $\mu(E) = 0$. En tal caso, se denota por $\mu \ll \nu$. Si $\mu \ll \nu$ y $\nu \ll \mu$ entonces ambas medidas son equivalentes.

Teorema 3.1 (Radon-Nikodym). *Sea (X, Σ) un espacio donde se definen dos medidas Σ -finitas, μ y ν . Entonces, si ν es absolutamente continua con respecto a μ , $\mu \ll \nu$, existe una función Σ -medible $f : X \rightarrow [0, \infty]$ tal que, para cualquier conjunto medible $E \in \Sigma$,*

$$\nu(E) = \int_E f d\mu.$$

Esta función f suele denominarse por $d\nu/d\mu$ y se denomina *derivada de Radon-Nikodym*. La notación refleja su analogía con una derivada clásica, ya que f representa la densidad de la medida ν con respecto a μ , describiendo cómo varía ν en relación con μ sobre los conjuntos medibles.

Sea (Ω, \mathcal{F}, P) un espacio de probabilidad y $X : \Omega \rightarrow \mathbb{R}$ una variable aleatoria. Se define la esperanza de cualquier función medible f respecto la variable X como,

$$E[f(X)] = \int_{\Omega} f(X(\omega)) dP(\omega).$$

3.2. Esperanza condicionada

Antes de definir la esperanza condicionada conviene matizar el espacio de Lebesgue sobre un espacio de medida genérico, $L_p(\Omega, \Sigma, \mu)$, este no es otro que aquel formado por todas las funciones f que son Σ -medibles y cuyo valor absoluto elevado a p tiene integral finita, es decir

$$L_p(\Omega, \Sigma, \mu) = \left\{ f \text{ } \Sigma\text{-medible} : \int_{\Omega} |f|^p d\mu < \infty \right\}.$$

Siendo este una generalización de L_p , donde las funciones de este último están definidas sobre el cuerpo $\mathbb{K} = \mathbb{R}$ o $\mathbb{K} = \mathbb{C}$ y son medibles en la σ -álgebra de Borel.

3.2. Esperanza condicionada

Sea (X, Σ, μ) un espacio de probabilidad y sea $\Sigma' \subset \Sigma$ una *sub- σ -álgebra* de Σ . Si $f \in L_1(X, \Sigma, \mu)$, se define una medida ν sobre Σ'

$$\nu(E) = \int_E f d\mu, \quad E \in \Sigma'.$$

Esta cumple que ν es absolutamente continua con respecto a $\mu|_{\Sigma'}$, luego por el Teorema de Radon Nikodym existe una única función $\psi \in L_1(X, \Sigma', \mu)$ tal que,

$$\nu(E) = \int_E \psi d\mu, \quad E \in \Sigma'$$

y es, además, la única función que cumple

$$\int_E f d\mu = \int_E \psi d\mu, \quad E \in \Sigma'. \quad (3.1)$$

En este caso, a ψ se le conoce como la *esperanza condicionada a la σ -álgebra Σ' de f* y se denotará por $\mathbb{E}[f|\Sigma']$.

Obsérvese que la principal diferencia entre f y ψ es que f es Σ -medible mientras que ψ simplemente es Σ' -medible. Ambas funciones son similares en media sobre los conjuntos de Σ' pero fuera de esta σ -álgebra no puede asegurarse siquiera que ψ sea Σ -medible.

Intuitivamente, la *esperanza condicionada* de f puede considerarse como la mejor aproximación de dicha función con la información contenida en Σ' . A continuación se van a presentar un par de propiedades de este nuevo concepto.

- Si Σ' está generada por una partición de X de conjuntos medibles disjuntos $\{A_n\}$, entonces la esperanza condicionada viene dada por

$$\mathbb{E}[f|\Sigma'](t) = \sum_{k=1}^{\infty} \frac{1}{\mu(A_k)} \left(\int_{A_k} f d\mu \right) \chi_{A_k}(t).$$

Como los conjuntos $\{A_k\}$ forman una partición disjunta de X , para cada $t \in X$ existe un único índice $k \in \mathbb{N}$ tal que $t \in A_k$, es decir

$$\chi_{A_k}(t) = 1 \quad y \quad \chi_{A_j}(t) = 0 \quad \text{para todo } j \neq k.$$

Obsérvese que la función $\mathbb{E}[f|\Sigma']$ toma en cada conjunto disjunto el valor promedio de f sobre el mismo, de forma que la propiedad (3.1) siga cumpliéndose.

- Si $f \in L_1(X, \Sigma, \mu)$, para cada función simple Σ' -medible g se tiene que

$$\int_X g f d\mu = \int_X g \mathbb{E}[f|\Sigma'] d\mu$$

y

$$\mathbb{E}[gf|\Sigma'] = g \mathbb{E}[f|\Sigma'].$$

Se quiere probar que, para cualquier $A \in \Sigma$ se cumpla que

3. El Sistema de Haar

$$\int_A g f d\mu = \int_A g \psi d\mu.$$

Recuérdese que una función simple viene dada por una suma finita

$$g(x) = \sum_{k=1}^N a_k \chi_{E_k}(x), \quad \text{con } E_k \in \Sigma',$$

donde $\{a_n\}$ son escalares y $\{E_n\}$ forman una partición de X . Entonces puede desarrollarse la igualdad de la siguiente forma,

$$\begin{aligned} \int_A g f d\mu &= \int_A \sum_{k=1}^n a_k \chi_{E_k}(x) f d\mu = \sum_{k=1}^n a_k \int_{E_k \cap A} f d\mu = \sum_{k=1}^n a_k \int_{E_k \cap A} \psi d\mu \\ &= \int_A \sum_{k=1}^n a_k \chi_{E_k}(x) \psi d\mu = \int_A g \psi d\mu. \end{aligned}$$

Válido para cualquier $A \in \Sigma'$ y, por lo tanto, se concluye que $\mathbb{E}[gf|\Sigma'] = g\psi = g\mathbb{E}[f|\Sigma']$.

Cuando no haya duda sobre el espacio medible sobre el que se está trabajando y el desarrollo se centre en la medida a usar se denotará simplemente por $L_p(\mu) := L_p(\Omega, \Sigma, \mu)$. Mientras que si lo relevante es la σ -álgebra se denotará por $L_p(\Sigma) := L_p(\Omega, \Sigma, \mu)$.

Lema 3.2. *Sean (X, Σ, μ) un espacio de probabilidad y $\Sigma' \subset \Sigma$ una sub- σ -álgebra. Entonces para cada $1 \leq p \leq \infty$, se tiene que $\mathbb{E}(\cdot|\Sigma')$ es una proyección lineal normalizada de $L_p(X, \Sigma, \mu)$ en $L_p(X, \Sigma', \mu)$.*

Demostración. Por un lado, la propiedad $\mathbb{E}(\cdot|\Sigma')^2 = \mathbb{E}(\cdot|\Sigma')$, es sencilla de probar. Para $f \in L_p(X, \Sigma, \mu)$ si se denota $\psi = \mathbb{E}(f|\Sigma')$, entonces $\mathbb{E}(\mathbb{E}(f|\Sigma')|\Sigma') = \mathbb{E}(\psi|\Sigma')$ debe cumplir que

$$\int_A \psi d\mu = \int_A \mathbb{E}[\psi|\Sigma'] d\mu \quad \forall A \in \Sigma'.$$

Pero ψ ya es Σ' -medible luego, por unicidad, debe de darse que $\mathbb{E}(f|\Sigma')^2 = \mathbb{E}(\psi|\Sigma') = \psi = \mathbb{E}(f|\Sigma')$. Respecto a la linealidad, se prueba inmediatamente por la linealidad de la integral.

Para $1 \leq p \leq \infty$ fijo, si $h \in L_p(\Sigma')$, como consecuencia del *teorema de extensión de Hahn-Banach*, su norma puede expresarse como

$$\|h\|_p = \sup\{|T(h)| : T \in L_p^*(\Sigma'), \|T\| \leq 1\},$$

Sean $1 < p, q < \infty$ con $1/p + 1/q = 1$, para cada $h \in L_p(\Sigma')$ y $g \in L_q(\Sigma')$, entonces la aplicación

$$T(h) = \int_X hg d\mu,$$

es un operador funcional. De hecho, la aplicación $g \mapsto T$ es un isomorfismo isométrico de $L_q(\Sigma')$ en el dual de $L_p(\Sigma')$.

Se considera ahora que g es una función en $L_q(\Sigma')$ con $\|g\|_q \leq 1$, usando la desigualdad de Hölder se tiene que

$$|T(h)| = \left| \int_X hg \, d\mu \right| \leq \|h\|_p \|g\|_q \leq \|h\|_p$$

luego $\|T\| = \sup\{|T(h)| : h \in L_p(\Sigma'), \|h\| \leq 1\} \leq 1$ y puede confirmarse

$$\|h\|_p = \sup \left\{ \left| \int_X hg \, d\mu \right| : g \text{ función simple } \Sigma'\text{-medible con } \|g\|_q \leq 1 \right\}.$$

Como $\mathbb{E}(f|\Sigma') \in L_p(\Sigma')$, su norma puede calcularse tomando el supremo sobre funciones $g \in L_q(\Sigma')$ con $\|g\|_q \leq 1$. Dado que las funciones simples Σ' -medibles son densas en $L_q(\Sigma')$, basta considerar funciones simples en el supremo y, como la elección de g es libre, el valor absoluto se puede omitir.

$$\begin{aligned} \|\mathbb{E}(f|\Sigma')\|_p &= \sup \left\{ \int_X \mathbb{E}(f|\Sigma')g \, d\mu : g \text{ función simple } \Sigma'\text{-medible con } \|g\|_q \leq 1 \right\} \\ &= \sup \left\{ \int_X fg \, d\mu : g \text{ función simple } \Sigma'\text{-medible con } \|g\|_q \leq 1 \right\} \\ &\leq \sup \left\{ \int_X fg \, d\mu : g \text{ función simple con } \|g\|_q \leq 1 \right\} = \|f\|_p. \end{aligned}$$

Donde la desigualdad surge por eliminar la restricción de ser Σ' -medible a las funciones simples.

Para el caso $p = \infty$, recuérdese que, si $f \in L^\infty(\Sigma)$, su norma viene dada por $\|f\|_\infty = \inf\{C \geq 0 : |f(x)| \leq C \text{ p.c.t. } x \in X\}$.

Se toma a continuación $M > \|f\|_\infty$, entonces se cumple que $|f(x)| \leq M$ para casi todo $x \in X$. Por lo tanto,

$$\mathbb{E}(f|\Sigma')(x) \leq \mathbb{E}(M|\Sigma')(x) = M$$

De forma análoga, como $-f \leq M$, se tiene que

$$-\mathbb{E}(f|\Sigma')(x) = \mathbb{E}(-f|\Sigma')(x) \leq M,$$

Luego,

$$|\mathbb{E}(f|\Sigma')| \leq M \quad \text{c.t.p.}$$

Como esto es válido para todo $M > \|f\|_\infty$, se concluye que

$$\|\mathbb{E}(f|\Sigma')\|_\infty \leq \|f\|_\infty.$$

3. El Sistema de Haar

3.3. Sistema de Haar

Definición 3.3. Sea $\{h_n\}$ la sucesión de funciones en $[0, 1]$ definida por $h_1 = 1$ y para $k \in \mathbb{N}$, $s \in \{1, 2, \dots, 2^k\}$, $n = 2^k + s$

$$h_n(t) = \begin{cases} 1 & \text{si } t \in \left[\frac{2s-2}{2^{k+1}}, \frac{2s-1}{2^{k+1}}\right], \\ -1 & \text{si } t \in \left[\frac{2s-1}{2^{k+1}}, \frac{2s}{2^{k+1}}\right], \\ 0 & \text{en otro caso.} \end{cases}$$

A la base que forman estas funciones se le llama *sistema de Haar*.

Dados $k \in \mathbb{N}$, $s \in \{1, 2, \dots, 2^k\}$, todo intervalo de la forma

$$\left[\frac{2s-2}{2^{k+1}}, \frac{2s-1}{2^{k+1}}\right] \cup \left[\frac{2s-1}{2^{k+1}}, \frac{2s}{2^{k+1}}\right] = \left[\frac{s-1}{2^k}, \frac{s}{2^k}\right] = I_{k,s}$$

se llamará *intervalo diádico*. También se denotará directamente, en caso de ser necesario, por I_n con $n = 2^k + s$.

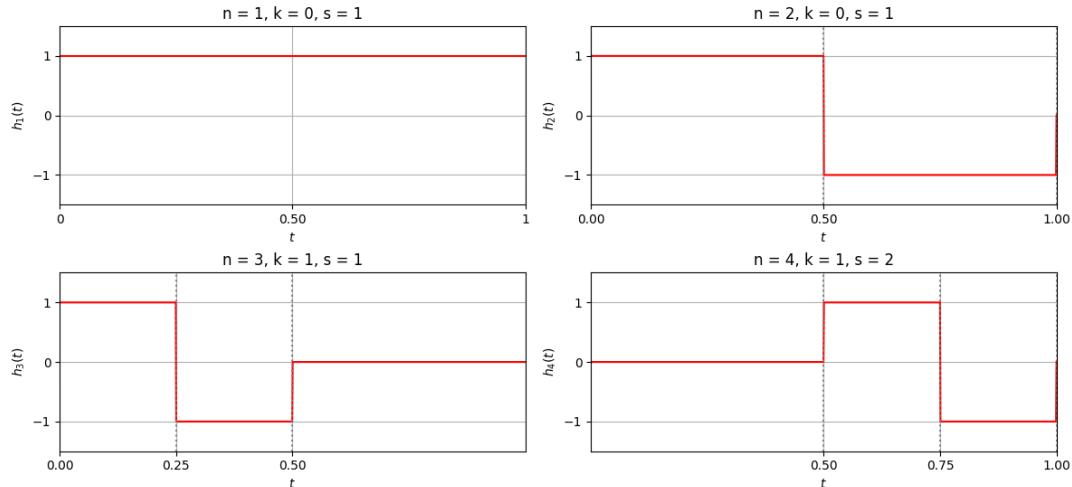


Figura 3.1.: Primeras funciones del sistema de Haar a partir de $n=2$.

Obsérvese que este sistema se puede definir como diferencias de funciones características,

$$h_n(t) = \chi_{\left[\frac{2s-2}{2^{k+1}}, \frac{2s-1}{2^{k+1}}\right]} - \chi_{\left[\frac{2s-1}{2^{k+1}}, \frac{2s}{2^{k+1}}\right]}$$

Proposición 3.4. El sistema de Haar es una base monótona en L_p para $1 \leq p < \infty$.

Demostración. Sea $\mathcal{B}[0, 1]$ la σ -álgebra de Borel, y sea $\{\mathcal{B}_n\} \subset \mathcal{B}([0, 1])$ la sucesión de σ -álgebras definida de la siguiente forma. Si $n = 1$, se toma la σ -álgebra trivial, $\mathcal{B}_1 = \{\emptyset, [0, 1]\}$. Para $n = 2^k + s$, con $k \in \mathbb{N} \cup \{0\}$ y $s \in \{1, 2, \dots, 2^k\}$, \mathcal{B}_n es la sub- σ -álgebra de $\mathcal{B}[0, 1]$ generada por los intervalos diádicos de la siguiente familia

$$\mathcal{F}_n = \left\{ \begin{array}{ll} \left[\frac{j-1}{2^{k+1}}, \frac{j}{2^{k+1}} \right] & , j = 1, 2, \dots, 2s, \\ \left[\frac{j-1}{2^k}, \frac{j}{2^k} \right] & , j = s+1, 2, \dots, 2^k. \end{array} \right.$$

Obsérvese que se están agrupando los intervalos diádicos por escalas de resolución o frecuencias y se está tomando la σ -álgebra que generan.

Se fija a continuación $1 \leq p < \infty$, para cada $n \in \mathbb{N}$, \mathbb{E}_n será la esperanza condicionada en la σ -álgebra \mathcal{B}_n . Por el [Lema 3.2](#), \mathbb{E}_n es la proyección normalizada de $L_p([0, 1])$ sobre $L_p([0, 1], \mathcal{B}_n, \lambda)$, el espacio de funciones constantes en los intervalos de \mathcal{F}_n . De hecho, viene dada por

$$\mathbb{E}_n[f|\mathcal{B}_n](x) = \frac{1}{|I|} \int_I f(t) dt, \quad \text{si } x \in I \in \mathcal{F}_n$$

Se denotará por $L_p(\mathcal{B}_n) := \{f \in L_p([0, 1]) : f|_I \text{ constante para cada } I \in \mathcal{F}_n\}$. Está claro que el rango de \mathbb{E}_n es n , además, $\mathbb{E}_n \mathbb{E}_m = \mathbb{E}_m \mathbb{E}_n = \mathbb{E}_{\min\{m,n\}}$, $\forall m, n \in \mathbb{N}$.

Por otro lado, como $\sup\{\|\mathbb{E}_n\| : n \in \mathbb{N}\} < \infty$, usando la consecuencia parcial del *principio de acotación uniforme*, [Proposición 1.3](#), se llega a que el conjunto

$$\{f \in L_p : \|\mathbb{E}_n(f) - f\| \rightarrow 0\}$$

es cerrado y contiene al conjunto $\cup_{k=1}^{\infty} L_p(\mathcal{B}_k)$, el cual es denso en L_p . Por lo tanto, se tiene que $\|\mathbb{E}_n(f) - f\| \rightarrow 0$, para todo $f \in L_p$.

Por la [Proposición 2.5](#), L_p tiene una base cuyas proyecciones naturales son $\{\mathbb{E}_n\}$. Esta base es el sistema de Haar ya que, para cada $n \in \mathbb{N}$, $\mathbb{E}_m[h_n] = h_n$ si $m \geq n$ y $\mathbb{E}_m[h_n] = 0$ si $m < n$. De hecho, la constante base es $\sup\{\|\mathbb{E}_n\| : n \in \mathbb{N}\} = 1$. ■

El sistema de Haar como se ha definido está normalizado en $L_p([0, 1])$ para $p = \infty$. Sin embargo, para $1 \leq p < \infty$, se tiene que para cada $k \in \mathbb{N}$ y cada $s \in \{1, 2, \dots, 2^k\}$ se cumple que $\|h_n\| = (1/|I_n|)^{1/p} = (1/2^k)^{1/p}$ con $n = 2^k + s$. Para normalizar el sistema en tal caso basta tomar

$$h_n^p = \frac{h_n}{\|h_n\|} = \frac{1}{|I_n|^{1/p}} h_n.$$

Por otro lado, se deduce que los funcionales duales asociados al sistema de Haar son

$$h_n^* = \frac{1}{|I_n|} h_n, \quad n \in \mathbb{N},$$

pues se cumple que

$$\langle h_n, h_n^* \rangle = \int_{I_n} h_n h_n^* dt = \frac{1}{|I_n|} \int_{I_n} h_n h_n dt = \frac{1}{|I_n|} |I_n| = 1,$$

y

$$\langle h_m, h_n^* \rangle = \int_{I_m} h_m h_n^* dt = \frac{1}{|I_m|} \int_{I_m} h_m h_n dt = 0, \quad \text{si } m \neq n.$$

Obsérvese que si $f \in L_p([0, 1])$ con $1 \leq p < \infty$, como

$$\mathbb{E}_n[f] = \sum_{k=1}^n \langle f, h_k^* \rangle h_k,$$

se tiene que

$$\mathbb{E}_n[f] - \mathbb{E}_{n-1}[f] = \langle f, h_n^* \rangle h_n = \left(\frac{1}{|I_n|} \int_0^1 f(t) h_n(t) dt \right) h_n,$$

y la expansión en serie de $f \in L_p([0, 1])$ viene dada por

$$f = \lim_{n \rightarrow \infty} \mathbb{E}_n[f] = \sum_{k=1}^{\infty} (\mathbb{E}_n[f] - \mathbb{E}_{n-1}[f]) = \sum_{k=1}^{\infty} \left(\frac{1}{|I_k|} \int_0^1 f(t) h_k(t) dt \right) h_k.$$

El siguiente paso será probar que la base de Haar es una base incondicional de $L_p([0, 1])$ cuando $1 < p < \infty$. Para ello, será necesarios dos lemas que se presentan y prueban a continuación.

Lema 3.5. *Sea $p > 2$ y $\frac{1}{p} + \frac{1}{q} = 1$. Entonces, para $0 \leq t \leq 1$, se cumple*

$$t^p - p^p q^{-p} (1-t)^p \leq p^2 q^{1-p} \left(t - \frac{1}{q} \right).$$

Demostración. Para $0 \leq t \leq 1$ se define la función,

$$f(t) = t^p - p^p q^{-p} (1-t)^p - p^2 q^{1-p} \left(t - \frac{1}{q} \right),$$

cuya primera y segunda derivada vienen dadas por,

$$f'(t) = pt^{p-1} + p^{p+1} q^{-p} (1-t)^{p-1} - p^2 q^{1-p},$$

$$f''(t) = p(p-1)t^{p-2} - p^{p+1}(p-1)q^{-p}(1-t)^{p-2}.$$

Se observa que, $f(0) = -p^p q^{-p} + p^2 q^{-p} = q^{-p}(p^2 - p^p) < 0$ puesto que $p > 2$. Además, $f(1) = 1 - p^2 q^{1-p} \left(1 - \frac{1}{q} \right) = 1 - p^2 q^{1-p} \frac{1}{p} = 1 - pq^{1-p}$. Se tiene que $pq^{1-p} = p \left(\frac{1}{q} \right)^{p-1} = p \left(1 - \frac{1}{p} \right)^{p-1} > 1$ para $p > 2$, por lo tanto,

$$f(0) < 0, \quad f(1) < 0.$$

Por otro lado,

$$f \left(\frac{1}{q} \right) = q^{-p} - p^p p^{-p} q^{-p} = 0,$$

$$\begin{aligned} f' \left(\frac{1}{q} \right) &= pq^{1-p} + p^{p+1} p^{1-p} q^{-p} - p^2 q^{1-p} \\ &= pq^{1-p} + p^2 q^{-p} - p^2 q^{1-p} = pq^{-p}(q(1-p) + p) \\ &= pq^{-p} \left(p \frac{1-p}{p-1} + p \right) = pq^{-p}(-p + p) = 0, \end{aligned}$$

$$\begin{aligned} f''\left(\frac{1}{q}\right) &= (p-1)pq^{2-p} - (p-1)p^{p+1}p^{2-p}q^{-p} \\ &= (p-1)\left(pq^{2-p} - p^3q^{-p}\right) = (p-1)pq^{-p}(q^2 - p^2) < 0, \end{aligned}$$

puesto que $q < p$. Poniendo todo en común,

$$f(0) < 0, \quad f(1) < 0,$$

$$f\left(\frac{1}{q}\right) = 0, \quad f'\left(\frac{1}{q}\right) = 0, \quad f''\left(\frac{1}{q}\right) < 0.$$

Supóngase a continuación que f toma un valor positivo para algún punto del intervalo abierto, $s \in]0, 1[$. Como $f(0) < 0$ y $f(1) < 0$, deben existir al menos dos puntos dentro del intervalo $]0, 1[$ donde la función se anule. Sin embargo, el punto $1/q$ no es una opción a pesar de anular la función, pues se trata de un máximo local. Se tiene así que $f(t) = 0$ tiene al menos tres soluciones en el intervalo $]0, 1[$.

Por el *Teorema de Rolle*, existen mínimo dos soluciones de $f'(t) = 0$ en los intervalos abiertos que generan dichas tres soluciones. Esto es, hay tres puntos en $]0, 1[$ que anulan $f'(t) = 0$, las dos soluciones mencionadas y $1/q$, que no puede considerarse solución tras la aplicación del *Teorema de Rolle* por ser algún extremo de los intervalos.

Finalmente, tras volver a aplicar dicho teorema, se concluye que deben existir dos puntos que cumplan $f''(t) = 0$. La función $f''(t)$ puede expresarse como,

$$f''(t) = Pt^{p-2} - Q(1-t)^{p-2},$$

con $P = p(p-1)$ y $Q = p^{p+1}(p-1)q^{-p}$, ambos positivos por ser $p > 2$. Entonces,

$$f''(t) = 0 \iff Pt^{p-2} = Q(1-t)^{p-2} \iff g(t) = \left(\frac{t}{1-t}\right)^{p-2} = \frac{Q}{P}.$$

Pero la función $g(t)$ es monótona creciente en el intervalo $[0, 1]$. Para probarlo basta con estudiar su derivada y ver que todos los términos son positivos. Llegando así a una contradicción, pues la ecuación $f''(t) = 0$ no puede tener dos soluciones en dicho intervalo. Por lo que no existe ningún punto $t \in]0, 1[$ que haga a f positiva y,

$$t^p - p^p q^{-p} (1-t)^p \leq p^2 q^{1-p} \left(t - \frac{1}{q}\right), \quad \forall t \in [0, 1].$$

■

Lema 3.6. Sea $p > 2$ y sea $\phi : \mathbb{R}^2 \Rightarrow \mathbb{R}$ dada por

$$\phi(x, y) = (|x| + |y|)^{p-1}((p-1)|x| - |y|)$$

entonces,

I. Si $\frac{1}{p} + \frac{1}{q} = 1$, se cumple para todo $(x, y) \in \mathbb{R}^2$,

$$(p-1)^p |x|^p - |y|^p \geq pq^{1-p} \phi(x, y).$$

3. El Sistema de Haar

II. ϕ es dos veces diferenciable y satisface la condición

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 2 \left\| \frac{\partial^2 \phi}{\partial x \partial y} \right\| \geq 0. \quad (3.2)$$

Demostración. I. Se realiza el cambio de variable $t = |y|/(|x| + |y|)$ en el Lema 3.5, con $x \neq 0, y \neq 0$,

$$\left(\frac{|y|}{|x| + |y|} \right)^p - p^p q^{-p} \left(1 - \frac{|y|}{|x| + |y|} \right)^p \leq p^2 q^{1-p} \left(\frac{|y|}{|x| + |y|} - \frac{1}{q} \right),$$

$$\left(\frac{|y|}{|x| + |y|} \right)^p - p^p q^{-p} \left(\frac{|x|}{|x| + |y|} \right)^p \leq p^2 q^{1-p} \left(\frac{|y|}{|x| + |y|} - \frac{1}{q} \right),$$

$$\begin{aligned} |y|^p - p^p q^{-p} |x|^p &\leq p^2 q^{1-p} \left(\frac{|y|}{|x| + |y|} - \frac{1}{q} \right) (|x| + |y|)^p \\ &= p^2 q^{1-p} \left(\frac{|y|}{|x| + |y|} - 1 + \frac{1}{p} \right) (|x| + |y|)^p \\ &= p^2 q^{1-p} \left(\frac{-|x|}{|x| + |y|} + \frac{1}{p} \right) (|x| + |y|)^p \\ &= p^2 q^{1-p} \frac{(1-p)|x| + |y|}{p(|x| + |y|)} (|x| + |y|)^p \\ &= pq^{1-p} ((1-p)|x| + |y|) (|x| + |y|)^{p-1}. \end{aligned}$$

Teniendo en cuenta que $(p/q)^p = (p(1-1/p))^p = (p-1)^p$,

$$|y|^p - (p-1)^p |x|^p \leq pq^{1-p} ((1-p)|x| + |y|) (|x| + |y|)^{p-1}.$$

Por lo que,

$$(p-1)^p |x|^p - |y|^p \geq pq^{1-p} ((p-1)|x| - |y|) (|x| + |y|)^{p-1} = pq^{1-p} \phi(x, y).$$

II. Por ser $p > 2$, se trata de una función continua dos veces derivable. Para la igualdad basta con probarla únicamente en el primer cuadrante, donde $x > 0, y > 0$. Sea $u = x + y$ y $v = (p-1)x - y$, entonces $\phi(x, y) = u^{p-1}v$. Por ello,

$$\frac{\partial \phi}{\partial x} = (p-1)u^{p-2}v + u^{p-1}(p-1),$$

$$\frac{\partial^2 \phi}{\partial x^2} = (p-2)(p-1)u^{p-3}v + 2(p-1)^2u^{p-2},$$

y

$$\frac{\partial \phi}{\partial y} = (p-1)u^{p-2}v - u^{p-1},$$

$$\frac{\partial^2 \phi}{\partial y^2} = (p-2)(p-1)u^{p-3}v - 2(p-1)u^{p-2}.$$

Por lo que

$$\begin{aligned}\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} &= 2(p-2)(p-1)u^{p-3}v + 2(p-2)(p-1)u^{p-2} \\ &= 2(p-2)(p-1)u^{p-3}(v+u) \geq 0.\end{aligned}$$

Por otro lado

$$\begin{aligned}\frac{\partial^2 \phi}{\partial x \partial y} &= (p-2)(p-1)u^{p-3}v - (p-1)u^{p-2} + (p-1)^2u^{p-2} \\ &= (p-2)(p-1)u^{p-3}v + (p-2)(p-1)u^{p-2} \\ &= (p-2)(p-1)u^{p-3}(v+u),\end{aligned}$$

luego se cumple (3.2). ■

Ahora sí se tiene todo lo necesario para demostrar la incondicionalidad del sistema de Haar.

Teorema 3.7. *Supongamos $1 < p < \infty$, tal que $\frac{1}{p} + \frac{1}{q} = 1$, y sea $p^* = \max\{p, q\}$. La base de Haar $\{h_n\}$ es incondicional con constante incondicional como mucho $p^* - 1$. Es decir,*

$$\left\| \sum_{j=1}^n \epsilon_j a_j h_j \right\|_p \leq (p^* - 1) \left\| \sum_{j=1}^n a_j h_j \right\|_p, \quad \forall n \in \mathbb{N},$$

para todo a_1, \dots, a_n y para cualesquiera signos $\epsilon_1, \dots, \epsilon_n$.

Demostración. Sea $p > 2$, luego $p^* = p$. Para cada $n \in \mathbb{N}$, sea $f_0 = g_0 = 0$ y para $1 \leq m \leq n$ se define

$$f_m = \sum_{k=1}^m a_k h_k, \quad g_m = \sum_{k=1}^m \epsilon_k a_k h_k.$$

Se va a probar por inducción sobre m que

$$\int_0^1 \phi(f_m(s), g_m(s)) ds \geq 0, \quad 1 \leq m \leq n. \quad (3.3)$$

Siendo ϕ la función de Lema 3.6.

- El caso $m = 0$ es trivial.
- Para un determinado valor de m , se define la función $F : [0, 1] \rightarrow \mathbb{R}$, dada por

$$F(t) = \int_0^1 \phi((1-t)f_{m-1}(s) + t f_m(s), (1-t)g_{m-1}(s) + t g_m(s)) ds$$

y, si se asume que $F(0) \geq 0$, cumple que $F(1) \geq 0$.

- Se denota por $u_t = (1-t)f_{m-1} + t f_m$ y $v_t = (1-t)g_{m-1} + t g_m$,

3. El Sistema de Haar

$$F'(t) = a_m \int_0^1 \frac{\partial \phi}{\partial x}(u_t, v_t) h_m ds + \epsilon_m a_m \int_0^1 \frac{\partial \phi}{\partial y}(u_t, v_t) h_m ds.$$

Obsérvese que $F'(0) = 0$ ya que $\frac{\partial \phi}{\partial x}(u_0, v_0)$ y $\frac{\partial \phi}{\partial y}(u_0, v_0)$ son constantes en el intervalo del soporte de h_m . Diferenciando de nuevo se obtiene

$$F''(t) = a_m^2 \int_0^1 \left(\frac{\partial^2 \phi}{\partial x^2}(u_t, v_t) + \frac{\partial^2 \phi}{\partial y^2}(u_t, v_t) + 2\epsilon_m \int_0^1 \frac{\partial^2 \phi}{\partial y \partial x}(u_t, v_t) \right) ds.$$

Por el [Lema 3.6 II](#), se cumple que $F''(t) \geq 0$. Por lo que $F(1) \geq F(0)$ y, por lo tanto, se cumple [\(3.3\)](#).

Para terminar la demostración con $p > 2$ se toman $x = f_n$ e $y = g_n$ en [Lema 3.6 I](#). Integrando ambas partes de la desigualdad y usando [\(3.3\)](#) se obtiene

$$\int_0^1 ((p-1)^p |f_n(s)|^p - |g_n(s)|^p) ds \geq 0.$$

Para $1 < p < 2$ se cumple análogamente. Con f_n, g_n igual que antes, tomamos $g'_n \in L_q(\mathcal{B}_n)$ tal que $\|g'_n\|_q = 1$ y

$$\int_0^1 g'_n(s) g_n(s) ds = \|g_n\|_p.$$

Entonces $g'_n(s) = \sum_{j=1}^n b_j h_j$ para algunos $\{b_1, \dots, b_n\}$ y

$$\|g_n\|_p = \sum_{j=1}^n |I_j| \epsilon_j a_j b_j \leq \|f_n\|_p \left\| \sum_{j=1}^n \epsilon_j b_j h_j \right\|_q \leq (q-1) \|f_n\|_p.$$

■

Probar que para el caso $p = 1$ el sistema de Haar no es una base incondicional es sencillo.

Proposición 3.8. *La base de Haar no es incondicional para $p = 1$.*

Demostración. Para este caso, se van a identificar a los elementos del sistema de Haar por sus soportes y f_N denotará la función cuya expansión con respecto a la base de Haar viene dada por,

$$f_N = 2^{1-2N} \chi_{[0,1]} + \sum_{k=0}^{2N} 2^{k+1-2N} h_{[0,2^{-k}[}.$$

Sea

$$g_N = \sum_{k=0}^{2N} 2^{k+1-2N} h_{[0,2^{-k}[},$$

esta función cumple que

$$g_N(t) = -2^{2k+1-2N} \quad \text{para } 2^{-2k-1} \leq t \leq 2^{-2k} \text{ y } 0 \leq k \leq N.$$

Por lo tanto,

$$\|g_N\|_1 \geq \sum_{k=0}^N 2^{2k+1-2N} 2^{-2k-1} = (N+1)2^{-2N} = (N+1)\|f_N\|_1.$$

Luego, el sistema de Haar no puede ser incondicional para $p = 1$. ■

3.4. Sucesiones de Rademacher

Para finalizar este capítulo se introduce un nuevo tipo de sucesiones junto con algunas desigualdades asociadas a ellas. El estudio de estas sucesiones será de utilidad para analizar y reforzar propiedades del sistema de Haar en el siguiente capítulo.

Definición 3.9. Sea $\{r_k\}$ la sucesión de funciones definidas en $[0, 1]$ dadas por

$$r_k(t) = \text{signo}(\sin 2^k \pi t),$$

a estas funciones se les conoce como *funciones de Rademacher*. Estas toman únicamente los valores ± 1 y alternan su signo 2^k veces en el intervalo $[0, 1]$.

También pueden definirse de forma constructiva sobre los intervalos diádicos de la siguiente manera:

$$\begin{aligned} r_1(t) &= \begin{cases} 1 & t \in [0, \frac{1}{2}[\\ -1 & t \in [\frac{1}{2}, 1[\end{cases} \\ r_2(t) &= \begin{cases} 1 & t \in [0, \frac{1}{4}[\cup [\frac{1}{2}, \frac{3}{4}[\\ -1 & t \in [\frac{1}{4}, \frac{1}{2}[\cup [\frac{3}{4}, 1[\end{cases} \\ &\dots \\ r_{k+1}(t) &= \begin{cases} 1 & t \in \bigcup \left[\frac{2s-2}{2^{k+1}}, \frac{2s-1}{2^{k+1}} \right[\\ -1 & t \in \bigcup \left[\frac{2s-1}{2^{k+1}}, \frac{2s}{2^{k+1}} \right[\end{cases}. \end{aligned}$$

Como se puede observar, las funciones de Rademacher están estrechamente relacionadas con el sistema de Haar. En efecto, para cada $k \in \mathbb{N}$, se puede expresar

$$r_{k+1} = \sum_{s=1}^k h_{2^k+s}.$$

Es decir, cada función de Rademacher coincide con la suma de todos los elementos del sistema de Haar que corresponden al mismo nivel de resolución. Esto permite interpretar a la sucesión $\{r_k\}$ como un *bloque básico* con respecto a la base de Haar en $L_p([0, 1])$, para cada $1 \leq p < \infty$.

Teorema 3.10. (*Desigualdad de Khintchine*) Para cada $1 \leq p < \infty$ existen constantes positivas A_p

3. El Sistema de Haar

y B_p tales que, para cualesquiera $\{a_1, \dots, a_N\}$ escalares, se cumple

$$A_p \left(\sum_{k=1}^N |a_k|^2 \right)^{\frac{1}{2}} \leq \left\| \sum_{k=1}^N a_n r_n \right\| \leq \left(\sum_{k=1}^N |a_k|^2 \right)^{\frac{1}{2}}, \quad 1 \leq p < 2,$$

y

$$\left(\sum_{k=1}^N |a_k|^2 \right)^{\frac{1}{2}} \leq \left\| \sum_{k=1}^N a_n r_n \right\| \leq B_p \left(\sum_{k=1}^N |a_k|^2 \right)^{\frac{1}{2}}, \quad p > 2.$$

Es decir, cualquier combinación lineal de funciones de Rademacher tiene norma controlada por la norma euclídea de los coeficientes. La constante sirve para modificar el rango en el que dicha norma puede variar y su comportamiento varía en $p = 2$.

Obsérvese que la desigualdad de Khintchine muestra que la sucesión $\{r_k\}$ es una base ortonormal en L_2 y, además, es equivalente a la base canónica de ℓ_2 en $L_p([0, 1])$ para todo $1 \leq p < \infty$, en el sentido de normas. Es decir, cualquier combinación lineal finita de funciones de Rademacher tiene su norma controlada por la norma euclídea de sus coeficientes, con constantes independientes del número de términos.

A pesar de su ortonormalidad en L_2 , la sucesión $\{r_k\}$ no es completa en este espacio. Por ejemplo, la función $r_1 \cdot r_2$ es ortogonal a todo el espacio generado por $\{r_k\}$, lo que muestra que $[\{r_k\}] \subsetneq L_2$.

Sin embargo, es posible construir un sistema ortonormal completo para $L_2([0, 1])$ a partir de las funciones de Rademacher. Basta con añadir la función constante $r_0 = 1$, así como productos finitos de la forma $r_{k_1} r_{k_2} \dots r_{k_n}$ con $k_1 < k_2 < \dots < k_n$.

En el caso $L_\infty([0, 1])$, la sucesión $\{r_k\}$ ya no es equivalente a la base de ℓ_2 , pero sí se puede demostrar que es isométricamente equivalente a la base canónica de ℓ_1 .

La desigualdad de Khintchine puede también interpretarse diciendo que todas las normas $\{\|\cdot\|_p : 1 \leq p < \infty\}$ son equivalentes en el cierre lineal de las funciones de Rademacher en L_p . Este hecho se puede generalizar a espacios de Banach arbitrarios.

Definición 3.11. Se conoce como *sucesión de Rademacher* a la sucesión de variables aleatorias mutuamente independientes $\{\mathcal{R}_k\}$, definidas en algún espacio de probabilidad (Ω, \mathcal{F}, P) , tales que $P[\mathcal{R}_k = 1] = P[\mathcal{R}_k = -1] = \frac{1}{2}$ para todo $k \in \mathbb{N}$.

Se observa que las funciones de Rademacher $\{r_k\}$ forman una sucesión de variables aleatorias sobre $[0, 1]$, equivalentes en distribución a una familia de variables de Rademacher $\{\mathcal{R}_k\}$ sobre algún espacio de probabilidad. De este modo, para cualquier conjunto de elementos $\{x_1, x_2, \dots, x_N\}$ de un espacio de Banach X se cumple

$$\int_0^1 \left\| \sum_{k=1}^N r_k(t) x_k \right\| dt = E \left[\left\| \sum_{k=1}^N \mathcal{R}_k x_k \right\| \right] = \int_{\Omega} \left\| \sum_{k=1}^N \mathcal{R}_k(\omega) x_k \right\| dP.$$

Para finalizar, se va a estimar $\|(\sum_{k=1}^n |f_k|^2)^{\frac{1}{2}}\|_p$ con las medias de Rademacher, es decir con $E[\|\sum_{k=1}^N \mathcal{R}_k f_k\|_p^p]^{\frac{1}{p}}$, en un espacio $L_p(\mu)$ genérico.

Teorema 3.12. Sea $1 \leq p < \infty$, para cada conjunto finito de funciones $\{f_1, f_2, \dots, f_N\}$ en $L_p(\mu)$ se cumple la siguiente desigualdad

$$A_p \left\| \left(\sum_{k=1}^N |f_k|^2 \right)^{\frac{1}{2}} \right\|_p \leq \left(E \left[\left\| \sum_{k=1}^N \mathcal{R}_k f_k \right\|_p^p \right] \right)^{\frac{1}{p}} \leq B_p \left\| \left(\sum_{k=1}^N |f_k|^2 \right)^{\frac{1}{2}} \right\|_p.$$

Siendo A_p y B_p las constantes de la desigualdad de Khintchine, en concreto, $A_p = 1$ para $1 \leq p \leq 2$ y $B_p = 1$ para $2 \leq p < \infty$.

Demostración. Para cada $\omega \in \Omega$, de la desigualdad de Khintchine se tiene que,

$$\begin{aligned} A_p \left(\sum_{k=1}^N |f_k(\omega)|^2 \right)^{\frac{1}{2}} &\leq \left\| \sum_{k=1}^N r_k f_k(\omega) \right\|_p \\ &= \left(\int_0^1 \left| \sum_{k=1}^N r_k(t) f_k(\omega) \right|^p dt \right)^{\frac{1}{p}} \\ &= \left(E \left[\left\| \sum_{k=1}^N \mathcal{R}_k f_k(\omega) \right\|_p^p \right] \right)^{\frac{1}{p}}, \end{aligned}$$

con $A_p = 1$ para $2 \leq p < \infty$. Elevando a p , integrando en el espacio de medida y usando el Teorema de Fubini para intercambiar la integral de la medida con la esperanza se tiene,

$$\begin{aligned} A_p^p \int_{\Omega} \left(\sum_{k=1}^N |f_k(\omega)|^2 \right)^{\frac{1}{2}} d\mu &= A_p^p \left\| \left(\sum_{k=1}^N |f_k|^2 \right)^{\frac{1}{2}} \right\|_p^p \leq \int_{\Omega} E \left[\left\| \sum_{k=1}^N \mathcal{R}_k f_k(\omega) \right\|_p^p \right] d\mu \\ &= E \left[\int_{\Omega} \left\| \sum_{k=1}^N \mathcal{R}_k f_k(\omega) \right\|_p^p d\mu \right] = E \left[\left\| \sum_{k=1}^N \mathcal{R}_k f_k \right\|_p^p \right]. \end{aligned}$$

La otra estimación es similar, usando la otra desigualdad de Khintchine se obtiene para cada $\omega \in \Omega$,

$$\left\| \sum_{k=1}^N r_k f_k(\omega) \right\|_p = \left(E \left[\left\| \sum_{k=1}^N \mathcal{R}_k f_k(\omega) \right\|_p^p \right] \right)^{\frac{1}{p}} \leq B_p \left(\sum_{k=1}^N |f_k(\omega)|^2 \right)^{\frac{1}{2}}.$$

Y elevando a p , integrando y con el Teorema de Fubini se consigue la otra parte de la desigualdad

$$E \left[\left\| \sum_{k=1}^N \mathcal{R}_k f_k \right\|_p^p \right] \leq B_p^p \int_{\Omega} \left(\sum_{k=1}^N |f_k(\omega)|^2 \right)^{\frac{1}{2}} d\mu = B_p^p \left\| \left(\sum_{k=1}^N |f_k|^2 \right)^{\frac{1}{2}} \right\|_p^p.$$

La demostración concluye juntando ambas estimaciones y elevando a $1/p$,

3. El Sistema de Haar

$$A_p \left\| \left(\sum_{k=1}^N |f_k|^2 \right)^{\frac{1}{2}} \right\|_p \leq \left(E \left[\left\| \sum_{k=1}^N \mathcal{R}_k f_k \right\|_p^p \right] \right)^{\frac{1}{p}} \leq B_p \left\| \left(\sum_{k=1}^N |f_k|^2 \right)^{\frac{1}{2}} \right\|_p.$$

■

4. Aproximando con Bases Greedy

Este capítulo estudia en detalle las bases *greedy*, comenzando con la introducción de conceptos clave como la aproximación mediante combinaciones lineales y el error óptimo de aproximación para un número fijo de términos.

Posteriormente, se presenta la noción formal de base *greedy* y, con ayuda del concepto de base *democrática*, se establece una caracterización útil que permite determinar cuándo una base posee dicha propiedad. Esta caracterización se utiliza finalmente para demostrar que el sistema de Haar es una base *greedy*.

4.1. Bases Greedy

Supóngase que $\{e_n\}$ es una base seminormalizada en un espacio de Banach X , es decir, existe $c > 0$ tal que $1/c \leq \|e_n\| \leq c$ para todo $n \in \mathbb{N}$, entonces se denota por Σ_m al conjunto de todas las combinaciones lineales de m elementos de la base, esto es,

$$\Sigma_m := \left\{ y = \sum_{k \in B} a_k e_k : B \subset \mathbb{N}, |B| = m, a_k \in \mathbb{R} \right\}.$$

Se define el *error de la mejor aproximación de m términos* de $x \in X$ como

$$\sigma_m(x) := \inf\{\|x - y\| : y \in \Sigma_m\}.$$

Este valor representa el menor error posible de obtener al aproximar x mediante una combinación de m vectores de la base.

El objetivo es construir un algoritmo que, para cada elemento $x \in X$ y cada $m \in \mathbb{N}$, devuelva un representante $y_m \in \Sigma_m$ cuyo error de aproximación sea uniformemente comparabale con dicho error mínimo $\sigma_m(x)$, es decir que

$$\|x - y_m\| \leq C\sigma_m(x)$$

siendo $C \in \mathbb{R}^+$ una constante independiente de x y m .

Véase con un ejemplo, considérese el caso en que H es un espacio de Hilbert y $\{e_n\}$ una base ortonormal en tal espacio. Cualquier elemento $x \in H$ se puede expresar como

$$x = \sum_{n=1}^{\infty} \langle x, e_n \rangle e_n,$$

y cumple

$$\|x\|^2 = \sum_{n=1}^{\infty} |\langle x, e_n \rangle|^2.$$

4. Aproximando con Bases Greedy

Una estrategia natural para obtener la mejor aproximación sería ordenar los coeficientes $\langle x, e_n \rangle$ por valor absoluto decreciente. Si Λ_m es el conjunto de índices de los m primeros coeficientes, la mejor aproximación sería

$$y_m = \sum_{k \in \Lambda_m} \langle x, e_k \rangle e_k$$

con un error

$$\sigma_m(x)^2 = \|x - y_m\|^2 = \sum_{k \notin \Lambda_m} |\langle x, e_k \rangle|^2.$$

Esta construcción es bastante intuitiva y muestra cómo los términos más significativos pueden proporcionar una buena aproximación en norma.

Se desea generalizar este ejemplo. Para ello, sea $\{G_m\}$ un sucesión de aplicaciones de X en X , donde para cada elemento $x \in X$, $G_m(x)$ se calcula tomando los mejores m coeficientes de x , es decir

$$G_m(x) = \sum_{k \in B} e_k^*(x) e_k,$$

siendo $B \subset \mathbb{N}$ el conjunto formado por los índices de los m funcionales biortogonales asociados a la base cuya imagen es mayor en valor absoluto al aplicarlos sobre x . Cabe destacar que este conjunto no tiene porque ser único y las aplicaciones $\{G_m\}$ no son lineales ni continuas.

Dado que el comportamiento de $\{G_m\}$ depende de $\{e_n\}$, se introduce un nuevo tipo de bases esenciales para este procedimiento.

Definición 4.1. Una base $\{e_n\}$ se dice *greedy* si existe una constante $C \geq 1$ tal que, para cada $x \in X$ y $m \in \mathbb{N}$ se cumple,

$$\|x - G_m(x)\| \leq C\sigma_m(x).$$

A la menor constante C que verifica la desigualdad anterior se la denomina *constante de greedy* de la base.

4.2. Caracterización de las bases greedy

Antes de proceder a la caracterización de las bases greedy, es necesario introducir un concepto fundamental que interviene de manera directa en dicha caracterización.

Definición 4.2. Una base $\{e_n\}$ se dice *democrática* si existe una constante $D \geq 1$ tal que para cualesquiera dos conjuntos finitos $A, B \subset \mathbb{N}$ con el mismo número de elementos, $|A| = |B|$, se tiene que,

$$\left\| \sum_{k \in A} e_k \right\| \leq D \left\| \sum_{k \in B} e_k \right\|.$$

Esta propiedad garantiza que todos los conjuntos de elementos de la base de una misma cardinalidad generen combinaciones cuyas normas estén controladas por una constante, sin depender de los elementos como tal.

Por ejemplo, la base usual de ℓ_p para $1 \leq p < \infty$ es un caso sencillo de base democrática. Dado cualquier conjunto finito $A \subset \mathbb{N}$ con $|A| = m$, se tiene

$$\left\| \sum_{k \in A} e_k \right\| = m^{1/p},$$

que evidentemente no depende del conjunto A , sino del número de elementos.

A continuación, se presenta un resultado que será útil al final de este capítulo cuando se trate de demostrar que el sistema de Haar es una base greedy.

Lema 4.3. *Sea B una base democrática en un espacio de Banach X y sea \hat{B} una base de un espacio de Banach finito dimensional Y . Entonces la suma directa de B y \hat{B} es una base democrática en $X \oplus Y$.*

Obsérvese que, en un espacio de dimensión finita, el número de subconjuntos de la base con un mismo número de elementos es finito y, por lo tanto, la suma directa con un espacio finito dimensional no introduce inestabilidad en la comparación de normas.

Puede formularse ahora un teorema que establece condiciones necesarias y suficientes para que una base sea greedy.

Teorema 4.4 (Caracterización de Bases greedy). *Una base $\{e_n\}$ es Greedy si, y solo si, es incondicional y democrática.*

Demostración. Supóngase que $\{e_n\}$ es greedy, con constante greedy C . Se va a probar primero la incondicionalidad, es decir, para cada $x \in X$ y $S \subset \mathbb{N}$ se considera la proyección

$$P_S(x) = \sum_{n \in S} e_n^*(x) e_n,$$

y se debe cumplir que,

$$\|P_S(x)\| \leq (C + 1)\|x\|.$$

Se toma entonces $S \subset \mathbb{N}$ un conjunto de índices fijo con cardinalidad $|S| = m$. Para cada $x \in X$ se elige $\alpha > \sup\{|e_n^*(x)| : n \notin S\}$ y se considera el vector

$$y = x - P_S(x) + \alpha \sum_{n \in S} e_n$$

que cumple,

$$\sigma_m(y) := \inf\{\|y - z_m\| : z_m \in \Sigma_m\} \leq \|y - \alpha \sum_{n \in S} e_n\| = \|x - P_S(x)\| \leq \|x\|.$$

Obsérvese que para y , los mayores m coeficientes $|e_k^*(y)|$ son todos α mientras que el resto de coeficientes serán como mucho $e_k^*(x)$, menores que α pues $\alpha > \sup\{|e_n^*(x)| : n \notin S\} = \sup\{|e_n^*(x - P_S(x))| : n \in \mathbb{N}\}$. Se tiene así que la mejor aproximación de y viene dada por $G_m(y) = \alpha \sum_{n \in S} e_n$.

En conclusión, se ha elegido el vector y estratégicamente para que su error de aproximación sea menor que la norma de x y su mejor aproximación esté controlada:

$$\sigma_m(y) \leq \|x\|, \quad G_m(y) = \alpha \sum_{n \in S} e_n.$$

Como $\{e_n\}$ es greedy, se cumple

4. Aproximando con Bases Greedy

$$\|x - P_S(x)\| = \|y - G_m(y)\| \leq C\sigma_m(y) \leq C\|x\|$$

y como $\|P_S(x)\| = \|x - (x - P_S(x))\| \leq \|x\| + \|x - P_S(x)\|$, es inmediato que

$$\|P_S(x)\| \leq (C+1)\|x\|.$$

Quedando así demostrada la incondicionalidad.

Se prueba a continuación la democracia. Se consideran los subconjuntos arbitrarios $P, Q \subset \mathbb{N}$, con misma cardinalidad $|P| = |Q| = m$ y sea $S \subset \mathbb{N}$, $|S| = m$ otro conjunto tal que $P \cap S = \emptyset = Q \cap S$. Para cada $\varepsilon > 0$ se define el vector

$$x = (1 + \varepsilon) \sum_{k \in P} e_k + \sum_{k \in S} e_k,$$

cuyo mínimo error de m términos viene dado por

$$\sigma_m(x) := \inf\{\|x - z_m\| : z_m \in \Sigma_m\} \leq \left\| x - \sum_{k \in S} e_k \right\| = (1 + \varepsilon) \left\| \sum_{k \in P} e_k \right\|$$

y su mejor aproximación es $G_m(x) = (1 + \varepsilon) \sum_{k \in P} e_k$, por lo que

$$\left\| \sum_{k \in S} e_k \right\| = \|x - G_m(x)\| \leq C\sigma_m(x) \leq C(1 + \varepsilon) \left\| \sum_{k \in P} e_k \right\|.$$

De forma análoga se puede obtener que,

$$\left\| \sum_{n \in Q} e_n \right\| \leq C(1 + \varepsilon) \left\| \sum_{n \in S} e_n \right\|.$$

Como ε es todo lo pequeño que se quiera,

$$\left\| \sum_{n \in Q} e_n \right\| \leq C^2 \left\| \sum_{n \in P} e_n \right\|.$$

Es decir, para cualesquiera dos conjuntos de índices con la misma cardinalidad, la propia constante Greedy mantiene la democracia de la base. Quedando así probada esta propiedad.

Supóngase ahora que $\{e_n\}$ es *K-incondicional* y *D-democrática*. Sean $x \in X$, $m \in \mathbb{N}$, para cada $\varepsilon > 0$ se toma

$$p_m = \sum_{n \in B} a_n e_n,$$

siendo $B \subset \mathbb{N}$ un conjunto de cardinalidad m , de forma que

$$\|x - p_m\| \leq \sigma_m(x) + \varepsilon.$$

Por otro lado,

$$G_m(x) = \sum_{n \in S} e^*(x) e_n = P_S(x)$$

para algún $S \subset \mathbb{N}$, con $|S| = m$, entonces

$$\|x - G_m(x)\| = \|x - P_S(x) + P_B(x) - P_B(x)\| = \|x - P_B(x) + P_{B \setminus S}(x) - P_{S \setminus B}(x)\|.$$

Como $\{e_n\}$ es K -incondicional, se tiene que

$$\begin{aligned} \|x - P_B(x) - P_{S \setminus B}(x)\| &= \|x - P_{B \cup S}(x)\| \\ &= \|P_{\mathbb{N} \setminus (B \cup S)}(x)\| \\ &= \|P_{\mathbb{N} \setminus (B \cup S)}(x - p_m)\| \\ &\leq K\|x - p_m\| \leq K(\sigma_m(x) + \varepsilon) \end{aligned}$$

y

$$\|P_{S \setminus B}(x)\| = \|P_{S \setminus B}(x - p_m)\| \leq K\|x - p_m\| \leq K(\sigma_m(x) + \varepsilon).$$

De la definición de G_m se obtiene que

$$\gamma := \min\{|e_k^*(x)| : k \in S \setminus B\} \geq \max\{|e_k^*(x)| : k \in B \setminus S\} := \beta,$$

pues en caso contrario, dicho máximo se debería alcanzar en un funcional de G_m .

Con todo esto se tiene

$$\gamma \left\| \sum_{k \in S \setminus B} e_k \right\| \leq K\|P_{S \setminus B}(x)\| \quad (4.1)$$

y

$$\|P_{B \setminus S}(x)\| \leq K\beta \left\| \sum_{k \in B \setminus S} e_k \right\|. \quad (4.2)$$

Como $|B \setminus S| = |S \setminus B|$, usando la D -democracia y juntandola con (4.1) y (4.2), se consigue

$$\|P_{B \setminus S}(x)\| \leq K\beta \left\| \sum_{k \in B \setminus S} e_k \right\| \leq K\gamma \left\| \sum_{k \in B \setminus S} e_k \right\| \leq KD\gamma \left\| \sum_{k \in S \setminus B} e_k \right\| \leq K^2 D \|P_{S \setminus B}(x)\|,$$

y juntando todo finalmente

$$\begin{aligned} \|x - G_m(x)\| &\leq \|x - P_B(x) - P_{S \setminus B}(x)\| + \|P_{B \setminus S}(x)\| \leq K(\sigma_m(x) + \varepsilon) + K^2 D \|P_{S \setminus B}(x)\| \\ &\leq K(\sigma_m(x) + \varepsilon) + K^3 D (\sigma_m(x) + \varepsilon) \leq (K + K^3 D)(\sigma_m(x) + \varepsilon). \end{aligned}$$

Tomando límite $\varepsilon \rightarrow 0$ se tiene que

$$\|x - G_m(x)\| \leq (K + K^3 D)\sigma_m(x)$$

y la base $\{e_n\}$ es greedy. ■

4. Aproximando con Bases Greedy

4.3. El caso de la base de Haar

Este capítulo se concluye demostrando que el sistema de Haar es una base greedy. Esta propiedad será de gran utilidad para poder guardar información de forma eficiente, tanto temporal como de frecuencia, de las series temporales con las que se trabajarán. Para ello el siguiente lema sobre series geométricas será de gran utilidad.

Lema 4.5. *Sean $1 < r < \infty$ y $0 < p < \infty$, existen constantes positivas c_{rp} y C_{rp} tales que, para cualquier conjunto finito $A \subset \mathbb{N}$, se cumple que*

$$c_{rp} \left(\sum_{k \in A} r^{pk} \right)^{\frac{1}{p}} \leq \left(\sum_{k \in A} r^{2k} \right)^{\frac{1}{2}} \leq C_{rp} \left(\sum_{k \in A} r^{pk} \right)^{\frac{1}{p}}.$$

Este lema permitirá, asociando puntualmente las funciones del sistema de Haar con una serie geométrica de radio $r = 2$, obtener una desigualdad de gran ayuda para la demostración.

Ahora sí se puede presentar el siguiente resultado.

Proposición 4.6. *El sistema de Haar normalizado $\{h_n^p\}$ es una base greedy en $L_p[0, 1]$ con $1 < p < \infty$.*

Demostración. Ya se vió en el [Teorema 3.7](#) que el sistema de Haar es una base incondicional, el siguiente objetivo será probar que además es una base democrática para poder aplicar el [Teorema 4.4](#) y concluir así la demostración.

Por el [Lema 4.3](#) es suficiente estimar $\|\sum_{n \in A} h_n^p\|$ para subconjuntos finitos $A \subset \mathbb{N} \setminus \{1\}$, es decir, podemos realizar la partición $\{h_1^p\} \cup \{h_n^p\}$ con $n \geq 2$ y estudiar simplemente el segundo caso.

Obsérvese que estas funciones $\{h_n^p\}$ toman solo los valores $0, 2^{n/p}$ y $-2^{n/p}$, y que para cada $t \in [0, 1]$ solo una de dichas funciones alcanza $2^{k/p}$ en valor absoluto para algún $k \in \mathbb{N}$. Tomando $r = 2^{1/p}$, se puede usar el [Lema 4.5](#) previamente presentado y concluir que existen constantes c_p y C_p positivas de forma que, para cada subconjunto $A \subset \mathbb{N} \setminus \{1\}$ y para cada $t \in [0, 1]$, teniendo en cuenta que $|h_k^p(t)|^p = 2^k = r^{pk}$ y $|h_k^p(t)|^2 = 2^{2k/p} = r^{2k}$, se tiene

$$c_p \left(\sum_{k \in A} |h_k^p(t)|^p \right)^{\frac{1}{p}} \leq \left(\sum_{k \in A} |h_k^p(t)|^2 \right)^{\frac{1}{2}} \leq C_p \left(\sum_{k \in A} |h_k^p(t)|^p \right)^{\frac{1}{p}}. \quad (4.3)$$

Tomando la norma en $L_p([0, 1])$ se obtiene

$$\left\| \left(\sum_{k \in A} |h_k^p(t)|^p \right)^{\frac{1}{p}} \right\|_p = \left(\int_0^1 \sum_{k \in A} |h_k^p(t)|^p dt \right)^{\frac{1}{p}} = \left(\sum_{k \in A} \int_0^1 |h_k^p(t)|^p dt \right)^{\frac{1}{p}}$$

y, como $\|h_n^p\|_p = 1$,

$$\left(\sum_{k \in A} \int_0^1 |h_k^p(t)|^p dt \right)^{\frac{1}{p}} = |A|^{\frac{1}{p}}.$$

Con esto la desigualdad (4.3) tras aplicarle la norma queda,

$$c_p |A|^{\frac{1}{p}} \leq \left\| \left(\sum_{k \in A} |h_k^p(t)|^2 \right)^{\frac{1}{2}} \right\|_p \leq C_p |A|^{\frac{1}{p}}. \quad (4.4)$$

Por otro lado, el [Teorema 3.12](#) y la incondicionalidad de $\{h_n^p\}$ para $1 < p < \infty$ nos dicen que existen constantes A'_p y B'_p tales que, para cada sucesión $\{a_n\} \in c_{00}$, se tiene

$$A'_p \left\| \left(\sum_{k=1}^{\infty} |a_k|^2 |h_k^p|^2 \right)^{\frac{1}{2}} \right\|_p \leq \left\| \sum_{k=1}^{\infty} a_k h_k^p \right\|_p \leq B'_p \left\| \left(\sum_{k=1}^{\infty} |a_k|^2 |h_k^p|^2 \right)^{\frac{1}{2}} \right\|_p.$$

Gracias a la incondicionalidad se sabe que la convergencia de la serie no se ve afectada por los signos de la sucesión de Rademacher $\{\mathcal{R}_n\}$ y la norma queda controlada.

Si a continuación se toma la sucesión dada por

$$a_n = \begin{cases} 1, & n \in A, \\ 0, & n \notin A, \end{cases}$$

es evidente que,

$$\left\| \left(\sum_{k=1}^{\infty} |a_k|^2 |h_k^p|^2 \right)^{\frac{1}{2}} \right\|_p = \left\| \left(\sum_{k \in A} |h_k^p|^2 \right)^{\frac{1}{2}} \right\|_p, \quad \left\| \sum_{k=1}^{\infty} a_k h_k \right\|_p = \left\| \sum_{k \in A} h_k \right\|_p.$$

Combinando esto con (4.4) se tiene la desigualdad

$$A'_p c_p |A|^{\frac{1}{p}} \leq A'_p \left\| \left(\sum_{k \in A} |h_k^p|^2 \right)^{\frac{1}{2}} \right\|_p \leq \left\| \sum_{k \in A} h_k^p \right\|_p \leq B'_p \left\| \left(\sum_{k \in A} |h_k^p|^2 \right)^{\frac{1}{2}} \right\|_p \leq B'_p C_p |A|^{\frac{1}{p}}.$$

Para cualquier subconjunto $A \subset \mathbb{N} \setminus \{1\}$, lo que implica la democracia de $\{h_n^p\}$, pues las normas quedan acotadas por la cantidad de elementos en el conjunto. ■

En este capítulo se ha desarrollado la teoría de las bases *greedy*, comenzando con la noción de error de mejor aproximación mediante combinaciones de un número fijo de términos. A través de la caracterización en términos de incondicionalidad y democracia, se ha establecido un criterio robusto para identificar cuándo una base permite una reconstrucción eficiente basada en los coeficientes más relevantes.

Como aplicación central, se ha demostrado que el sistema de Haar constituye una base *greedy* en $L^p([0, 1])$ para $(1 < p < \infty)$, lo que garantiza su utilidad en contextos donde se requiere representar funciones de forma comprimida y estable, preservando al mismo tiempo información estructural tanto en el dominio del tiempo como en el de la frecuencia.

Este resultado refuerza la relevancia del sistema de Haar dentro del análisis funcional, y proporciona una base teórica sólida para su uso en algoritmos de compresión y análisis de señales. En el siguiente capítulo se van a desarrollar mejor estos conceptos, desde un punto de vista más intuitivo y algorítmico. De esta forma estaremos capacitados para comprender mejor las ideas que fundamentan el modelo desarrollado y el potencial de este.

5. La transformada de Haar

El objetivo principal de este capítulo es describir brevemente la herramienta utilizada para establecer la conexión entre la parte matemática y la informática de este trabajo. Para ello, se realizará un repaso general de la teoría implicada, sin entrar en detalles de rigor matemático. Los conceptos aquí introducidos han sido adquiridos con el apoyo de los libros [\[Wal02\]](#) y [\[Sal07\]](#). Formalizar este capítulo supondría, en sí mismo, otro trabajo distinto, de extensión comparable o incluso superior al desarrollado en este proyecto.

Cuando se observa una función, se están leyendo los distintos valores que esta toma a lo largo del tiempo. Sin embargo, es bien conocido que puede analizarse desde el punto de vista de sus frecuencias. Para dicho propósito, la herramienta de referencia es la transformada de Fourier. Esta permite descomponer una señal en una suma de senos y cosenos, proporcionando información en el dominio de frecuencias.

Uno de los principales inconvenientes de esta transformada es que elimina toda información relativa al dominio temporal. Surge así la transformada de Gabor, que utiliza una ventana fija trasladada a lo largo del dominio para obtener información temporal en una frecuencia determinada. Esto supuso un avance en el análisis de señales, al permitir una representación conjunta en tiempo y frecuencia, aunque con menor resolución en ambos dominios por separado.

Para paliar esta limitación, aparece la transformada wavelet, cuya principal diferencia es la incorporación de un parámetro de escala variable. Es decir, en lugar de fijar una frecuencia y analizar su evolución temporal, se permite escalar en frecuencia y, simultáneamente, extraer información del dominio temporal. La [Figura 5.1](#) aporta una idea intuitiva de esto.

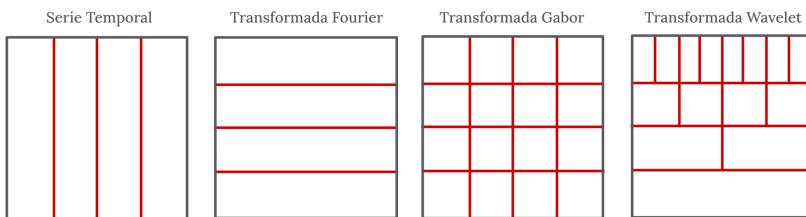


Figura 5.1.: En orden de izquierda a derecha puede apreciarse la información que se obtiene con cada representación. El dominio temporal está representado en el eje horizontal y el de frecuencias en el vertical.

Las siguientes secciones tienen como objetivo mostrar conceptualmente este tipo de transformada, evaluando sus propiedades más relevantes y cómo estas justifican las decisiones tomadas en este trabajo. Se definirá brevemente la transformada wavelet y algunas de sus propiedades fundamentales, para finalizar con el caso específico de la transformada de Haar. Sobre esta se mostrarán algunos ejemplos experimentales y se comentará la conexión entre esta herramienta y el desarrollo informático del proyecto.

5.1. Transformada Wavelet

El principal aporte de esta transformada es que busca una información más equilibrada. Las frecuencias bajas tienden a variar menos a lo largo del tiempo, luego la información que pueden aportar sobre el dominio temporal es escasa, a diferencia de las frecuencias altas. Estas tienden a variar considerablemente a lo largo de un periodo, siendo posible extraer mayor información temporal.

Para conseguir esto, este tipo de transformadas se basa en unas funciones específicas llamadas *wavelets*. Estas pueden generalizarse de la siguiente forma. Primero, es necesario tomar una función $\psi(t)$ que será la llamada *wavelet madre* y, a partir de esta, se define la familia de funciones

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right), \quad a, b \in \mathbb{R}, \quad a \neq 0.$$

Esta familia depende de dos parámetros: el parámetro a , que controla la escala de la función, y el parámetro b , que controla su traslación. Aquí aparece la principal diferencia con la transformada de Gabor: mientras que esta solo admitía un parámetro de traslación, ahora se ha añadido un nuevo parámetro para controlar la escala que se está analizando.

Es importante destacar que hay distintos tipos de wavelets dependiendo del propósito que se busque alcanzar. En la [Figura 5.2](#) pueden observarse algunos de estos tipos. En este trabajo se ha optado por usar la wavelet de Haar por diversos motivos. Entre ellos, se trata de la wavelet más básica, aportando buenas propiedades a la vez que sencillez. Como se verá en la segunda parte de este trabajo, se desarrolla un modelo desde cero y añadir complejidad estudiando diferentes wavelets no haría otra cosa que desviar la atención del objetivo final.

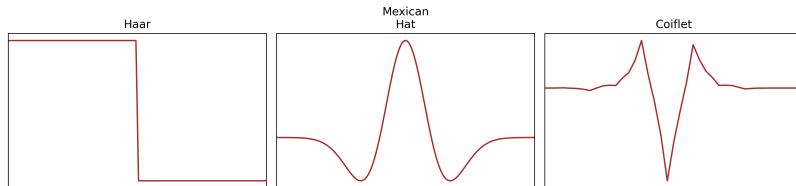


Figura 5.2.: Algunas wavelets comunes.

Cuando se pretende transformar una función a su dominio de frecuencias, se está usando la siguiente integral:

$$(Tf)(y) = \int_{-\infty}^{\infty} K(t, y) f(t) dt.$$

La entrada es una función f y la salida es otra función Tf . Esta transformada está determinada por la función $K(t, y)$, llamada *núcleo* o *kernel*. Si se busca una transformada de Fourier, el núcleo a usar sería $K(t, y) = e^{-2\pi iyt}$. Para realizar la transformada wavelet, el núcleo sería la propia función $\psi_{a,b}(t)$, pero la elección de ψ no es arbitraria. Para poder considerar una función como wavelet madre se debe cumplir un criterio de admisibilidad. El criterio consiste en que la siguiente constante debe ser finita:

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\widehat{\psi}(\omega)|^2}{|\omega|} d\omega < \infty,$$

donde se tiene que

$$\widehat{\psi}_{a,b}(\omega) = \frac{1}{\sqrt{|a|}} e^{-ib\omega} \widehat{\psi}(a\omega),$$

siendo $\widehat{\psi}$ la transformada de Fourier de la wavelet madre.

Si dicho criterio se cumple, entonces se puede definir la transformada wavelet como

$$W_\psi[f](a, b) = \langle f, \psi_{a,b} \rangle = \int_{-\infty}^{\infty} f(t) \psi_{a,b}(t) dt.$$

Donde cada parámetro a y b define la frecuencia y el intervalo de tiempo donde se va a calcular la transformada.

Transformada Wavelet Discreta

Al trabajar con datos discretos, aplicar la definición continua de la transformada wavelet no es práctico. Evaluar la integral para muchos valores de escala y posición es computacionalmente costoso. Por eso, se usa la transformada wavelet discreta (DWT), que permite una implementación eficiente a través de filtros y submuestreo.

Dada una función discretizada $f[n]$, el algoritmo aplica dos filtros:

- Un filtro de paso bajo $h[n]$ para obtener los coeficientes de aproximación a , que recogen la parte más general de la señal.
- Un filtro de paso alto $g[n]$ para obtener los coeficientes de detalle d , que recogen los cambios rápidos o las variaciones locales.

La operación consiste en una convolución seguida de submuestreo por 2, de forma que, en cada iteración se vaya cambiando la frecuencia sobre la que se están pasando los filtros. Si se parte de $a_{(0)}[n] = f[n]$, entonces los coeficientes de aproximación y de detalle del primer nivel serían:

$$\begin{aligned} a_{(1)}[k] &= \sum_n a_{(0)}[n] \cdot h[2k - n], \\ d_{(1)}[k] &= \sum_n a_{(0)}[n] \cdot g[2k - n]. \end{aligned}$$

A partir de ahí, se repite el mismo proceso de manera iterada sobre el coeficiente de aproximación anterior. Es decir, se usaría $a_{(1)}$ para obtener $a_{(2)}$ y $d_{(2)}$, y así sucesivamente. En general, los coeficientes de un nivel de escala s determinado vienen dados por:

$$\begin{aligned} a_{(s)}[k] &= \sum_n a_{(s-1)}[n] \cdot h[2k - n], \\ d_{(s)}[k] &= \sum_n a_{(s-1)}[n] \cdot g[2k - n]. \end{aligned}$$

5. La transformada de Haar

A cada paso, los coeficientes de aproximación van resumiendo la señal en menor resolución, mientras que los de detalle recogen la diferencia entre escalas. Al final, la señal queda representada como una aproximación en el nivel superior, es decir menor frecuencia, más los detalles en distintos niveles.

El algoritmo, por tanto, devuelve una serie de coeficientes de aproximación $a_{(1)}, \dots, a_{(M)}$, junto con una serie de coeficientes de detalle $d_{(1)}, \dots, d_{(M)}$. La transformada wavelet de la serie queda resumida por los coeficientes de aproximación del nivel M , junto con todos los coeficientes de detalle de todos los niveles. Es decir, la función discretizada $f[n]$ se representa mediante el conjunto $\{a_{(M)}, d_{(1)}, \dots, d_{(M)}\}$.

5.2. Wavelet de Haar

Como ya se ha mencionado, hay una gran variedad de transformadas wavelet y, en este trabajo, se ha optado por usar la transformada de Haar. En el [Capítulo 3](#) el sistema de Haar se definió como

$$h_n(t) = \begin{cases} 1 & \text{si } t \in \left[\frac{2s-2}{2^{k+1}}, \frac{2s-1}{2^{k+1}} \right], \\ -1 & \text{si } t \in \left[\frac{2s-1}{2^{k+1}}, \frac{2s}{2^{k+1}} \right], \\ 0 & \text{en otro caso,} \end{cases}$$

donde $k \in \mathbb{N}$, $s \in \{1, 2, \dots, 2^{k-1}\}$ y $n = 2^k + s$. La wavelet madre viene dada por los primeros valores posibles, $k = 1$ y $s = 1$, luego $\phi(t) = h_1(t)$. Los distintos valores de a y b van asociando cada función $\psi_{a,b}$ con una función h_n .

La wavelet de Haar puede aplicarse para calcular la transformada de una función, ya que cumple el criterio de admisibilidad. La transformada de Fourier de la wavelet madre viene dada por

$$\widehat{\psi}(\omega) = ie^{-i\omega/2} \frac{\sin^2(\omega/4)}{\omega/4},$$

y la constante C_ψ está acotada:

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\widehat{\psi}(\omega)|^2}{|\omega|} d\omega = 16 \int_{-\infty}^{\infty} \frac{1}{|\omega|^3} |\sin(\omega/4)|^4 d\omega < \infty.$$

Sin embargo, como se ha visto antes, por temas de computabilidad lo mejor es usar la transformada de Haar discreta. En este caso, los filtros a usar son muy simples, lo cual es una de las ventajas principales de esta transformada:

- El filtro paso bajo, para las tendencias globales, es decir las aproximaciones, viene dado por:

$$h[n] = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right].$$

- El filtro paso alto, para las variaciones locales, los detalles, se define:

$$g[n] = \left[\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \right].$$

De esta forma, si se parte de la función discretizada $a_0[n] = f[n]$, los primeros coeficientes se obtienen mediante las expresiones:

$$\begin{aligned} a_{(1)}[k] &= \frac{1}{\sqrt{2}}(a_0[2k] + a_0[2k+1]), \\ d_{(1)}[k] &= \frac{1}{\sqrt{2}}(a_0[2k] - a_0[2k+1]). \end{aligned}$$

Generalizando para cualquier nivel de escala s , se pueden obtener sus coeficientes mediante:

$$\begin{aligned} a_{(s)}[k] &= \frac{1}{\sqrt{2}}(a_{(s-1)}[2k] + a_{(s-1)}[2k+1]), \\ d_{(s)}[k] &= \frac{1}{\sqrt{2}}(a_{(s-1)}[2k] - a_{(s-1)}[2k+1]). \end{aligned}$$

La transformada de Haar permite construir una representación jerárquica de la serie temporal, donde la información queda organizada por niveles de resolución. Esta estructura multiescala resulta especialmente relevante en el contexto de este trabajo, ya que, como se explicará en detalle en el [Capítulo 7](#), el modelo propuesto se apoya en los coeficientes de detalle asociados a diversos intervalos diádicos que contienen a un segmento determinado de la serie.

La idea central consiste en aplicar la transformada de Haar a la serie y, a partir de un nivel de descomposición específico, extraer los coeficientes de detalle correspondientes. Estos coeficientes definen una jerarquía natural en la señal, segmentándola en intervalos. Dicha jerarquización permite asignar a cada segmento de la serie no solo sus valores locales, sino también información contextual relacionada con el comportamiento global de la serie en escalas superiores.

Más concretamente, dado un segmento fijo, lo que se propone es identificar todos los intervalos diádicos que lo contienen y tomar los coeficientes de detalle asociados a dichos intervalos. Esta información multiescala se vectoriza y se incorpora como entrada a un modelo de aprendizaje automático que será introducido más adelante. En la [Figura 5.3](#) se muestra esquemáticamente la estructura jerárquica inducida por la transformada de Haar. Además, puede observarse cómo un segmento concreto queda contenido en distintos intervalos y aporta, por tanto, información estructural hacia múltiples escalas.

El principio que motiva esta estrategia es que el comportamiento de un segmento de la serie no puede comprenderse únicamente desde su escala local; también resulta fundamental considerar cómo se inserta en estructuras de mayor tamaño. Los coeficientes de detalle correspondientes a los intervalos que contienen dicho segmento encapsulan precisamente esta información contextual. En la [Figura 5.4](#) se observa cómo un único segmento puede contribuir a la reconstrucción de tendencias globales en la serie. Como es esperable, su aportación es más significativa en las regiones próximas, donde su influencia es mayor.

Finalmente, cabe señalar el papel que desempeña el hecho de que el sistema de Haar constituya una base greedy. Esta propiedad permite que, incluso al recortar ciertos niveles para llevar a cabo la segmentación, sea posible conservar una representación razonablemente precisa de la serie a partir de los coeficientes restantes. Además, los coeficientes de mayor magnitud

5. La transformada de Haar

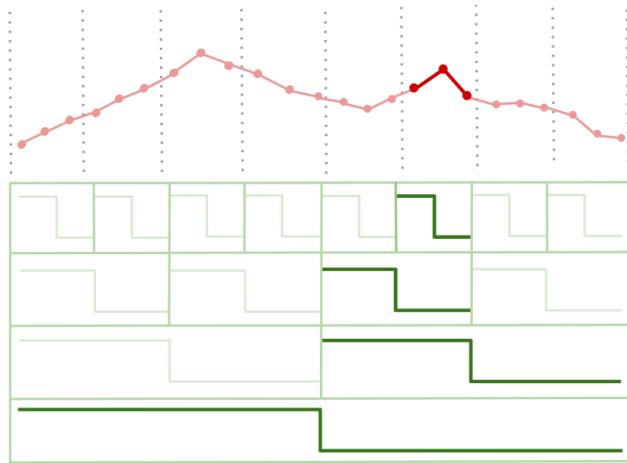
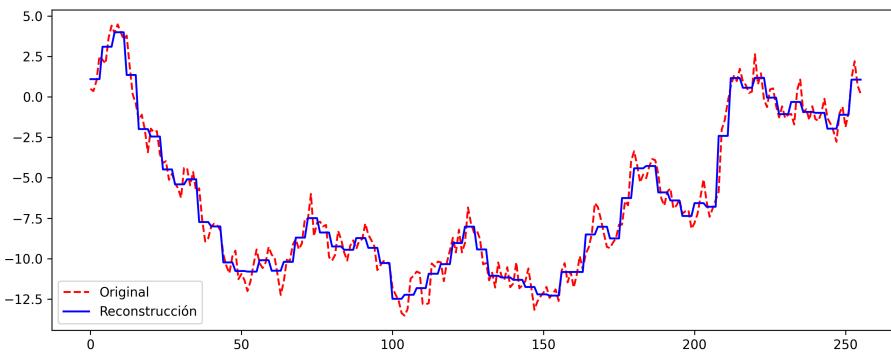
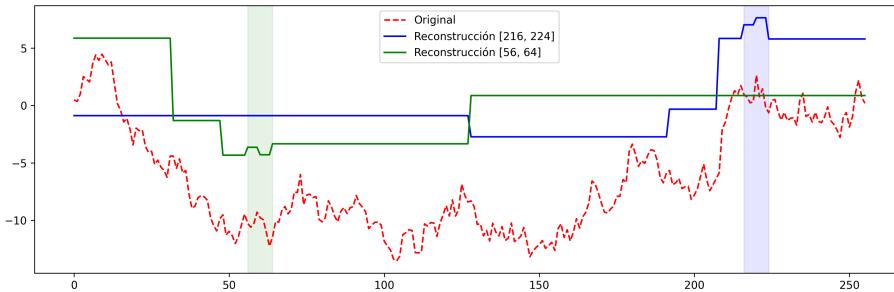


Figura 5.3.: Jerarquización que ofrece la transformada de Haar, se puede ver una analogía con el esquema presentado en la [Figura 5.1](#). Conforme se aumenta de frecuencia se tiene más información temporal. El último nivel segmenta la serie en intervalos de igual longitud.

contienen la información más relevante, y al ser vectorizados e incorporados al modelo, se podría estudiar la estructura vectorial que inducen y cómo esta enriquece al propio modelo.



(a) Reconstrucción de una serie a partir de su transformada de Haar.



(b) Reconstrucción separada de dos segmentos.

Figura 5.4.: Análisis y reconstrucción por segmentos usando la transformada de Haar sobre una serie temporal. En (a) se muestra la reconstrucción usando todos los coeficientes de detalle salvo los dos primeros, creando una segmentación en la serie. En (b) se pueden observar las reconstrucciones independientes de dos segmentos específicos tomando solo los coeficientes de los intervalos diádicos que contienen a cada uno de los segmentos.

Parte II.

Clasificación de Series Temporales

6. Estado del arte

Una vez se ha explorado la idea de la transformada de Haar, la cual tendrá un papel fundamental a la hora de justificar el modelo, es hora de estudiar el problema a resolver: la clasificación de series temporales.

Este capítulo empieza con unas nociones para comprender la importancia y complejidad de dicho problema, posteriormente se realiza una revisión del estado del arte y se concluye el capítulo introduciendo los conceptos esenciales sobre aprendizaje automático para el desarrollo de la propuesta.

6.1. Clasificación de series temporales

Las series temporales son secuencias de datos ordenadas en el tiempo, donde cada observación representa una magnitud o conjunto de magnitudes medidas en instantes consecutivos. Este tipo de datos aparece de forma natural en numerosos contextos como la monitorización de procesos industriales, el registro de señales biomédicas, las finanzas o la climatología. A diferencia de otras formas de datos, las series temporales presentan una fuerte dependencia temporal: el orden de los valores es crucial y puede reflejar dinámicas, tendencias, estacionalidades o patrones repetitivos que permiten inferir información relevante sobre los datos observados.

El problema de la **clasificación de series temporales** consiste en asignar una etiqueta a una secuencia completa de datos. A diferencia de los problemas clásicos de clasificación sobre datos tabulares o imágenes, donde cada entrada es un vector fijo de características, aquí las entradas son secuencias que pueden tener longitud variable o no, y que pueden estar compuestas por una o varias dimensiones. El objetivo será investigar si es posible construir un modelo fundacional para clasificación de series temporales, partiendo de una estructura inicialmente diseñada para modelos de lenguaje.

La clasificación de series temporales tiene una gran relevancia práctica en diversos ámbitos. Por ejemplo, en el ámbito médico, la clasificación de series temporales permite detectar patologías a partir de señales fisiológicas como electrocardiogramas, electroencefalogramas o registros de glucosa en sangre, aportando diagnósticos cada vez más tempranos y con mayor fiabilidad. En el sector industrial, el análisis de series temporales provenientes de sensores de vibración, presión o temperatura puede utilizarse para mejorar la fase de mantenimiento, anticipando fallos y reduciendo tiempos de inactividad en procesos críticos. En el sector financiero, la identificación automática del tipo de comportamiento de una serie de precios permite modelar el riesgo, detectar anomalías o aplicar estrategias de inversión automatizadas. De igual modo, en el sector energético, la clasificación de perfiles de consumo eléctrico facilita la segmentación de usuarios, la predicción de picos de demanda y el diseño de tarifas dinámicas. Estos son solo algunos de los campos más relevantes y con mayor uso de estas series pero, como se ha comentado, su origen puede ser de naturalezas muy variadas y con aplicaciones muy diversas.

6. Estado del arte

En todos estos escenarios, la correcta clasificación de series temporales es esencial para la toma de decisiones. Sin embargo, esta tarea presenta numerosos desafíos: las señales temporales suelen contener ruido, pueden presentar diferentes escalas, tener longitudes variables, estructuras internas complejas, entre otros inconvenientes. Además, los conjuntos de datos reales frecuentemente presentan clases desbalanceadas y estructuras no estacionarias, lo que complica la extracción de patrones relevantes.

Ante esta complejidad, el desarrollo de modelos robustos, precisos y escalables se ha convertido en una línea activa de investigación dentro del campo del aprendizaje automático. En particular, existe un creciente interés en el diseño de arquitecturas que no requieran una fuerte ingeniería manual de características y que puedan aprender directamente a partir de los datos en crudo. Los modelos basados en la arquitectura *Transformer* han demostrado un rendimiento sobresaliente en otras tareas secuenciales como el procesamiento del lenguaje natural, gracias a su mecanismo de atención que permite capturar dependencias a largo plazo de forma eficiente. Sin embargo, su aplicación directa a series temporales numéricas presenta desafíos importantes, como la necesidad de redefinir la noción de *token* y adaptar los mecanismos de codificación posicional y de entrada.

Este proyecto se enmarca dentro de esta línea de investigación, proponiendo y evaluando una arquitectura adaptada específicamente a la clasificación de series temporales. En punto de partida será el diseño de una etapa previa de transformación del dato, **Hitsbe**, donde la serie es fragmentada y transformada en una serie de vectores embebidos para ser procesada por un modelo *Transformer*. Posteriormente, se crea un modelo completo, **HitsBERT**, con este módulo conectado a un transformer tipo *BERT* [DCLT19]. Finalmente se realiza una etapa de preentrenamiento seguida de una etapa de extracción de características. El objetivo es averiguar si este enfoque para caracterizar series temporales permite abrir una nueva vía de investigación.

6.2. Principales métodos

En 2017 se realizó una amplia comparativa [BLB⁺17] de algoritmos de clasificación de series temporales sobre 85 conjuntos de datos univariados del archivo de la Universidad de California Riverside (UCR). En dicho estudio se evaluaron 11 clasificadores pertenecientes a cinco grandes enfoques algorítmicos: distancia, intervalos, *shapelets*, diccionario e híbridos.

En una evaluación más reciente [MSB24] se utilizaron 112 datasets del repositorio UCR, ampliado progresivamente en los últimos años, y se propuso una clasificación más extensa con ocho categorías. A las cinco originales se añadieron los algoritmos basados en convoluciones, en características y en aprendizaje profundo.

A continuación se presenta una breve introducción a cada una de estas categorías. El objetivo es mostrar la amplia variedad de algoritmos propuestos para el problema de la clasificación de series temporales, reseñando las peculiaridades más interesantes de algunos de ellos.

Algoritmos basados en distancias

Los métodos más clásicos en clasificación de series temporales se basan en comparar series completas mediante funciones de distancia. La más conocida dentro de este grupo es *Dynamic Time Warping*, DTW, una distancia elástica capaz de alinear secuencias desfasadas. Combinada con clasificadores como 1-NN, ha sido durante años el estándar de referencia.

Dentro de las distancias elásticas, podemos encontrar una amplia gama como LCSS, ERP, MSM. En esto se apoya **Elastic Ensemble** [OL19], que se basa en una combinación de 11 distancias elásticas junto con clasificadores 1-NN, consiguiendo mejorar el rendimiento de 1-NN DTW. Sin embargo, la mejora importante llegó con **Proximity Forest** [LSP⁺19]. Este algoritmo construye un conjunto de árboles de decisión utilizando las mismas distancias que Elastic Ensemble, estas se seleccionan aleatoriamente en los distintos nodos y se van expandiendo recursivamente hasta que se obtienen nodos puros.

Estos algoritmos se caracterizan por ser intuitivamente simples con la desventaja general de ser costosos en eficiencia al estar calculando continuamente distancias a lo largo de distintas fases temporales.

Algoritmos basados en características

Estos algoritmos transforman cada serie en un vector de características descriptivas, sobre el cual se aplican clasificadores tradicionales. Dado que su objetivo es condensar la información de toda la secuencia, se consideran métodos de transformación *serie-a-vector*, y suelen seguir un esquema sencillo de extracción y selección de características.

Uno de los enfoques más conocidos es **TSFresh** [CBNKL18], que genera alrededor de 800 características por serie. Aunque estas pueden utilizarse directamente, el propio método incorpora una etapa de selección denominada *FRESH*, basada en tests de hipótesis como el de *Fisher* o el de *Kolmogorov-Smirnov*, y que controla el número de falsos positivos mediante el procedimiento de *Benjamini-Yekutieli*. Las características resultantes pueden emplearse con clasificadores tradicionales como Random Forest.

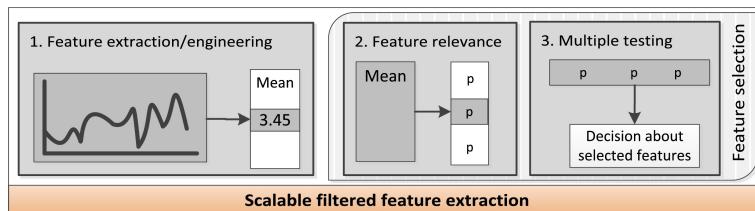


Figura 6.1.: Arquitectura del modelo TSFresh. Pueden observarse los tres pasos que realiza el algoritmo: extracción de características (1.), cálculo de los p-valores (2.) y aplicación de los tests (3.). Imagen extraída de [CBNKL18]

Sin embargo, los mejores resultados se obtienen al utilizar todas las características extraídas por *TSFresh* sin aplicar *FRESH*, y combinarlas con un clasificador *Rotation Forest*. Esta combinación, conocida como **FreshPRINCE** [MB22], ha demostrado ser uno de los enfoques más competitivos dentro de esta categoría.

Algoritmos basados en intervalos

Los algoritmos de esta categoría dividen las series temporales en intervalos de longitud fija o variable, y extraen estadísticas sencillas de cada uno. A diferencia de los métodos basados en características globales, estos se centran en fragmentos específicos, lo cual permite mitigar el impacto del ruido y captar patrones locales. El enfoque más representativo es **Time**

6. Estado del arte

Series Forest [DRTV13], que genera múltiples árboles donde cada uno trabaja con intervalos aleatorios resumidos con estadísticas básicas como la media o la pendiente.

A partir de este enfoque se desarrollaron variantes como **STSF** y **RSTSF** [CNQK21], que incorporan representaciones adicionales (como el periodograma o las diferencias) y aplican una selección supervisada de intervalos basada en la capacidad de separación de clases. **Canonical Interval Forest** [MLB20] mejora la diversidad mediante una selección aleatoria más amplia de intervalos y estadísticas, y su versión extendida, **DrCIF** [MLF⁺21], añade múltiples representaciones de la serie para generar descriptores aún más variados.

El avance más reciente en esta línea es **QUANT** [DSW24], que utiliza cuantiles dentro de intervalos diádicos sobre cuatro representaciones (original, diferencias de primer y segundo orden, y coeficientes de Fourier), generando hasta 480 características clasificadas mediante un modelo *Extra Trees*. En conjunto, los métodos basados en intervalos han evolucionado hacia representaciones más ricas y diversas, combinando estadísticas simples con estructuras de ensamble eficientes, y logrando así una mejora significativa en precisión y robustez.

Algoritmos basados en *shapelets*

Los algoritmos basados en *shapelets* buscan identificar pequeños fragmentos característicos de serie que permiten distinguir entre clases. Estos patrones no dependen de la fase, por lo que pueden aparecer en cualquier punto temporal y seguir siendo informativos. El enfoque más representativo es **Shapelet Transform Classifier** [HLB⁺14], que transforma cada serie en un vector de distancias a los *shapelets* más discriminativos. Estas distancias se utilizan luego como entrada para un clasificador. Inicialmente, se utilizó *HESCA*, un ensamblaje heterogéneo de clasificadores con ponderación basada en rendimiento. En versiones más recientes, se ha optado por un único clasificador como *Rotation Forest*, y la selección de *shapelets* se realiza de forma aleatoria.

Una línea alternativa es la de **MrSEQL** [LNGI⁺19] y su extensión **MrSQM**, [NI23] que discretizan subseries en palabras mediante transformaciones simbólicas como SAX, y aplican modelos lineales como regresión logística para identificar los patrones más relevantes. **MrSQM** mejora esta idea con una selección más eficiente de subseries y un control más efectivo de la redundancia.

Más recientemente, **Random Dilated Shapelet Transform (RDST)** [GVE22] ha introducido la extracción masiva de *shapelets* aleatorios, calculando no solo la distancia, sino también su posición y frecuencia. Mediante el uso de dilataciones, *RDST* captura patrones en distintas escalas temporales, aproximándose conceptualmente a los enfoques basados en convoluciones.

Algoritmos basados en diccionario

Los métodos basados en diccionario transforman fragmentos de serie en palabras discretas que luego se resumen en un histograma, ignorando la posición exacta de los patrones. En lugar de medir distancias como los *shapelets*, estos enfoques discretizan subseries mediante transformaciones simbólicas. El algoritmo más representativo es **BOSS** [Sch15], que aplica la transformación *Symbolic Fourier Approximation (SFA)*: cada subserie se normaliza, se transforma con la Transformada Rápida de Fourier y se discretiza en símbolos, generando vectores dispersos que reflejan la frecuencia de aparición de cada palabra.

WEASEL 1.0 [SL17] extiende esta idea incluyendo múltiples longitudes de ventana, bigramas y selección de características mediante el test *chi-cuadrado*, entrenando un clasificador *Ridge* sobre el espacio resultante. Su evolución, **WEASEL 2.0** [SL23], introduce dilataciones y aplica hasta 150 transformaciones *SFA* con parámetros aleatorios. Aunque no realiza selección explícita de atributos relevantes, combina diccionarios muy ricos (entre 30k y 70k dimensiones), capturando una gran variedad de patrones. En conjunto, los métodos basados en diccionario ofrecen una alternativa robusta y escalable, aprovechando la frecuencia de patrones discretizados para representar estructuras repetitivas sin necesidad de alineamientos.

Algoritmos basados en convoluciones

Este enfoque aplica convoluciones con múltiples *kernels* para generar representaciones de la serie temporal, capturando patrones locales mediante mapas de activación que se resumen en estadísticas. El algoritmo más representativo es **ROCKET** [DSW20], que utiliza miles de *kernels* aleatorios y aplica dos operaciones de *pooling* (máximo y proporción de valores positivos) para construir vectores de características, entrenando clasificadores lineales como *Ridge* o Regresión Logística.

MiniROCKET [DSW21] mejora la eficiencia al fijar parámetros clave, manteniendo la precisión y multiplicando su velocidad. Su extensión, **MultiROCKET** [TDB⁺22], incorpora nuevas operaciones de resumen y deriva de la serie, generando hasta 50 mil características. Finalmente, **Hydra** [DSW23] introduce una estrategia basada en diccionario, agrupando *kernels* y contando cuántos fueron los más activados en su grupo. Su combinación con *MultiROCKET*, conocida como **MultiROCKET-Hydra** [DSW23], representa el estado del arte en esta familia de métodos, equilibrando precisión y eficiencia.

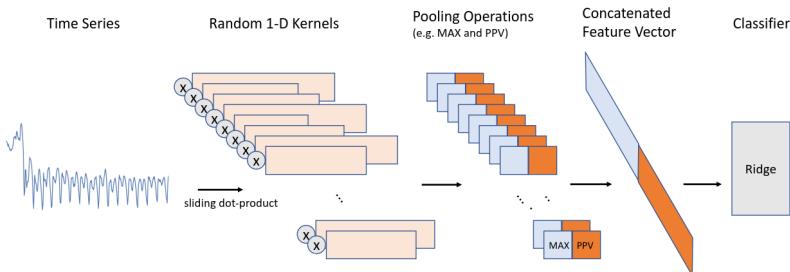


Figura 6.2.: Esquema de convoluciones en el que se basan ROCKET, MiMiniROCKET y MultiROCKET. Imagen extraída de [MSB24]

Algoritmos basados en aprendizaje profundo

El aprendizaje profundo ha sido una de las áreas más activas en clasificación de series temporales, aunque su impacto ha sido más limitado que en otras disciplinas como la visión por computador. Los resultados más consistentes se concentran en unos pocos modelos destacados, entre ellos **ResNet**¹ [WYO17], uno de los primeros en mostrar buenos resultados,

¹ResNet está diseñado para imágenes. Sin embargo, en este caso se trata de una adaptación para clasificación de series temporales.

6. Estado del arte

y especialmente **InceptionTime** [IFLF⁺20], una red basada en módulos convolucionales de distinto tamaño, con capas residuales y promedio global para estabilizar el entrenamiento.

Sobre *InceptionTime* se han desarrollado variantes como **H-InceptionTime** [IFDWF22], que incorpora filtros manuales para detectar patrones específicos como picos o tendencias, logrando mejoras sostenidas en precisión. Más recientemente, **LITETime** [IFDB⁺23] propone una versión mucho más eficiente mediante convoluciones separables y técnicas de dilatación y multiplexado, manteniendo un rendimiento comparable al modelo original.

Algoritmos híbridos

Los algoritmos híbridos combinan diferentes representaciones de series temporales en un solo modelo, lo que permite aprovechar las fortalezas de múltiples enfoques. Estos métodos no se limitan a una sola transformación, sino que integran varias, como distancias elásticas, *shapelets*, diccionarios o intervalos, en un ensamble más amplio, logrando una mayor robustez y precisión en la clasificación.

El primer modelo destacado fue **Flat-COTE** [BLHB15], un ensamble plano de 35 clasificadores que cubren representaciones en el dominio temporal, la autocorrelación, el espectro de potencia y los *shapelets*. Cada clasificador se entrena por separado y su voto se pondera según su precisión en validación cruzada.

La evolución continuó con **HIVE-COTE v1.0** [BFL⁺20] que organiza los clasificadores en módulos jerárquicos según el tipo de representación, introduce mejoras de escalabilidad mediante componentes más eficientes como **cBOSS** [MVB19]. La versión más reciente, **HIVE-COTE v2.0** [MLF⁺21], reemplaza los módulos por versiones actualizadas, permite clasificar series multivariantes y sustituye la validación cruzada por una estimación del error más eficiente mediante *out-of-bag*.

6.3. Aprendizaje automático.

Para finalizar este capítulo se introducen algunos conceptos esenciales del aprendizaje automático. En particular se repasan brevemente las redes neuronales y se realiza una introducción al modelo *Transformer*. El objetivo de este capítulo es comprender las ideas que fundamentan el modelo sobre el que se desarrolla este trabajo, facilitando así la comprensión de la propuesta presentada.

Las nociones que se muestran a continuación han sido adquiridas durante el desarrollo del grado en las asignaturas *Aprendizaje Automático* [Sanb] y *Visión por Computador* [Sana], facilitando su presentación.

El aprendizaje automático es una rama de la inteligencia artificial que se centra en el estudio de los algoritmos que pueden extraer información de los datos aportados e inferir una hipótesis que permita generalizar sobre datos nuevos. Mientras que, en los algoritmos clásicos, se usan determinadas reglas sobre los datos para obtener respuestas, en el aprendizaje automático, se usan los datos disponibles para obtener reglas que nos permitan inferir sobre los datos futuros.

Dentro de esta disciplina se pueden encontrar diversos tipos de aprendizaje. Entre ellos, una de las clasificaciones más generales es el aprendizaje supervisado, aprendizaje por refuerzo y aprendizaje no supervisado:

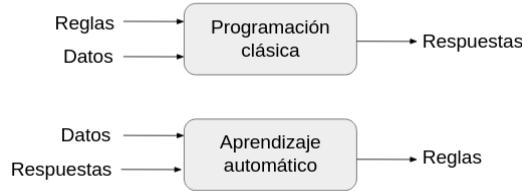


Figura 6.3.: Esquema de diferencias principales entre la programación clásica (arriba) y el aprendizaje automático (debajo). Imagen traducida de [Cho18].

- **Aprendizaje supervisado:** se basa en el uso de datos etiquetados, es decir, ejemplos en los que se conoce la salida esperada. Por lo general, es necesaria la intervención humana para realizar dicho etiquetado. Es el enfoque más utilizado en tareas de clasificación y regresión.
- **Aprendizaje por refuerzo:** en este caso no se proporcionan etiquetas directas, sino señales de recompensa que indican la calidad de las acciones tomadas por un agente en un entorno. El modelo debe aprender a tomar decisiones secuenciales que maximicen la recompensa acumulada a lo largo del tiempo. La intervención humana en este caso es menor, en vez de etiquetar todos los datos, basta con definir las recompensas y castigos.
- **Aprendizaje no supervisado:** se emplea cuando los datos no están etiquetados. El objetivo es descubrir patrones internos o agrupaciones en los datos. Este tipo de aprendizaje es especialmente útil cuando no se dispone de anotaciones manuales y, por lo tanto, no hay intervención humana o es mínima.

Este trabajo se centrará principalmente en el aprendizaje supervisado y aprendizaje por refuerzo. En el aprendizaje supervisado se parte de la búsqueda de una función objetivo f que se desea replicar. Los datos se generan a partir de dicha función, $y_i = f(x_i)$ y se plantea una hipótesis, \mathcal{H} , sobre el tipo de función buscada. A partir de esto, se puede desarrollar un algoritmo de aprendizaje. Tras el entrenamiento, se evalúa la hipótesis g para ver si se ha alcanzado un resultado satisfactorio o si es necesario realizar alguna modificación. Este proceso se refleja esquematizado en la Figura 6.4

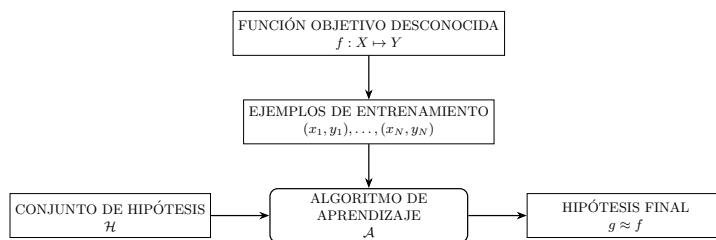


Figura 6.4.: Esquema básico sobre la búsqueda de hipótesis en un problema de aprendizaje supervisado. Imagen traducida de [AMMIL12]

Hasta este punto, no se ha detallado con precisión qué es exactamente lo que aprenden estos modelos. La respuesta es que los modelos aprenden un conjunto de **pesos**. Por ejemplo, en

6. Estado del arte

un problema de regresión lineal con N variables de entrada, el objetivo es encontrar una aproximación de la función objetivo mediante una combinación lineal de las variables:

$$\hat{y} = w_0 + w_1x_1 + w_2x_2 + \cdots + w_Nx_N,$$

donde los parámetros w_0, w_1, \dots, w_N representan los pesos del modelo, los cuales se ajustan durante el entrenamiento para minimizar el error de predicción.

Para evaluar qué tan buena es una predicción, se utiliza una **función de pérdida o función de error**, que compara las salidas generadas por el modelo con los valores reales. El objetivo del proceso de aprendizaje es minimizar dicha función, ajustando tales pesos.

En el caso de la regresión lineal, una función de pérdida común es el **error cuadrático medio**, que puede minimizarse de forma analítica mediante la pseudoinversa de la matriz de entrada, o bien utilizando métodos iterativos como el descenso por gradiente, que se introducirá a continuación.

Cabe destacar que estos conceptos se aplican principalmente al aprendizaje supervisado y por refuerzo. En el aprendizaje no supervisado, como en algoritmos de agrupamiento (*K-medias*, *DBSCAN*), no siempre hay pesos explícitos que ajustar, sino que el aprendizaje se basa en otras estructuras.

Los problemas en aprendizaje automático suelen clasificarse en dos grandes categorías: **regresión y clasificación**. En regresión, el objetivo es predecir un valor continuo a partir de las entradas, mientras que en clasificación se busca asignar una etiqueta a cada ejemplo, es decir, predecir una clase.

Para obtener dicho resultado, se parte de una función de pérdida cuya entrada son los pesos del modelo, $W = (w_1, w_2, \dots, w_N)$, y los datos a evaluar. Esta función depende del tipo de problema. En regresión, como se mencionó, se suele usar el error cuadrático medio:

$$\mathcal{L}(W) = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2.$$

En clasificación binaria, una función común es la **entropía cruzada** o *cross-entropy*:

$$\mathcal{L}(W) = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)].$$

Una vez establecida la función de pérdida, el siguiente paso es minimizarla. En algunos casos simples, es posible obtener una solución analítica que proporcione los valores óptimos de los parámetros. Sin embargo, el algoritmo por excelencia y el que establece la base para otros métodos diseñados para minimizar funciones de pérdida en modelos complejos, es el **descenso del gradiente**.

Este método se basa en el uso del gradiente de la función de pérdida, un vector que indica la dirección de mayor aumento en el valor de dicha función. Como el objetivo es minimizar la pérdida, el algoritmo ajusta los pesos en la dirección opuesta al gradiente.

Supóngase que en la iteración k del algoritmo, los pesos actuales del modelo son $W_k = (w_{k1}, w_{k2}, \dots, w_{kN})$. Primero, se calcula el gradiente de la función de pérdida evaluado en dichos pesos:

$$\nabla \mathcal{L}(W_k) = \left(\frac{\partial \mathcal{L}}{\partial w_1}(W_k), \frac{\partial \mathcal{L}}{\partial w_2}(W_k), \dots, \frac{\partial \mathcal{L}}{\partial w_N}(W_k) \right).$$

A continuación, se actualizan los pesos en la dirección contraria al gradiente,

$$W_{k+1} = W_k - \nabla \mathcal{L}(W_k).$$

Como puede observarse, el gradiente establece el "paso.^a realizar. Para poder modificar la intensidad en las iteraciones, se añade un parámetro $\eta > 0$ que permite variar el módulo del gradiente, lo que daría lugar a la expresión

$$W_{k+1} = W_k - \eta \cdot \nabla \mathcal{L}(W_k)$$

El factor $\eta > 0$ es conocido como **tasa de aprendizaje** y supone uno de los parámetros más relevantes a la hora de optimizar un modelo de aprendizaje automático. Este procedimiento se repite hasta que se cumpla algún criterio de parada, como alcanzar un número máximo de iteraciones o que la diferencia de la pérdida entre iteraciones consecutivas sea despreciable.

A partir de este algoritmo han surgido diversas modificaciones que mejoran el tiempo de convergencia o son capaces de encontrar mínimos más óptimos. Cabe destacar que las funciones de pérdida suelen ser funciones complejas en dimensiones elevadas donde el objetivo no es encontrar un mínimo absoluto, lo que se pretende es encontrar un mínimo óptimo. Uno de ellos es *Adam* [KB14], que ajusta dinámicamente la tasa de aprendizaje basándose en técnicas de optimización.

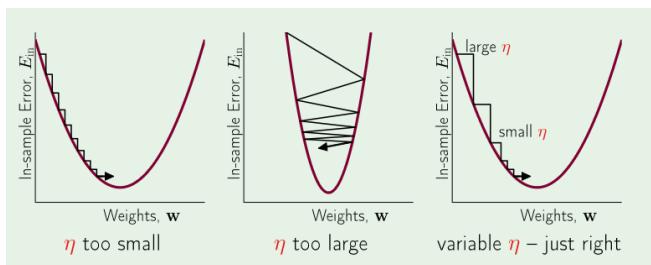


Figura 6.5.: En esta imagen puede apreciarse la importancia de elegir una correcta tasa de aprendizaje. A la izquierda puede apreciarse como una tasa de aprendizaje demasiado baja puede hacer que se alcance el mínimo deseado a un ritmo ineficiente. En la imagen del medio se observa como una tasa demasiado elevada puede hacer que el modelo no llegue siquiera a converger. En la imagen de la derecha puede observarse como una tasa de aprendizaje variable puede optimizar la búsqueda de dicho mínimo. Imagen extraída de [AM09].

Con estos conceptos básicos repasados, se presentan a continuación y de manera muy breve las redes neuronales. Este modelo ha sido y sigue siendo una herramienta central en el aprendizaje automático. El motivo de realizar este repaso se debe a que es una parte fundamental dentro de los modelos que se están persiguiendo, el *Transformer*.

6. Estado del arte

6.3.1. Redes neuronales

Las redes neuronales son uno de los modelos más usados en el ámbito de la inteligencia artificial actualmente. Su estructura y funcionamiento están inspirados en los del cerebro animal. Su unidad más básica es el **perceptrón**, inspirado en la neurona, y una combinación de perceptrones agrupados en diversas capas dan una gran capacidad a estos modelos para adaptarse a patrones no lineales.

El perceptrón es un modelo matemático que toma N valores de entrada $X = (x_1, x_2, \dots, x_N)$, los cuales se multiplican por un conjunto de pesos. El resultado se pasa por una función de activación que devuelve la salida que toma el modelo,

$$y = f \left(\sum_{i=1}^N w_i x_i + b \right)$$

El parámetro b se conoce como **sesgo**, y permite al modelo desplazar la función de activación, proporcionando mayor flexibilidad para ajustar los datos y mejorar la capacidad de aprendizaje del modelo.

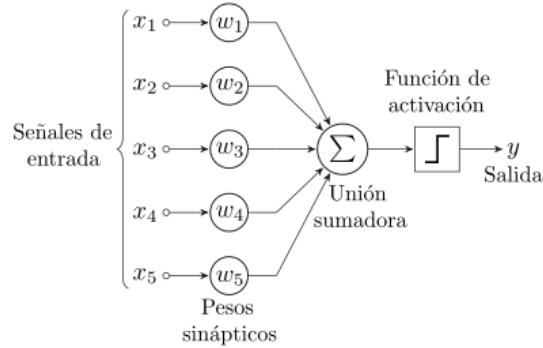


Figura 6.6.: Esquema de un perceptrón con 5 entradas. Imagen extraída de [Dan21].

Estos perceptrones se combinan en diversas capas dando lugar al **perceptrón multicapa**, el tipo de red neuronal más básico. Este modelo está compuesto por una capa de entrada, una o más capas ocultas y una capa de salida, donde cada neurona de una capa se conecta con todas las de la siguiente. Gracias a la concatenación de funciones de activación no lineales, el perceptrón multicapa puede aprender relaciones altamente complejas y resolver problemas no separables linealmente.

Hay muchos tipos de funciones de activación, como pueden ser la función de *Heaviside*, la identidad, la logística o la función rectificadora (ReLU), entre otras. Cada función modifica distintos aspectos del comportamiento del modelo. Por ejemplo, permiten controlar el grado de saturación de la señal. Estas propiedades pueden ser clave para que el modelo aproxime funciones complejas, o facilitar el aprendizaje en distintas capas de la red, entre otras cosas.

El aprendizaje de los pesos en una red neuronal se realiza mediante **retropropagación del error**, el algoritmo fundamental detrás de estos modelos. Este procedimiento es clave, pues al tratarse de capas de perceptrones conectadas entre ellas, los pesos de una capa dependen directamente de todos los anteriores, estableciendo una fuerte relación que es necesaria controlar a la hora de actualizar los pesos. Para ello, usa el descenso del gradiente sobre los

pesos con la peculiaridad de que la actualización de estos en las primeras capas dependen de la variación de los pesos en las capas posteriores. Gracias a la regla de la cadena, es posible propagar estos errores hacia atrás y calcular con precisión cómo debe ajustarse cada peso en cada capa.

El descenso del gradiente es la base de los algoritmos de entrenamiento en redes neuronales, aunque en la práctica, como se ha mencionado previamente, se utilizan variantes más sofisticadas como el descenso del gradiente estocástico, *Adam* o *RMSprop*, que mejoran la velocidad de convergencia y la estabilidad del entrenamiento.

Para finalizar la sección, cabe destacar que, a partir del perceptrón multicapa pueden derivarse arquitecturas especializadas, las cuales amplían el uso de estos modelos a distintos tipos de datos. Por ejemplo, si se incorporan una o varias capas con convoluciones previas, que aplican filtros con pesos sobre los datos de entrada, se obtiene una red neuronal convolucional, ampliamente utilizada en tareas como procesamiento de imágenes o visión por computador. Por otro lado, si se introduce retroalimentación entre las neuronas, permitiendo que la salida de una neurona en un paso aporte información a los pasos posteriores, se forma una red neuronal recurrente. Permitiendo al modelo capturar dependencias temporales, creando así una breve memoria que se va actualizando en cada paso. Como es lógico, estas redes son especialmente útiles para trabajar con datos secuenciales.

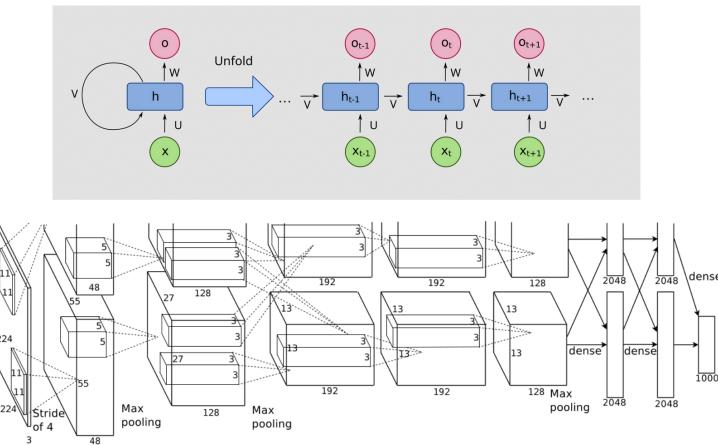


Figura 6.7.: En la imagen superior puede observarse la idea fundamental tras una red recurrente, con la retroalimentación en la parte izquierda y el despliegue de la misma en la parte derecha. En la imagen inferior puede observarse la arquitectura de una red convolucional entrenada en *ImageNet*, donde las imágenes pasan previamente tras unas capas convolucionales. Imágenes superior e inferior extraídas respectivamente de [fde17, KSH12]

En el ámbito de las series temporales, las redes neuronales recurrentes presentan limitaciones a la hora de modelar dependencias a largo plazo, al procesar las secuencias elemento a elemento, es difícil retener información lejana al elemento actual. Por otro lado, aunque las redes convolucionales sí pueden ofrecer cierto paralelismo, su capacidad para modelar dependencias globales está restringida por los tamaños de los núcleos a usar. En este contexto es donde el *Transformer* puede jugar un papel fundamental introduciendo el mecanismo de

6. Estado del arte

atención. Este permite procesar secuencias de datos en paralelo y capturar dependencias a largo plazo.

6.3.2. El modelo *Transformer*

Propuesto en 2017 por Google en el artículo *Attention is All You Need* [VSP⁺17], el **Transformer** supuso una revolución en el campo del aprendizaje automático, especialmente en tareas de procesamiento del lenguaje natural. Su propósito original era traducir frases de un lenguaje a otro.

6.3.2.1. Arquitectura general

El modelo original está compuesto por dos grandes bloques: el **codificador** (*encoder*) y el **decodificador** (*decoder*). En este caso, el trabajo se va a centrar únicamente en el codificador, pues con este bloque es suficiente para intentar clasificar series temporales.

En el codificador, cada capa contiene dos subcomponentes: una capa de autoatención y una red neuronal *feedforward*. Por otro lado, en el decodificador, se añade además un mecanismo de atención cruzada que permite a este acceder a la salida del codificador. Todos los subcomponentes tienen de conexiones residuales y normalización de capas entre ellos para estabilizar el entrenamiento.

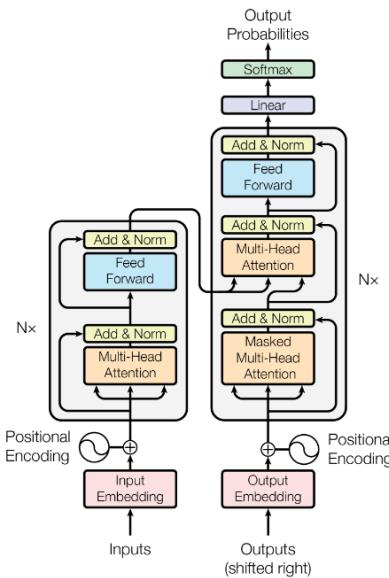


Figura 6.8.: Arquitectura general del modelo *Transformer*. A la izquierda puede encontrarse el codificador mientras que en la parte derecha de la figura se encuentra el decodificador. Como puede observarse no hay grandes diferencias entre ambas partes. Imagen extraída de [VSP⁺17].

6.3.2.2. Embeddings

Antes de comenzar a explicar cómo funciona este modelo, cabe destacar que los *Transformers* contienen una gran cantidad de parametrización, por ejemplo, las distintas dimensiones en las que se trabaja, y algunas explicaciones pueden variar de un modelo a otro. Esta explicación se basa en el primer modelo presentado en 2017, sin embargo, en la actualidad existe una gran cantidad de adaptaciones de este. Con esto aclarado, se procede a desarrollar su funcionamiento.

La entrada del modelo original son frases, las cuales se separan en bloques llamados *tokens*, que representan elementos básicos gramaticales. Para una mejor comprensión de esta sección, se puede imaginar que cada *token* corresponde a una palabra.

Antes de ser procesados por el modelo, estos *tokens* se representan mediante vectores de alta dimensión **embebidos**, también llamados *embeddings*. Es decir, se convierte información gramatical en información numérica. Para realizar esta transformación, se utiliza una matriz de pesos formada por tantas filas como *tokens* haya en el vocabulario y tantas columnas como dimensión tenga el modelo. De esta forma, cada *token* tiene asociada una fila de esta matriz, y la transformación se realiza simplemente mediante indexación.

Representar palabras como vectores puede resultar muy interesante. Imagínese que se tienen las palabras "*coche*", "*conductor*", "*piloto*" y "*avión*" vectorizadas. Si la representación en el espacio vectorial es adecuada, podrían cumplirse ciertas relaciones semánticas como:

$$\text{coche} - \text{conductor} \approx \text{avión} - \text{piloto}$$

Esto implicaría que:

$$\text{piloto} + (\text{coche} - \text{conductor}) \approx \text{avión}$$

Este tipo de relaciones indica que el modelo ha captado que "*conductor*" se relaciona con "*coche*" del mismo modo que "*piloto*" con "*avión*". Este razonamiento se expresa frecuentemente como: "*Conductor* es a *coche* lo que *piloto* es a..." .

Tras realizar esta vectorización de la frase de entrada, lo que se obtiene es una secuencia de vectores. Sin embargo, estos vectores no contienen información sobre la posición que ocupan en la secuencia, y esta información es crucial para el procesamiento del lenguaje, por lo que el modelo la estaría perdiendo. Para mitigar esta pérdida y añadir información sobre el orden de los elementos, se emplea una **codificación posicional**, que se suma a los *embeddings* en el espacio vectorial.

Esta codificación posicional consiste en crear, para cada posición, un vector que aporte al modelo la información necesaria sobre la ubicación de cada *token* y sumarlo al primer vector. El tipo de codificación posicional original es la **sinusoidal** o absoluta. Una característica clave de esta codificación es que los vectores de posición están definidos antes del entrenamiento del modelo y son fijos. Cada vector se define según las siguientes fórmulas:

$$\begin{aligned} PE_{(pos,2i)} &= \sin\left(\frac{pos}{10000^{2i/d}}\right), \\ PE_{(pos,2i+1)} &= \cos\left(\frac{pos}{10000^{2i/d}}\right). \end{aligned}$$

6. Estado del arte

Estas funciones sinusoidales con frecuencias crecientes permiten que el modelo pueda aprender a distinguir la posición relativa de los *tokens* y sus combinaciones. Alternativamente, es posible emplear otros tipos de codificación, como las codificaciones aprendidas por pesos o las codificaciones relativas, estas últimas especialmente útiles para capturar relaciones basadas en distancias entre posiciones, en lugar de posiciones absolutas.

Una vez que los vectores se han sumado, para cada *token* original se tiene un vector embebido que lo representa en el espacio vectorial del modelo. El siguiente paso es introducir estos vectores en un bloque de atención.

6.3.2.3. El mecanismo de atención

Obsérvese que los vectores iniciales solo guardan información sobre el *token* al que están representando. No contienen ningún tipo de información sobre el contexto, es decir, sobre el resto de *tokens*. Supóngase que esta cadena es

$$\vec{E}_1, \vec{E}_2, \dots, \vec{E}_N.$$

Ahora se quiere buscar la información que cada *token* puede aportar sobre el resto. Para ello, primero cada vector es transformado en un vector de **consulta** o *query*. Este nuevo vector es de dimensión menor al original y se obtiene multiplicando el *token* correspondiente por una matriz, W_Q , cuyas entradas son pesos entrenables del modelo. De esta forma, para cada $i = 1, \dots, N$, se tiene que $\vec{Q}_i = W_Q \vec{E}_i$ y la sucesión original queda

$$\vec{Q}_1, \vec{Q}_2, \dots, \vec{Q}_N.$$

De manera análoga, la entrada se vuelve a transformar en el llamado vector **clave** o *key*. Se multiplica cada vector por una matriz W_K formada por pesos y con las mismas dimensiones que W_Q . De esta forma, se obtiene una segunda secuencia

$$\vec{K}_1, \vec{K}_2, \dots, \vec{K}_N.$$

Con estos dos conjuntos de vectores, Q y K , se realiza el producto escalar de cada vector consulta Q_i con cada uno de los vectores clave K_j , obteniendo así una secuencia de valores:

$$\langle \vec{Q}_i, \vec{K}_1 \rangle, \langle \vec{Q}_i, \vec{K}_2 \rangle, \dots, \langle \vec{Q}_i, \vec{K}_N \rangle$$

que se añade en la columna i de la **matriz de atención**, la cual se puede representar como $K^T \cdot Q$. Esta matriz tiene dicho nombre porque indica cuánto debe atender una posición a otra dentro de la secuencia. Recuérdese que el producto escalar, en cierta forma, mide la similitud entre dos vectores, luego un valor elevado en esta matriz implica que el vector clave ha aportado información relevante al vector consulta. Por lo tanto, con este método, el modelo logra captar, para un *token* específico, la característica definida por su consulta y que aportan el resto de *tokens* a través de las claves.

Posteriormente, estos valores son normalizados con la dimensión del espacio de consultas o de claves, $\sqrt{d_K}$. El principal objetivo de esta normalización, como suele ocurrir en el aprendizaje automático, es aumentar la estabilidad y mejorar la eficiencia del modelo. Estos nuevos valores ahora normalizados se introducen en una función softmax por columnas. De esta forma, en cada columna, se obtiene una distribución de las contribuciones de las claves. Hasta ahora, el procedimiento a seguir puede resumirse con la ecuación:

$$\text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right).$$

De nuevo, se toma la secuencia original de los *tokens* embebidos y se transforman ahora en vectores de **valores** o *values*. El procedimiento es similar, con la única diferencia de que, en esta ocasión, los vectores no se ven reducidos en dimensionalidad. Es decir, la matriz W_V mantiene la dimensión de los vectores. Con esta nueva secuencia de vectores,

$$\vec{V}_1, \vec{V}_2, \dots, \vec{V}_N,$$

se realiza una combinación lineal tomando cada columna de la matriz de atención como escalares. Es decir, si la columna i de la matriz de atención viene dada por

$$\text{softmax} \left(\frac{\langle \vec{Q}_i, \vec{K}_1 \rangle, \langle \vec{Q}_i, \vec{K}_2 \rangle, \dots, \langle \vec{Q}_i, \vec{K}_N \rangle}{\sqrt{d_K}} \right) = (w_{i1}, w_{i2}, \dots, w_{iN}),$$

se calcula un nuevo vector realizando la combinación lineal:

$$\Delta \vec{E}_i = \sum_{k=1}^N w_{ik} V_k.$$

De esta forma, se obtiene una nueva secuencia de vectores que se suma a la secuencia original, dando lugar a la salida del bloque de atención:

$$\vec{E}'_1 = \vec{E}_1 + \Delta \vec{E}_1, \quad \vec{E}'_2 = \vec{E}_2 + \Delta \vec{E}_2, \quad \dots \quad \vec{E}'_N = \vec{E}_N + \Delta \vec{E}_N.$$

Un detalle importante a destacar es que, en la construcción de la matriz de atención, antes de aplicar la función softmax, existe la posibilidad de enmascarar algunos de los valores obtenidos, con el objetivo de evitar que aporten información a consultas no deseadas. Por ejemplo, en el *Transformer* original se enmascaraba el triángulo superior de la matriz respecto a la diagonal con el objetivo de centrar la atención de un *token* únicamente en los *tokens* que le precedían.

Todos estos pasos establecen un bloque o **cabezal de atención**. Los *Transformers* están formados por varios cabezales que se ejecutan de forma paralela, dando lugar al **multicabezal de atención**. La salida de cada bloque se concatena con el resto y es proyectada en la dimensión original del modelo.

6.3.2.4. Capa feedforward

Tras el bloque de atención, cada capa incluye una red neuronal completamente conectada, que se aplica de forma independiente y en paralelo a cada vector de la secuencia. Esta red consta habitualmente de dos capas lineales con una función de activación no lineal intermedia.

Además, se incorporan dos elementos clave que contribuyen significativamente a la estabilidad y eficacia del entrenamiento: las **conexiones residuales** y la **normalización de capa**. Las conexiones residuales consisten en sumar la entrada original al resultado de cada subbloque, permitiendo un mejor flujo del gradiente durante el entrenamiento. Por otro lado, la normalización de capa permite estabilizar las activaciones y facilitar la convergencia del modelo.

6. Estado del arte

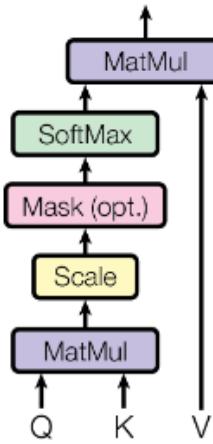


Figura 6.9.: Pasos que realiza un cabezal de atención. Imagen extraída de [VSP⁺17].

Uno de los principales inconvenientes del modelo *Transformer* es el coste computacional de la atención, que crece cuadráticamente con respecto a la longitud de la secuencia de entrada. Esto puede representar una limitación importante en aplicaciones donde las entradas son muy largas. Por este motivo, han surgido múltiples variantes del *Transformer* que buscan reducir esta complejidad, como Longformer [BPC20] o Linformer [WLK⁺20], entre otros.

Aunque fue diseñado originalmente para tareas de procesamiento del lenguaje natural, el modelo *Transformer* ha sido ampliamente adaptado a otros dominios:

- **Visión por computador:** los *Vision Transformers* reemplazan las convoluciones tradicionales por bloques de atención, y han demostrado rendimientos competitivos en clasificación de imágenes, segmentación y detección de objetos.
- **Series temporales:** los *Transformer* también se han aplicado con éxito a la predicción de series temporales, modelando dependencias temporales de forma no secuencial y permitiendo capturar patrones de largo alcance con mayor flexibilidad que los modelos tradicionales.

Gracias a su arquitectura flexible y potente, el *Transformer* se ha convertido en la base de los modelos de aprendizaje profundo más avanzados, como BERT [DCLT19], GPT [RNSS18], o LLaMA [TLI⁺23], y continúa evolucionando como una herramienta central en múltiples disciplinas del aprendizaje automático.

7. La propuesta

Tras aprender el funcionamiento de los modelos tipo *Transformers*, se puede apreciar a simple vista que adaptarlos al dominio de las series temporales no es tarea sencilla. En [WZZ⁺23] se puede encontrar un resumen de los principales *Transformers* adaptados a las series temporales. Llama la atención en este artículo la escasa cantidad de estos modelos que han sido adaptados para el problema de clasificación.

Algo común en estas alternativas es que los mayores cambios suelen hacerse en el mecanismo de atención, dejando a un lado el embebido de los datos. Estos modelos fueron diseñados expresamente para trabajar con el lenguaje natural y, a pesar de haber conseguido avances en otros campos, su diseño está pensado para explotar al máximo su potencial en dicho ámbito, algo que ha sido ampliamente demostrado en los últimos años. Por ello, antes de realizar grandes modificaciones en su arquitectura, es lógico pensar en buscar la manera de adaptar las series temporales a este tipo de modelos.

De esta idea surge **HitsBE** (*Haar Indexation for Time Series BERT Embeddings*), un módulo que aporta una representación vectorial en el espacio del modelo para series temporales, buscando similitud con el embebido de los *Transformers* para el lenguaje natural. Es decir, se busca, en cierta forma, parafrasear series temporales al idioma interno del modelo.

Como se ha visto en la sección de repaso, una frase se segmenta en *tokens*, pertenecientes a un vocabulario, y estos se embeben en un espacio vectorial mediante indexación en una matriz. Posteriormente se añade la información posicional y la secuencia de vectores se pasa por los mecanismos de atención. HitsBE funciona de forma análoga: se busca una representación de la serie con *tokens* de un “vocabulario” y se añade la información faltante mediante el uso de otros vectores.

Antes de empezar con la explicación del mecanismo, es necesario aclarar el concepto de vocabulario en este contexto.

7.1. Vocabulario

Un vocabulario se define como un conjunto de palabras pertenecientes a un idioma. En este caso, el idioma será el de las series temporales y las palabras serán pequeños fragmentos asociados a estas, de forma que, para una serie temporal, se busque una secuencia de pequeñas series temporales que pueda representarla en cierta medida.

Como el dominio sobre el que se define un conjunto de series temporales puede llegar a ser muy amplio, incluso infinito, se propone un vocabulario formado por palabras, es decir, pequeñas series temporales normalizadas en el intervalo $[0, 1]$. Aun así, dentro de este intervalo se pueden definir infinitas series temporales. El objetivo será encontrar una representación suficientemente buena para minimizar la pérdida de información. Además, en este trabajo, estas series serán de longitud fija con el objetivo de facilitar su extracción en la serie original y poder añadir información relevante que se pierde con esta representación.

7. La propuesta

En concreto, el tamaño elegido para las palabras es de $N = 8$, es decir, series temporales formadas por 8 evaluaciones en el intervalo $[0, 1]$. A partir de ahora, se mencionará “palabra” cuando se quiera hacer referencia a las pequeñas series del vocabulario, y se reservará “serie temporal” para referirse al objeto que se está estudiando.

Para la generación del vocabulario se optó por una combinación de estrategias. Por un lado, se definió un vocabulario base (`primal_vocab`) formado por funciones elementales con distintas formas y escalas: potencias de x , raíces, senos, cosenos y funciones cuadráticas, todas ellas evaluadas en 8 puntos distribuidos uniformemente en el dominio $[0, 1]$. Por otro lado, se amplió el vocabulario generando una gran cantidad de palabras aleatorias mediante una distribución uniforme en $[0, 1]^8$.

En un primer momento se agruparon todas las palabras en una lista, lo que suponía un coste computacional muy elevado. Para buscar qué palabra representa mejor a un segmento de la serie temporal que se está procesando, es necesario recorrer todo el conjunto del vocabulario. Para mitigar este problema, se introdujo el concepto de **indexación por monotonía**, un método para identificar y agrupar las palabras según el comportamiento de sus monotonías.

La idea es representar cada palabra mediante una cadena binaria que codifica la monotonía entre dos instantes de la palabra. Para ello, se asigna un 1 si el valor de la palabra en un instante determinado aumenta respecto al instante anterior, y un 0 en caso contrario. A continuación, la cadena binaria se convierte en el número entero correspondiente, y este indica qué entrada de la lista corresponde con la clase seleccionada. Esta codificación define 2^{N-1} clases de equivalencia sobre el vocabulario, y permite agruparlo en subconjuntos de palabras que comparten forma similar. De esta manera, al buscar la palabra que mejor representa un segmento dado de la serie, solo es necesario comparar con las palabras que comparten la misma estructura de monotonía. En este caso, como las palabras tienen longitud 8, se generan $2^7 = 128$ clases distintas.

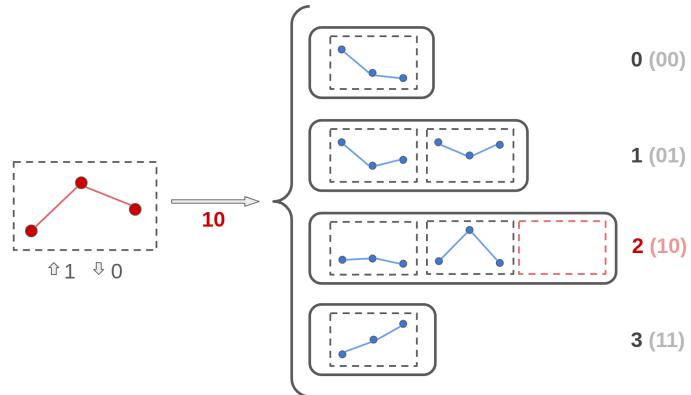


Figura 7.1.: Ejemplo del proceso de indexación de monotonía para un vocabulario con palabras de longitud $N = 3$. A la izquierda, una palabra que primero crece y luego decrece, asociándole el índice 10, que corresponde con la tercera entrada del vocabulario (índice 2 al pasar de binario a entero).

Este método de agrupación permite una mejora significativa de velocidad al recorrer la estructura del vocabulario. En un experimento realizado con un vocabulario de 50 000 palabras,

se observó que el tiempo necesario para realizar las mismas 55 representaciones de series temporales se redujo de 35.60 segundos con una estructura lineal a 2.17 segundos con indexación por monotonía. Esto supone una mejora en rendimiento superior a 16 veces respecto al sistema original.

La indexación por clases de monotonía, además de acelerar el proceso de búsqueda, permite analizar propiedades estadísticas del vocabulario. En particular, este agrupamiento abre la posibilidad de estudiar la entropía de la monotonía desde un punto de vista probabilístico, abriendo otra línea para investigaciones futuras.

En un vocabulario completamente aleatorio, como el generado mediante una distribución uniforme en $[0, 1]^8$, las clases de monotonía tienden a concentrarse en aquellas con una proporción equilibrada de subidas y bajadas. Este fenómeno tiene una justificación probabilística directa.

Si se parte de un valor inicial $x_0 = k$ en el intervalo $[0, 1]$, el valor del siguiente punto x_1 determina el primer bit del índice de monotonía. En este contexto, existe una probabilidad k de que $x_1 < x_0$ (bit 0) y una probabilidad $1 - k$ de que $x_1 > x_0$ (bit 1). A medida que se encadenan subidas (bits 1), el valor actual tiende hacia el extremo superior del intervalo, disminuyendo la probabilidad de seguir subiendo. De forma análoga, tras varias bajadas (bits 0), aumenta la probabilidad de que el siguiente valor sea mayor que el actual.

Este comportamiento estocástico hace que, en promedio, las secuencias de monotonía alternen subidas y bajadas, lo que genera una mayor concentración de palabras en clases con patrones equilibrados. Este efecto se refleja claramente en la distribución de frecuencias representada en la [Figura 7.2](#). Puede observarse que los picos más altos corresponden a las clases más equilibradas, y las zonas más bajas, a clases más extremas, casi siempre crecientes o decrecientes.

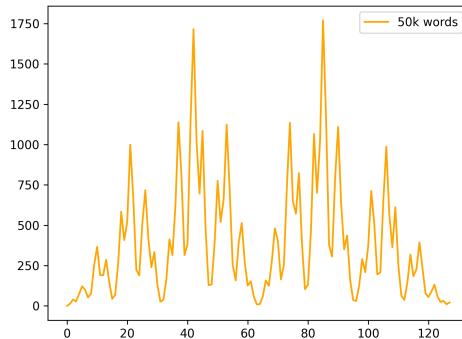


Figura 7.2.: Distribución para un vocabulario de 50.000 palabras. Puede observarse la agrupación que se genera, destacando la precisión simétrica.

Para mitigar esta desproporción, para un vocabulario de 50.000 palabras se han seleccionado las clases con menos de 100 elementos y se han añadido las palabras inversas de las clases de monotonía opuestas. De esta forma se diversifica el vocabulario en los conjuntos más débiles.

Respecto a la cantidad de palabras a usar, se consideró que tal cantidad era adecuada tras probar vocabularios mayores y observar que, las clases de monotonía más pobladas crecían

7. La propuesta

aumentando la redundancia considerablemente mientras que las clases menos probables apenas crecían en el número de palabras o no llegaban a hacerlo directamente. Puede verse un ejemplo visual de esta desproporción en Figura 7.3.

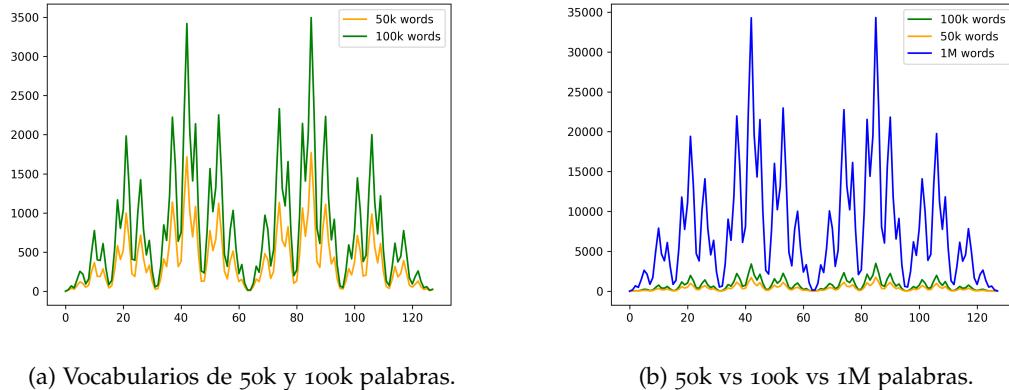


Figura 7.3.: Comparación entre vocabularios uniformes de diferentes tamaños. La imagen (a) muestra que al pasar de un vocabulario de 50k a otro de 100k palabras, las principales diferencias se concentran en las clases ya bien representadas en el vocabulario más pequeño. En (b), al incorporar un vocabulario de 1M de palabras, este efecto se intensifica, observándose una mayor densidad en las clases más frecuentes.

Una vez se ha aclarado el concepto de vocabulario para el ámbito de las series temporales. El siguiente paso es explicar como funciona el modelo propuesto para realizar la representación.

7.2. Hitsbe

Como se ha mencionado al comienzo de este capítulo, la idea de este módulo no es adaptar los *Transformers* para analizar series temporales, sino adaptar la representación de las series temporales para que puedan ser utilizadas por modelos de dicho tipo.

De forma análoga al funcionamiento del embebido en el *Transformer* original, el procedimiento consiste en segmentar la serie temporal y representar cada uno de estos segmentos mediante una palabra o *token* del vocabulario. Una vez se tiene dicha representación, se obtiene el vector correspondiente a cada *token* mediante la matriz de embebidos que, al igual que en el modelo original, devuelve los vectores por indexación.

Estos vectores representan, en cierta medida, las variaciones que sufre la serie dentro de cada segmento y, parcialmente, la intensidad de estas. Sin embargo, como se ha visto en la sección anterior, las palabras están normalizadas en el intervalo $[0, 1]$ por motivos de viabilidad. Esto provoca una pérdida importante de información respecto a la magnitud de la serie. Es decir, con esta representación de palabras no se puede saber, por ejemplo, en qué intervalo está contenida la imagen de la serie o si hay segmentos que muestran cambios de tendencia más intensos que otros. Por este motivo, se introduce la **indexación de Haar**. En resumidas cuentas, esta añade, en forma de vector, información sobre la magnitud contenida

en un segmento en el espectro de frecuencias y tiempo. Esta información es aportada por la transformada de Haar, la cual fue presentada en el [Capítulo 5](#), y cuya eficiencia fue demostrada a lo largo de la primera parte de este trabajo.

Para finalizar, al igual que en el modelo original, es necesario aportar codificación posicional para proporcionar al modelo información sobre el orden de la secuencia. Todo esto se explica con detalle en los siguientes apartados de esta sección.

7.2.1. Tokenización

Como se ha comentado, el primer paso del algoritmo consiste en segmentar la serie temporal en fragmentos de longitud fija. En este caso, se consideran series temporales de longitud $T = 1024$, por lo que la serie se divide en $\tau = T/N = 128$ segmentos disjuntos, siendo N el tamaño de las palabras del vocabulario. Es decir, cada segmento tiene una longitud de 8.

El siguiente paso es tomar cada segmento y normalizarlo en el intervalo $[0, 1]$, en este caso mediante una normalización *min-max*. Esta aporta estabilidad a la hora de realizar la comparación, ya que las palabras del vocabulario están definidas en el intervalo $[0, 1]^N$, mientras que los segmentos originales pueden tomar valores arbitrarios en \mathbb{R}^8 . A continuación, se busca la palabra que mejor representa al segmento. Esta búsqueda se realiza evaluando una distancia de similitud.

El vocabulario, como se ha indicado en el apartado anterior, se organiza según clases de monotonía. Por ello, se calcula el índice de monotonía del segmento que se pretende representar y se busca la mejor aproximación dentro de la clase correspondiente. Dentro de esa clase, el segmento se compara con todas las palabras disponibles, seleccionando la más cercana en términos de la distancia establecida. En este caso, se ha optado por la distancia basada en la norma $\|\cdot\|_1$, es decir, la suma de las diferencias en valor absoluto, también conocida como distancia *Manhattan*.

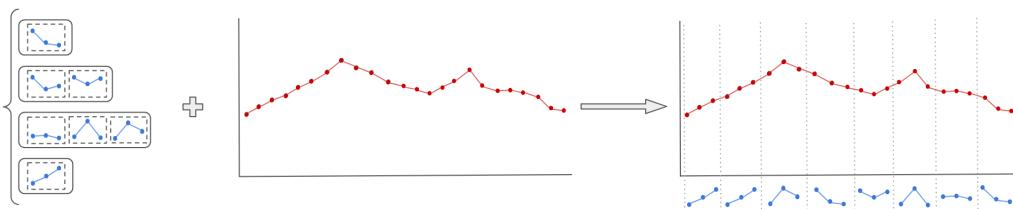


Figura 7.4.: Ejemplo del proceso de *tokenización* para una serie temporal de tamaño $T = 24$ usando un vocabulario de palabras de longitud $N = 3$. La serie por lo tanto, se divide en $\tau = 8$ segmentos. En cada segmento se añade la palabra del vocabulario que mejor lo representa.

Este enfoque permite mantener un vocabulario acotado y estructurado, que puede capturar diferencias sutiles en la forma local de la serie observando los cambios de monotonía. La elección de la distancia $d_{\|\cdot\|_1}$ se debe a que el objetivo es identificar la palabra que mejor se adapte en forma al segmento. En este contexto, no se desea que una coincidencia puntual en un valor específico tenga un peso desproporcionado en la medida de similitud, como ocurriría con la distancia $d_{\|\cdot\|_2}$, donde las diferencias se elevan al cuadrado. La distancia $d_{\|\cdot\|_1}$ proporciona mayor estabilidad en este sentido. Por otro lado, la norma $d_{\|\cdot\|_\infty}$ proporciona

7. La propuesta

una buena acotación de la palabra, pero pierde información puntual que sí aporta la norma elegida.

Una vez seleccionadas las palabras, cada una se traduce al vector correspondiente mediante la matriz de *embeddings*. El procedimiento consiste, virtualmente, en aplanar todo el vocabulario y tomar la posición que ocupa la palabra elegida. Esa posición será el índice que le corresponde en la matriz. Para ello, al cargar el vocabulario en HitsBE se guarda un vector auxiliar donde cada entrada almacena la suma acumulada de elementos de las clases previas.

7.2.2. Indexación de Haar

Como ya se ha mencionado, la *tokenización* introduce una nueva limitación: se pierde la información sobre la magnitud original del segmento. Como consecuencia, por ejemplo, dos segmentos con la misma forma pero escalas distintas serán representados mediante la misma palabra, ignorando diferencias de magnitud que pueden ser relevantes en el contexto.

Para compensar esta pérdida, se incorpora información adicional mediante la transformada de Haar, la cual, como se vio en el [Capítulo 5](#), permite descomponer la señal en coeficientes asociados a diferentes escalas de frecuencia y tiempo. Esta descomposición revela la estructura jerárquica de la serie, separando patrones de baja frecuencia o tendencias globales, de componentes de alta frecuencia o variaciones locales.

La implementación realiza la transformada de Haar de cada serie completa hasta una profundidad máxima, con el objetivo de capturar aquella información que los *tokens* no son capaces de representar. Para cada segmento se extrae, en cada nivel de descomposición, el coeficiente de detalle que corresponde a la función base de Haar cuyo intervalo diádico contiene dicho segmento. De esta forma, se obtiene para cada segmento la información sobre su magnitud en la transformada.

El algoritmo procede de la siguiente manera: se consideran los coeficientes de detalle de la transformada de Haar de la serie, denotados por c_{ab} , donde $a \in 1, \dots, M$ representa el nivel de frecuencia, $b \in 1, \dots, 2^{a-1}$ el desplazamiento temporal dentro de ese nivel, y M el número de niveles de la transformada. Es decir,

$$c_{11}, \quad c_{21}, c_{22}, \quad c_{31}, c_{32}, c_{33}, c_{34}, \quad \dots \quad c_{M1}, \dots, c_{M2^{M-1}}.$$

Estos coeficientes se organizan en una matriz de M filas y 2^{M-1} columnas, correspondiente al número de segmentos (máxima resolución). Esta matriz tiene la forma:

$$\begin{bmatrix} c_{M1} & c_{M2} & c_{M3} & c_{M4} & \cdots & c_{M2^{M-1}} \\ c_{(M-1)1} & c_{(M-1)1} & c_{(M-1)2} & c_{(M-1)2} & \cdots & c_{(M-1)2^{M-2}} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{21} & c_{21} & c_{21} & c_{21} & \cdots & c_{22} \\ c_{11} & c_{11} & c_{11} & c_{11} & \cdots & c_{11} \end{bmatrix}.$$

Cada columna representa un segmento de la serie y cada fila, un nivel de resolución de la transformada. Cada entrada de la matriz corresponde al coeficiente del intervalo diádico que contiene al segmento en la resolución correspondiente. Por ejemplo, para $M = 3$ la matriz quedaría:

$$\begin{bmatrix} c_{31} & c_{32} & c_{33} & c_{34} \\ c_{21} & c_{21} & c_{22} & c_{22} \\ c_{11} & c_{11} & c_{11} & c_{11} \end{bmatrix},$$

y los coeficientes de Haar del segundo segmento serían los elementos de la segunda columna: c_{32}, c_{21}, c_{11} , ordenados por frecuencia creciente.

Este vector se embebe en un subespacio del modelo mediante una matriz de pesos y se combina con el vector del *token* embebido y la codificación posicional. La Figura 7.5 ilustra gráficamente cómo se agrupan los segmentos según los niveles de la transformada de Haar, permitiendo capturar patrones en distintas escalas.

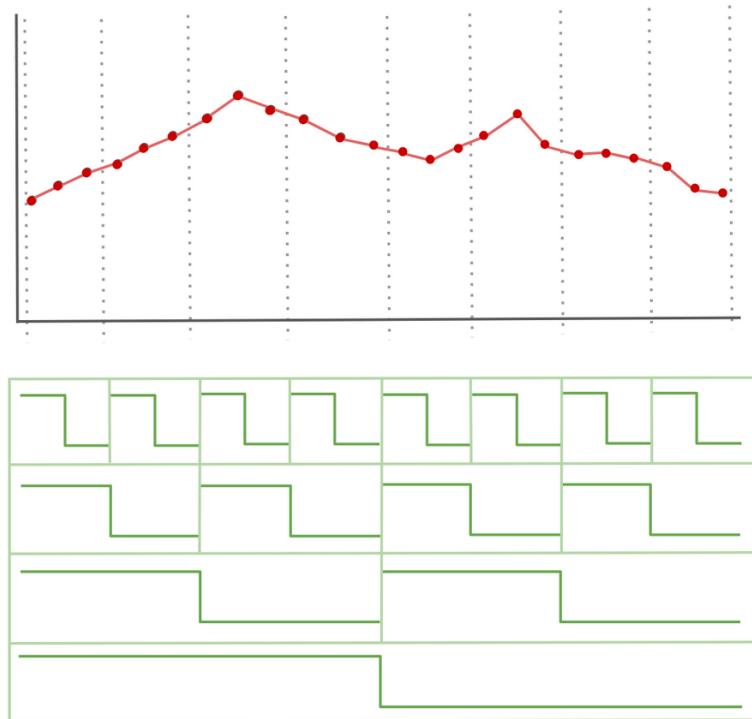


Figura 7.5.: Ejemplo ilustrativo del proceso de codificación mediante la transformada de Haar. La serie original es descompuesta en múltiples niveles, donde cada nivel representa una resolución distinta que captura patrones en diferentes escalas de tiempo.

7.2.3. Salida final

Con la secuencia de vectores de los *token* y la secuencia de vectores de la transformada de Haar, la única información que falta es la codificación posicional. Dado el alto nivel de personalización que implica diseñar un modelo como este desde cero, se ha optado por utilizar la codificación más simple, la misma que incorpora el *Transformer* original, es decir, la codificación posicional sinusoidal.

7. La propuesta

Los vectores se normalizan para evitar que alguno de ellos predomine sobre el resto, se suman y se devuelven como una secuencia de vectores de entrada a un *Transformer*, representando así la serie temporal.

En realidad, puede plantearse una analogía entre el lenguaje natural y la vectorización que se realiza con HitsBE. No se pretende justificar el método de forma exacta, sino explorar cómo un modelo *Transformer*, del mismo modo que aprende las reglas y estructuras que rigen un idioma, podría aprender las que dominan a determinados tipos de series temporales.

Una frase puede analizarse desde un punto de vista gramatical o sintáctico. Desde el grammatical, se pueden encontrar relaciones contextuales, por ejemplo, un determinante siempre debe concordar en género y número con el sustantivo al que acompaña, o, de forma más abstracta, la presencia de un determinante implica la cercanía de un sustantivo. No es descabellado pensar que existan series temporales donde la presencia de un patrón particular (representado por una palabra del vocabulario) implique la aparición de otros patrones con ciertas propiedades.

Desde el punto de vista sintáctico, una oración puede estar compuesta por varias proposiciones, cada una con sujeto y predicado, que a su vez poseen núcleos y sintagmas o complementos asociados. Así se forma una jerarquía en la que, a cada nivel y en cada sección de la oración, los elementos próximos aportan información relevante sobre el significado global. De forma análoga, la transformada de Haar permite capturar cómo se comportan distintas secciones de la serie cuando se observan en conjunto, revelando relaciones jerárquicas en distintas escalas de frecuencia y tiempo.

Esta analogía se ilustra en la [Figura 7.6](#). De la palabra “fuerte” puede extraerse, desde el punto de vista grammatical, que se trata de un adjetivo calificativo, y sintácticamente, que constituye el núcleo de un sintagma adjetival dentro del complemento directo del predicado de la segunda proposición de una oración yuxtapuesta. En el segmento resaltado en la parte inferior, se identifica la clase de monotonía (2 en este caso), la intensidad de los cambios internos y los coeficientes de Haar que corresponden a los intervalos diádicos que contienen dicho segmento, organizados jerárquicamente por frecuencia.

7.3. HitsBERT

Con HitsBE diseñado y desarrollado, el siguiente paso fue que cumpliera su propósito. Como su nombre indica, se ha creado para embeber series temporales en un modelo BERT. Este nombre BERT proviene de *Bidirectional Encoder Representations from Transformers*, y se basa en la idea de aprovechar la capacidad del mecanismo de atención del codificador para acceder simultáneamente a todo el contexto de la secuencia, tanto pasado como futuro.

Como se explicó en la introducción al modelo *Transformer*, durante el mecanismo de atención es posible aplicar enmascaramientos para limitar qué claves pueden ser atendidas por una consulta. Esta técnica se utiliza típicamente para impedir que un *token* acceda a información futura, lo cual es esencial en tareas de generación secuencial. Como el objetivo presente es la clasificación y no la predicción, tiene sentido basarse en un modelo con este tipo de modificación.

La diferencia de BERT es que explota esta atención bidireccional de forma explícita durante el preentrenamiento, mediante una tarea de predicción de *tokens* enmascarados (*Masked*

El piloto ha competido duro, se merece una fuerte ovación.

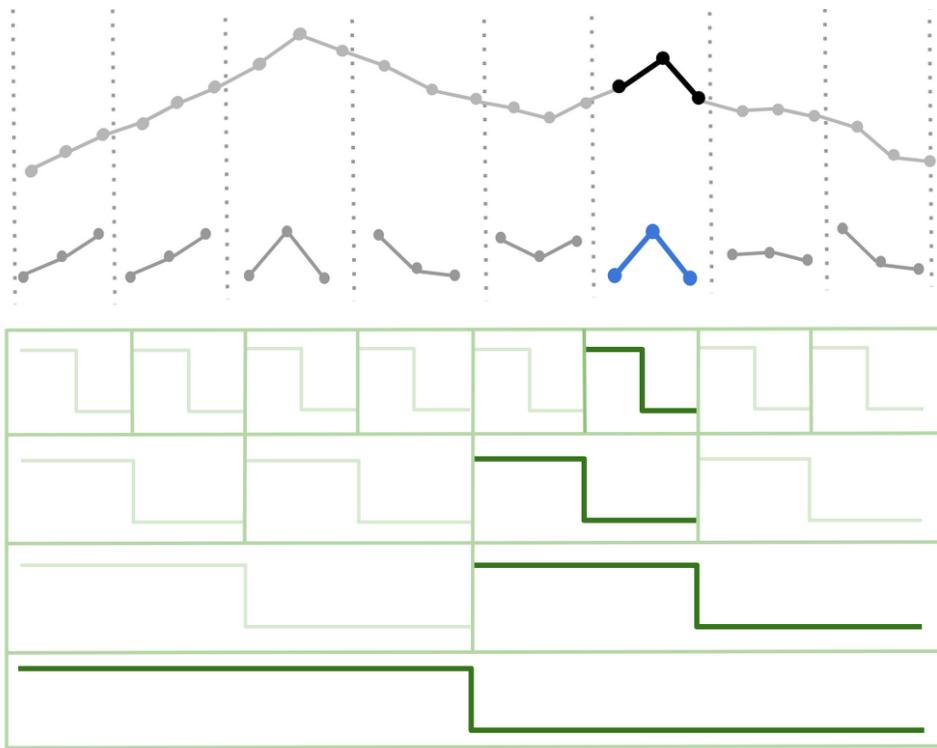
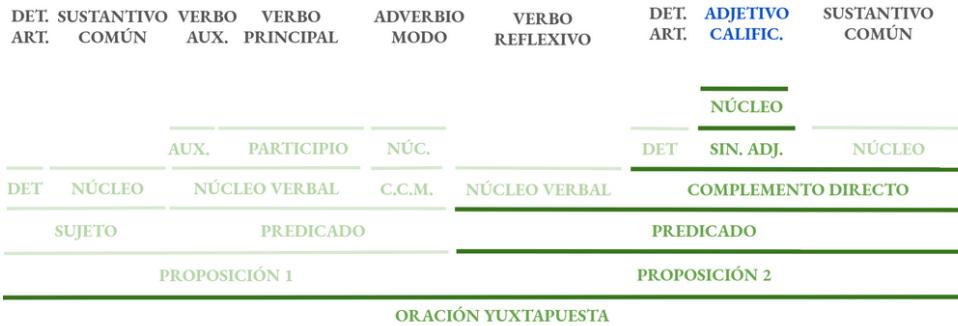


Figura 7.6.: En la parte superior de la imagen puede observarse un análisis gramatical (azul) y un análisis sintáctico (verde) sobre una palabra de una frase. Mientras que, en la parte inferior, puede observarse una representación de *token* (azul) y una extracción de los coeficientes de Haar que contienen al segmento (verde). Puede verse claramente la analogía que se plantea.

7. La propuesta

Language Modeling). Este enfoque permite al modelo capturar dependencias contextuales más ricas, lo que se ha traducido en mejoras significativas en tareas de comprensión del lenguaje. Por este motivo, y debido a que es un modelo de código abierto que permite una alta personalización, se ha optado por elegir este *Transformer* para este proyecto.

El modelo HitsBERT está compuesto por un módulo HitsBE y un modelo BERT concatenados. Los detalles más técnicos se discutirán más adelante; sin embargo, cabe resaltar que, cuando el modelo se utiliza para clasificación, se suele añadir un *token* especial llamado [CLS]. La salida correspondiente a este *token* es la que se emplea habitualmente para realizar la clasificación. Además, el modelo incluye los métodos necesarios para configurarse, así como para almacenar y guardar sus pesos.

A partir de este modelo, se ha diseñado otro más específico para el preentrenamiento. La principal diferencia radica en el método que procesa los datos a través del modelo, ya que este ha sido concebido específicamente para ejecutar el preentrenamiento desarrollado de forma genuina en este proyecto.

7.4. Detalles de implementación

La implementación del modelo se ha desarrollado íntegramente en **Python 3.10**. Para la gestión del entorno y las dependencias se ha utilizado el gestor de proyectos **Poetry**, que permite declarar versiones exactas y reproducir entornos de forma consistente. Además, se ha hecho uso de **Docker** para encapsular el entorno de ejecución, garantizando su portabilidad entre distintas máquinas. Esto incluye la gestión de versiones de Python, CUDA y bibliotecas específicas necesarias para el entrenamiento y evaluación del modelo.

Todo el trabajo desarrollado puede encontrarse en el siguiente repositorio <https://github.com/angelolmn/HitsBE>. A continuación, se describen los detalles técnicos más relevantes sobre el mismo.

Bibliotecas principales utilizadas

A continuación se detallan las principales bibliotecas utilizadas en la implementación del modelo, así como en los procesos de entrenamiento, visualización y evaluación:

- **PyTorch (v2.6.0)**: base de la implementación del modelo. Se utiliza para definir las capas de la arquitectura, realizar operaciones tensoriales, y ejecutar el entrenamiento y la inferencia.
- **Transformers (v4.48.3, Hugging Face)**: utilizada para heredar y modificar modelos tipo BERT, facilitando la integración de la arquitectura HitsBE.
- **NumPy (v1.21.0) y Pandas (v2.2.3)**: esenciales para el preprocesamiento de datos, generación de vocabularios y transformación de estructuras durante las distintas fases del proyecto.
- **PyWaveDecay (v1.8.0)**: empleada específicamente para calcular la transformada de Haar en el módulo HitsBE. Es el único punto donde se convierte temporalmente el tensor a `numpy.ndarray`, antes de devolver el resultado al entorno torch durante la ejecución del modelo.

- **Scikit-learn (v1.5.2, sklearn)**: se ha utilizado para aplicar algoritmos clásicos de clasificación y extraer métricas de rendimiento que sirvan de referencia en las comparaciones.
- **DeepSpeed (v0.16.7)**: usado para acelerar el entrenamiento a gran escala mediante técnicas como Zero Stage 1 y entrenamiento con precisión mixta. También permite una gestión más eficiente de la memoria y realizar puntos de guardado durante el entrenamiento fácilmente.
- **Aeon (v1.0.0)**: biblioteca especializada en el tratamiento de series temporales, útil para descargar, cargar y organizar eficientemente los datasets del UCR archive.
- **Matplotlib (v3.10.0)**: utilizada para generar figuras del artículo, visualizar patrones en las salidas del modelo y representar resultados cuantitativos de forma gráfica.
- **UMAP-learn (v0.5.7)**: herramienta clave para representar en dos dimensiones el espacio vectorial generado por HitsBERT, facilitando la interpretación de las representaciones embebidas.

Estructura del código fuente

El proyecto se ha organizado en dos carpetas, una para el desarrollo del modelo y otra para la realización de experimentos. La organización general puede encontrarse en la [Figura 7.7](#).

```

Proyecto/
    HitsBE/
        vocabulario.py
        hitsbe.py
        Modelos/
            hitsBERT.py
    Experimentos/
        Vocabulario/
        Datasets Preentrenamiento/
        Preentrenamiento/
        Datos/
        Comparaciones/

```

[Figura 7.7](#).: Estructura de carpetas del proyecto.

La carpeta más relevante es HitsBE/. En ella se pueden encontrar los siguientes ficheros:

- **vocabulario.py**: En este fichero se encuentra definida la clase `Vocabulary`. Los atributos son la longitud de la palabra y el número de palabras. Entre sus métodos destacan principalmente `_compute_index`, que toma como entrada una palabra y calcula su índice de monotonía; y `load_words` y `save_words`, que permiten cargar un vocabulario desde un fichero o guardararlo, respectivamente. Además, tiene implementados los métodos `__iter__`, `__len__` y `__getitem__`, que permiten iterar, obtener la longitud del vocabulario y acceder por índice de forma sencilla.

El constructor de la clase permite cargar un vocabulario desde un fichero directamente. En caso de que no se aporte ningún fichero, se define con `primal_vocab`.

7. La propuesta

- `hitsbe.py`: Este fichero contiene dos clases. Por un lado, `HitsbeConfig`, que permite guardar los parámetros relevantes de la clase principal en un archivo de configuración tipo JSON. Por otro lado, está la clase `Hitsbe`, que, como ya se ha visto, se encarga de vectorizar las series temporales como entrada para el Transformer. Esta última deriva de la clase `nn.Module`, esta es la que permite tratar la propuesta como un modelo de Torch, gestionando los pesos y facilitando la comunicación con el resto de herramientas de la misma biblioteca.

Esta clase contiene un vocabulario sobre el que se realiza la representación, y contiene las matrices de pesos `word_emb_matrix` y `haar_emb_matrix`, de tipo `Embedding` y `Parameter`, respectivamente. Además, las variables necesarias en memoria pero no entrenables, como la matriz de codificación posicional, se cargan mediante `register_buffer`.

Durante todo el proceso de representación, se realizan todas las operaciones posibles con Torch, mejorando el tiempo de ejecución. Únicamente para calcular la transformada de Haar, los tensores se convierten en arrays de NumPy para ejecutar la bibliotecas PyWaveDecay. Una opción a considerar sería implementar la transformada de Haar con PyTorch para mejorar aún más la eficiencia. El verdadero problema en este paso podría estar en la conversión de los datos de un tipo a otro, ya que implica moverlos entre dispositivos, lo cual suele ser costoso.

Por último, cabe destacar que este modelo admite series temporales de longitud máxima 1024, y siempre devuelve una máscara que indica qué segmentos de la serie están vacíos por tener esta una longitud menor. Siempre que esto ocurra, la clase centra la serie en la ventana de entrada.

- `Modelos/hitsBERT.py`: En este fichero pueden encontrarse dos clases, las cuales derivan de `nn.Module`. Por un lado, `HitsBERT`, que contiene un modelo HitsBE y un modelo BERT. Ambos pueden ser inicializados con sus respectivos archivos de configuración. Además, esta clase contiene el `token [CLS]`, el cual es necesario añadir entre la salida de HitsBE y la entrada de BERT.

El método `forward` de este modelo funciona de la siguiente manera: se pasa la entrada con dimensiones (`batch_size, ts_length`) por HitsBE; a la salida, de dimensiones (`batch_size, dim_seq, hidden_size`), se le concatena el `token [CLS]`, se expande la máscara para considerar el nuevo `token` y se pasa por BERT. Esta función devuelve la misma salida que el modelo BERT original.

Esta clase, además, contiene los métodos `save_pretrained` y `from_pretrained`, que, a partir de una ruta y un nombre, son capaces de guardar y cargar los pesos del modelo respectivamente. También gestiona directamente los ficheros de configuración, facilitando así el proceso.

La otra clase es `HitsBERTPretraining`, que contiene una instancia de `HitsBERT` que se pretende preentrenar. En líneas generales, puede usarse como cualquier otro modelo. Tiene un método `forward` que se explicará con detalle en la sección de preentrenamiento, y dos métodos para guardar y cargar los pesos, al igual que `HitsBERT`.

En la otra carpeta, `Experimentos`, se pueden encontrar subcarpetas que contienen ficheros creados y ejecutados en diferentes fases del proyecto, ya sea para comprobar el funcionamiento o para generar resultados para su análisis.

miento del modelo, generar imágenes, transformar datos, entrenar o evaluarlo, entre otras tareas.

- **Vocabulario/**: Contiene los ficheros usados para realizar experimentos de rendimiento y crear imágenes exploratorias sobre los vocabularios.
- **Datasets Preentrenamiento/**: Módulo de carga, normalización y preparación de los datasets (LongHorizon y Monash). Son diversos ficheros que permiten agrupar, barajar y transformar los datos usados durante el preentrenamiento. Más adelante se especificará con mayor detalle en qué consisten estos datos. A destacar en esta sección el fichero más relevante es `convert_data.py`, que contiene el proceso de enmascarado.
- **Preentrenamiento/**: En esta carpeta está todo lo necesario para realizar el preentrenamiento del modelo, desde los datos hasta el código de ejecución y los ficheros de configuración de *DeepSpeed*.
 - **Datos/**: Contiene los datos ya transformados para su uso durante el preentrenamiento.
 - **train.py**: Contiene el bucle principal de preentrenamiento. Se encarga de cargar el fichero de configuración, inicializar los modelos y parámetros, cargar los datos y entrenar por lotes. Además, guarda el modelo una vez finalizado.
 - **utils.py**: Contiene la función que entrena el modelo por lote. Se separa por modularidad y facilidad de depuración. Esta función se encarga de pasar los datos por el modelo, calcular el error y las métricas necesarias y guardar los resultados en un fichero.
 - **deepspeed_config.json**: Contiene los parámetros de entrenamiento configurables con DeepSpeed. En este caso, se ha decidido usar Zero Optimization y el optimizador AdamW con esquema para la tasa de aprendizaje WarmupDecayLR, lo cual se detallará en el apartado de preentrenamiento.
- **Datos/**: Contiene los datos del UCR archive en distintos formatos.
 - **Raw/**: Datasets en crudo, es decir, sin alterar.
 - **HitsBERTED Norm/**: Datasets procesados por modelos HitsBERT, preentrenado y sin preentrenar. Cada serie está representada por el token [CLS] de cada salida.
 - **HitsBERTED Mean/**: Datasets procesados por modelos HitsBERT, preentrenado y sin preentrenar. Cada serie está representada por el token [CLS] de cada salida.
 - **Ts Featured/**: Contiene los datasets expresados como vectores de características. Para ello se ha usado `tsfeatures`, un robusto extractor de características para series temporales.

Configuración del modelo

Para garantizar la reproducibilidad de los experimentos, en esta sección se detallan los hiperparámetros utilizados en cada uno de los componentes del modelo, así como las relaciones de dependencia entre ellos. Estas relaciones son fundamentales, ya que permiten coordinar los tamaños de las estructuras internas de cada clase y asegurar su compatibilidad.

7. La propuesta

La Figura 7.8 muestra un esquema visual con las dependencias clave entre los tres principales módulos del sistema: el Vocabulario, el modelo HitsBE y el modelo BERT. Por ejemplo, el tamaño máximo del embedding posicional de BERT es igual a la longitud de secuencia de entrada más uno, debido a la inclusión del token especial [CLS]. Asimismo, el tamaño del vocabulario debe coincidir en todos los módulos para permitir una codificación y decodificación coherente de los datos.

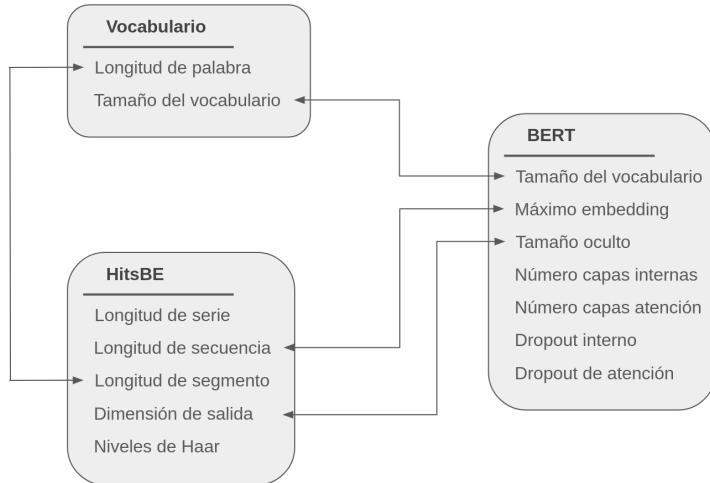


Figura 7.8.: Diagrama de dependencias entre los parámetros de cada clase. Destacar que máximo *embedding* de BERT es la Longitud de secuencia más uno debido al token [CLS].

A continuación, se resumen los valores seleccionados para cada módulo:

Parámetro	Valor
Longitud de palabra	8
Tamaño del vocabulario	50.000

Tabla 7.1.: Hiperparámetros del módulo Vocabulario.

Parámetro	Valor
Longitud de serie temporal	1024
Longitud de segmento	8
Longitud de secuencia	128
Número de niveles de Haar	8
Dimensión de salida	768

Tabla 7.2.: Hiperparámetros del módulo HitsBE.

7.4. Detalles de implementación

Parámetro	Valor
Tamaño oculto	768
Máximo embedding posicional	129
Tamaño del vocabulario	50.000
Número de capas internas	12
Número de cabezas de atención	12
Dropout interno	0.1
Dropout de atención	0.1

Tabla 7.3.: Hiperparámetros del modelo BERT.

8. Evaluación experimental

En este proyecto, la experimentación se ha dividido en dos bloques: por un lado, el preentrenamiento del modelo y, por otro, la evaluación de la capacidad de este para extraer características sobre series temporales.

A lo largo de este capítulo se explicarán los distintos procedimientos experimentales que se han llevado a cabo. Se detallarán los objetivos de cada fase, la metodología seguida, así como los resultados obtenidos y su interpretación en el contexto del modelo propuesto.

8.1. Preentrenamiento del modelo

El preentrenamiento de modelos tipo *Transformer* constituye una etapa fundamental en el desarrollo de estos. Su utilidad radica en la capacidad de estas arquitecturas para capturar estructuras internas de los datos sin requerir supervisión directa, lo que permite dotar al modelo de un conocimiento previo antes de abordar tareas específicas.

Se comenzará esta sección describiendo en qué consiste el preentrenamiento sobre el que se ha basado este modelo. Posteriormente, se presentará la propuesta de preentrenamiento adaptada a HitsBERT, explicando su diseño y los detalles técnicos de esta. Se finalizará mostrando los resultados y conclusiones obtenidas.

8.1.1. Enmascaramiento clásico

Uno de los enfoques más extendidos para el preentrenamiento es el *Masked Language Modeling* (MLM), introducido en el modelo BERT. Este procedimiento consiste en enmascarar aleatoriamente algunas posiciones de la secuencia de entrada y entrenar el modelo para que recupere los valores originales en dichas posiciones utilizando el contexto que no ha sido enmascarado.

Dada una frase de entrada, antes de embeberla en el modelo se seleccionan ciertas posiciones que se sustituyen por un token especial de máscara. A continuación, el modelo recibe esta entrada incompleta y debe predecir los elementos que faltan, basándose únicamente en la información disponible en las posiciones no enmascaradas.

Para ello, el *Transformer* genera una representación contextual para cada posición de la secuencia. En el caso de las posiciones enmascaradas, dichas representaciones se utilizan para producir una estimación del valor original, mediante una proyección lineal sobre el vocabulario y una capa *softmax* posterior.

Durante el entrenamiento, solo se evalúa la capacidad del modelo para reconstruir los valores enmascarados, ignorando el resto de posiciones. Esto obliga al modelo a desarrollar representaciones internas que capturen relaciones contextuales entre las distintas partes de la secuencia, sin necesidad de supervisión explícita.

8. Evaluación experimental

Este mecanismo de preentrenamiento ha demostrado ser altamente eficaz en diversas tareas, ya que permite al modelo adquirir una comprensión general del tipo de datos con los que va a trabajar antes de ser ajustado a una tarea específica.

8.1.2. Esquema propuesto

Siguiendo el mismo objetivo, se pretende que el modelo aprenda las estructuras internas propias de las series temporales. Como el modelo propuesto se basa en BERT, resulta lógico buscar una adaptación del esquema MLM para este nuevo tipo de dato. En concreto, se desea que el modelo sea capaz de diferenciar series que han sido alteradas y no presentan el comportamiento que realmente deberían.

En el esquema MLM original, la salida del modelo correspondiente a las posiciones enmascaradas se proyecta sobre el vocabulario embebido y, a dicha proyección, se le aplica una capa *softmax* para obtener una distribución de probabilidad y elegir la opción que el modelo considere correcta. En nuestro caso, buscar la "palabra" exacta en un vocabulario de series temporales no resulta del todo eficiente. Por un lado, no suele haber una única solución válida: una serie no sigue siempre un patrón específico y puede presentar ruido y pequeñas variaciones debidas a su alto componente estocástico. Así, en un segmento concreto, una palabra embebida podría ser tan válida como otra similar. Por otro lado, uno de los aportes más importantes es la información que la transformada de Haar aporta al modelo sobre el contexto de cada palabra y, al proyectar sobre el vocabulario embebido, esta información se perdería, limitando la representación que el modelo podría aportar.

Por ello, se ha optado por añadir una ligera supervisión al aprendizaje. Para cada serie temporal se generan dos copias de la misma y se selecciona aleatoriamente un intervalo diádico en una resolución determinada. En una de las copias, dicho intervalo se modifica siguiendo una de las siguientes reglas:

- Sustituir por palabras dentro de los mismos índices de monotonía.
- Sustituir por palabras con índices de monotonía inversos.
- Sustituir por otro intervalo diádico dentro del mismo nivel de resolución.
- Aplicar una transformación lineal al intervalo.

En la otra copia, las palabras dentro del segmento se cambian por otras elegidas de manera completamente aleatoria. De esta forma, para cada serie temporal se obtienen dos modificaciones de la serie en un intervalo diádico común. Las tres series se emplean como opciones para la salida del modelo, de modo que, en lugar de proyectar sobre todo el vocabulario, se proyecta únicamente sobre dichas opciones embebidas con la intención de predecir cuál representa la serie original. Es importante mencionar que el proceso de modificación de la serie se realiza antes del preentrenamiento, con el fin de evitar operaciones innecesarias durante la ejecución de este.

El resto del procedimiento es análogo: la serie se enmascara con un vector embebido que representa la máscara en este caso, se pasa por el modelo y la salida se compara con las opciones posibles. Se calcula el error y se actualizan los valores de los pesos del modelo. En la [Figura 8.1](#) puede verse un esquema de los pasos seguidos. Todo este procedimiento se lleva a cabo en la función forward de la clase HitsBERT.

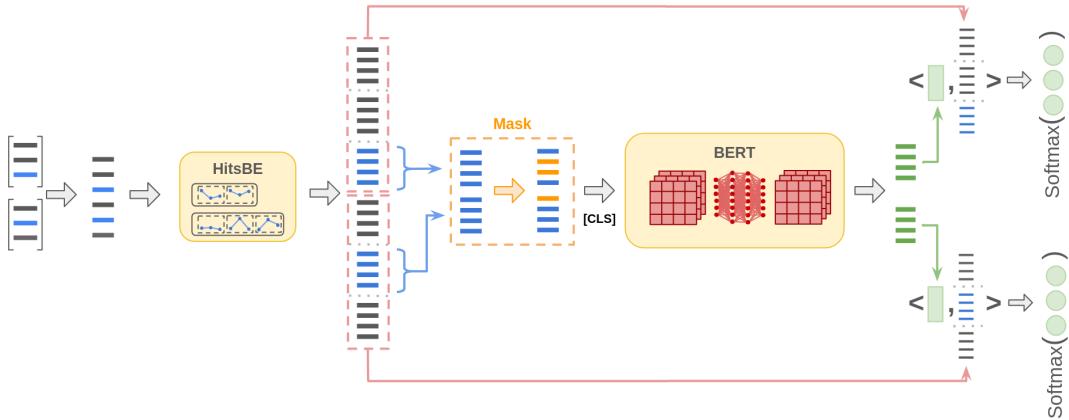


Figura 8.1.: Proceso que sigue el modelo para evaluar un conjunto de datos durante el preentrenamiento. De izquierda a derecha, las opciones a evaluar se concatenan y se embenen con HitsBE, lo que devuelve una secuencia de vectores para cada serie. La serie que será la opción correcta (azul) se enmascara, se le añade el token [CLS] y se introduce en BERT. La salida se compara con las opciones originales con la esperanza de que el modelo logre realizar una buena caracterización de la serie. Esta comparación se efectúa mediante un producto escalar sobre cada opción y la aplicación de una capa softmax sobre las respuestas.

8.1.3. Procesamiento de datos

Para este preentrenamiento se requiere un gran volumen de series temporales procedentes de dominios diversos, de modo que el modelo pueda aprender cómo varían y se estructuran tales datos en distintos contextos. Para evitar que el modelo accediera durante esta fase a series que pudieran aparecer posteriormente en tareas de evaluación o clasificación, se seleccionaron exclusivamente conjuntos de datos diseñados para predicción. Concretamente, se utilizaron dos repositorios ampliamente reconocidos: *Monash* y *LongHorizon*.

El repositorio *Monash* reúne series temporales de naturaleza muy diversa. Contiene datos climáticos y meteorológicos, registros económicos y de movilidad, información financiera, tráfico web, mediciones de fenómenos naturales (actividad solar y caudales fluviales), así como series de carácter demográfico y sanitario.

Por su parte, el repositorio *LongHorizon* aporta principalmente datos energéticos, tanto de ámbito industrial como doméstico. Además, incorpora registros meteorológicos, información financiera relacionada con tipos de cambio entre divisas e indicadores sobre la incidencia de gripe.

Algunos de estos conjuntos contienen pocas series, pero de gran longitud. En tales casos, cada serie se dividió en fragmentos de longitud 1024 (la máxima que admite el modelo) y cada fragmento se trató como una serie independiente. Recuérdese que el objetivo del preentrenamiento es que el modelo aprenda a distinguir entre versiones modificadas de una misma serie, por lo que este enfoque segmentado resulta completamente adecuado. Tras aplicar este procedimiento, se obtuvieron un total de 446 043 series distintas. En la Tabla 8.1 se detallan los conjuntos de datos empleados y la cantidad de series extraídas en cada caso. Una vez que se dispone de las series, se toman una a una y se crean las tres opciones: dos

8. Evaluación experimental

Origen: Monash	Nº series
bitcoin	3 240
covid_deaths	11 970
nn5_daily	4 995
saugeenday	1 035
sunspot	3 240
tourism_monthly	16 470
traffic_weekly	38 790
us_births	315
vehicle_trips	14 805
weather_monash	99 629
Total	199 489

(a) Series procedentes del repositorio Monash

Origen: LongHorizon	Nº series
ECL_1_bound	99 629
ETTh1	1 050
ETTh2	1 050
ETTm1	29 475
ETTm2	29 475
Exchange	4 425
ILI	450
weather	81 000
Total	246 554

(b) Series procedentes del repositorio LongHorizon

Tabla 8.1.: Distribución de series temporales agrupadas por origen.

modificadas y la serie sin alterar. Estas se barajan y se guarda la opción correspondiente a la serie correcta. Además, se almacena una máscara que indica el intervalo diádico que ha sido modificado.

Este proceso se repite 15 veces para cada serie, seleccionando aleatoriamente un intervalo diádico de 8 palabras, dos intervalos de 4 palabras, cuatro intervalos de 2 palabras y ocho intervalos con una única palabra enmascarada. Los datos generados se guardan en tres ficheros: uno con todas las opciones de las series generadas, otro con los índices de las series correctas y un tercero con todas las máscaras.

8.1.4. Detalles técnicos

Preentrenar un modelo *Transformer* es un procedimiento muy costoso en términos de cómputo y memoria. Se requiere una gran cantidad de datos y de iteraciones para que el modelo aprenda el “lenguaje” subyacente. En [IBL21] se propuso una manera de preentrenar BERT con un “presupuesto académico”. Para ello se emplearon ocho Nvidia Titan-V, cada una con 12 GB de memoria. Con los parámetros adecuados obtuvieron, en un día, resultados comparables a los preentrenamientos de varios días del artículo original.

Entre las características más importantes de ese proceso destacan:

- Uso de secuencias de 128 tokens, algo que HitsBERT ya hace.
- Entrenar un modelo **LARGE** y después aplicar poda para optimizar el rendimiento. Es preferible un **LARGE** entrenado el mismo tiempo que un **BASE**.
- Empleo del optimizador *AdamW* con $\beta_1 = 0.9$, $\beta_2 = 0.98$ y *weight decay* de 0.01. El *dropout* estándar y el de atención se fijan en 0.1.
- Uso del software *DeepSpeed*¹ para optimizar el entrenamiento.
- Enmascarar antes de la ejecución y cargar los datos en RAM para evitar cuellos de botella en disco.

¹<https://www.deepspeed.ai/>

En este caso se han seguido todas estas pautas, excepto la de emplear un modelo **LARGE**, debido a la limitación de memoria en las GPUs. Cada intento de preentrenar HitsBERT con un BERT **LARGE** terminó interrumpiéndose por falta de memoria.

En cuanto al resto de opciones, la más interesante es el uso de *DeepSpeed*. Este software, desarrollado por Microsoft para facilitar el entrenamiento eficiente de modelos de lenguaje a gran escala, ofrece optimizaciones como *ZeRO Optimization*. En este trabajo para dichas optimización se eligió la *fase 1*, que reduce significativamente la memoria. Otra opción atractiva es el uso de FP16, que permite duplicar el rendimiento y reducir a la mitad el consumo de memoria al trabajar con la mitad de precisión. Sin embargo, no se pudo aplicar porque los gradientes alcanzaban valores fuera del rango representable, devolviendo *Nan* y provocando la interrupción del proceso.

Respecto a esquema de la tasa de aprendizaje se decidió usar el mismo que se propone en el artículo. Se probaron tasas de aprendizaje de 0.01, 0.02 y 0.1, obteniéndose resultados prácticamente idénticos. También se experimentó con distintos porcentajes de *warm-up* en el esquema de la tasa de aprendizaje, sin observar grandes diferencias. Respecto al *hardware* se utilizó una NVIDIA GeForce RTX 2080 Ti con 12 GB de memoria en cada ejecución. El modelo se entrenó durante una época, recorriendo los 15 emmascaramientos del conjunto de datos. Se preentrenaron dos versiones de HitsBERT: una con pesos aleatorios con la misma distribución que los de BERT original sin preentrenar, y otra con los pesos de BERT **BASE UNCASED**. De esta forma, se puede buscar respuesta a una interesante pregunta: ¿Puede un LLM preentrenado aportar información relevante en el ámbito de las series temporales?

8.1.5. Resultados

A continuación se resumen los resultados más relevantes obtenidos durante la fase de preentrenamiento de HitsBERT.

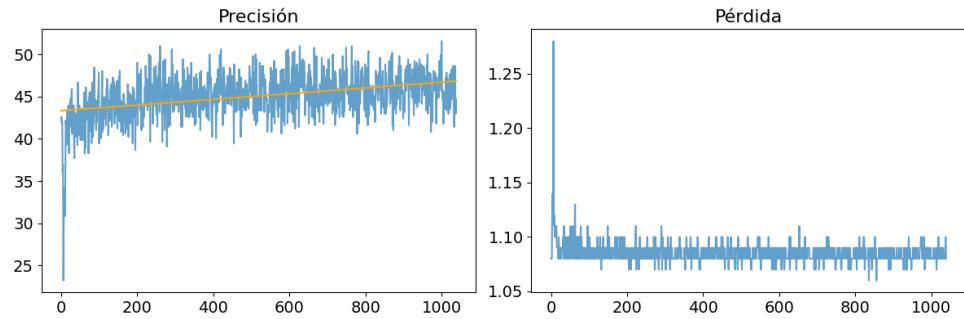
Tras repetir el proceso varias veces con distintos parámetro, tanto con inicialización aleatoria como reutilizando los pesos de BERT **BASE**, se obtuvieron curvas prácticamente idénticas, por lo que se consideró suficiente para pasar a la fase de evaluación. En la [Figura 8.2](#) se recogen las métricas de ambas configuraciones: la **precisión** aumenta de forma consistente, sobre todo en los primeros pasos (posiblemente impulsada por el *warm-up*), mientras que la **pérdida** desciende bruscamente al inicio y después permanece prácticamente plana. Es decir, el modelo acierta cada vez con mayor frecuencia la serie correcta, pero la probabilidad asignada a esa elección varía apenas unas centésimas, lo que sugiere que la función de pérdida se encuentra pronto en una meseta.

La precisión alcanzada supera rápidamente el 45 % incluso con pesos aleatorios, y alcanza el 50 % de manera consistente cuando se parte de pesos heredados, pese a que se ha trabajado con un corpus mucho más pequeño que el habitual en preentrenamientos de lenguaje. Cada experimento requirió entre 7 y 8 días de cómputo en una única GPU, un tiempo bastante superior al esperado.

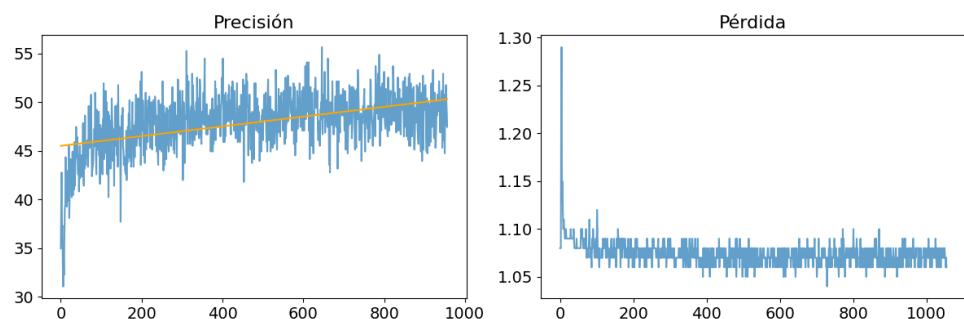
Para entender las razones de estos tiempos tan elevados, además de las limitaciones de *hardware*, se instrumentó el código para medir la duración de cada módulo durante el preentrenamiento. Los resultados se muestran en la [Figura 8.3](#).

El gráfico de la izquierda desglosa el tiempo dedicado únicamente al paso *forward* de HitsBERT. Se observa que HitsBERT consume el 99 % del tiempo, mientras que BERT apenas

8. Evaluación experimental



(a) Preentrenamiento con pesos iniciales aleatorios.



(b) Preentrenamiento con pesos iniciales de BERT BASE.

Figura 8.2.: Evolución de precisión y pérdida durante el preentrenamiento para cada esquema de inicialización de pesos.

requiere un 0.2 % y el resto de operaciones un 0.7 %. Tras realizar una prueba observando los tiempo dentro de HitsBE, se observa que la principal causa es la búsqueda intensiva de palabras en el diccionario pese a haberse optimizado todo lo posible con *PyTorch*. El gráfico de la derecha muestra la distribución completa durante la ejecución con un *mini-lote*. El *forward* del modelo representa el 38.1 %, pero la **actualización** de parámetros (retro propagación + optimizador) domina con un 61.3 %. Se obtiene así que una de las próximas tareas a resolver es la optimización de HitsBE.

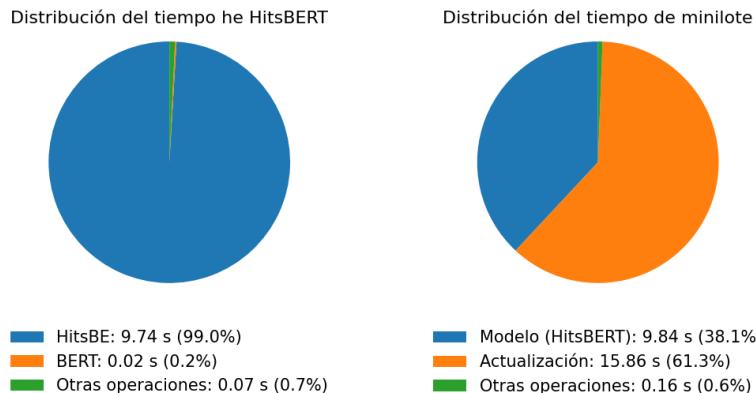


Figura 8.3.: Porcetajes de tiempos de ejecución durante el preentrenamiento. A la izquierda, el porcentaje de cada parte que compone HitsBERT. A la derecha, los porcentajes durante el procesamiento de un mini-lote

Para garantizar reproducibilidad se fijó la semilla global a `42` (`torch.manual_seed`) y se configuró el *DataLoader* con la misma semilla. Por otro lado, todas las dependencias están declaradas en `pyproject.toml` y se gestionan con *Poetry*. Para realizar el preentrenamiento basta desplegar el contenedor definido en el *Dockerfile* y ejecutar el archivo `train.py` de la carpeta `pretraining` con *DeepSpeed* pasando por parámetro el archivo de configuración que se deseé.

8.2. Experimentación

Una vez completado el preentrenamiento, el siguiente paso es comprobar la viabilidad de HitsBERT mediante experimentos de clasificación sencillos. Aunque el preentrenamiento no haya mostrado un gran rendimiento, no sería riguroso extraer conclusiones sin cerrar el ciclo de evaluación. En esencia, utilizar HitsBERT como clasificador implica verlo como un extracto de características. El objetivo, por tanto, es comparar las representaciones que produce con las de otro extracto de referencia sobre distintos conjuntos de datos.

Además, se realizará la evaluación entre los diferentes modelos entrenado también para profundizar en el comportamiento de HitsBE y HitsBERT.

Conjuntos de datos

Para la experimentación se han seleccionado varios conjuntos del *UCR Archive* [CKH⁺15], con la intención de cubrir un abanico razonable de longitudes, número de clase y dominios

8. Evaluación experimental

de origen. Las longitud de las series varían entre 234 y 945 y predominan los problemas binarios, aunque también hay conjuntos con 7, 11 e incluso 60 clases.

Dataset	Train	Test	Longitud	Clases
Car	60	60	577	4
Computers	250	250	720	2
Earthquakes	322	139	512	2
Fish	175	175	463	7
InsectWingbeatSound	220	1980	256	11
RefrigerationDevices	375	375	720	3
ScreenType	375	375	720	3
ShapeletSim	20	180	500	2
ShapesAll	600	600	512	60
Strawberry	613	370	235	2
UWaveGestureLibraryAll	896	3582	945	8
Wine	57	54	234	2

Tabla 8.2.: Resumen de los conjuntos de datos del UCR archive utilizados.

En la [Tabla 8.2](#) se recogen las principales características de los conjuntos de datos utilizados. Cada uno se ha transformado con HitsBERT inicializado con pesos aleatorios y con los pesos de BERT BASE. Para ambos casos se extrajeron el token [CLS] y la media de los vectores de salida. No obstante, se optó por emplear únicamente [CLS], ya que ambas opciones ofrecen resultados prácticamente idénticos y este es el enfoque más habitual.

La evaluación se realizó mediante validación cruzada manual de 5 *folds*. Se generó un archivo con cinco permutaciones fijas y sus particiones para cada conjunto de datos, de modo que todos los algoritmos recibieran exactamente los mismos datos en cada *fold*.

Para valorar HitsBERT como extractor, se le compara con *tsfeatures*, que resume cada serie en 16 características estadísticas: tendencia, número y longitud de periodos, curvatura, etc. Los vectores de *tsfeatures* se someten a la misma validación cruzada que HitsBERT para garantizar una comparación equitativa.

Como clasificador de referencia se emplea un *Random Forest* sencillo. Las métricas seleccionadas son la precisión media por conjunto y la Macro-F1. Para cada clase c , el F1 se define como la media armónica entre precisión (P_c) y exhaustividad (R_c):

$$F1_c = \frac{2P_cR_c}{P_c + R_c},$$

donde $P_c = (\text{TP}_c)/(\text{TP}_c + \text{FP}_c)$ y $R_c = (\text{TP}_c)/(\text{TP}_c + \text{FN}_c)$. En este caso, TP_c son los *verdaderos positivos* (instancias de la clase c clasificadas correctamente), FP_c los *falsos positivos* (instancias de otras clases que se clasifican como c) y FN_c los *falsos negativos* (instancias de la clase c que el modelo no detecta). El Macro-F1 promedia los $F1_c$ de todas las clases otorgando igual peso a cada una, independientemente de su tamaño.

En la [Tabla 8.3](#) se aprecia que, salvo una excepción, el enfoque clásico basado en *tsfeatures* supera a ambas variantes de HitsBERT. Sólo en Earthquakes el resultado es al revés y, en ningún caso, los pesos de BERT BASE ofrecen una ventaja sustancial sobre la inicialización aleatoria. Esta tendencia indica que el preentrenamiento puede que no haya logrado capturar rasgos discriminativos útiles para la clasificación temporal.

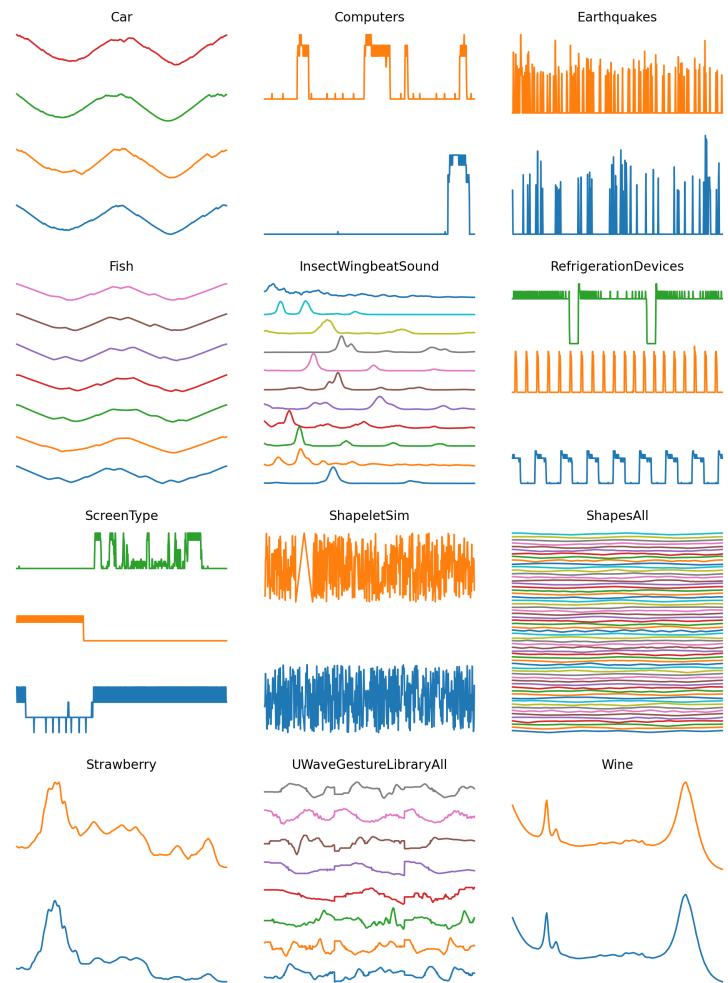


Figura 8.4.: Muestras de cada clase de las series temporales empleadas en la experimentación.

8. Evaluación experimental

Conjunto	Precisión			Macro-F1		
	tsf	BERT	Rand	ts	BERT	Rand
Car	0.800	0.250	0.350	0.784	0.221	0.252
Computers	0.732	0.508	0.496	0.731	0.503	0.493
Earthquakes	0.811	0.817	0.817	0.477	0.492	0.450
Fish	0.663	0.126	0.211	0.660	0.114	0.189
InsectWingbeatSound	0.291	0.073	0.077	0.269	0.064	0.073
RefrigerationDevices	0.731	0.309	0.363	0.727	0.307	0.360
ScreenType	0.600	0.272	0.344	0.599	0.270	0.344
ShapeletSim	0.900	0.200	0.600	0.867	0.180	0.500
ShapesAll	0.735	0.008	0.017	0.705	0.007	0.014
Strawberry	0.920	0.622	0.595	0.913	0.494	0.421
UWaveGestureLibraryAll	0.789	0.113	0.113	0.781	0.105	0.108
Wine	0.844	0.597	0.453	0.840	0.588	0.442

Tabla 8.3.: Precisión y macro-F1 promedio tras la validación cruzada. La columna *tsf* muestra los resultados obtenidos por *tsfeatures*, la columna *BERT* los resultados de HitsBERT con pesos iniciales de BERT BASE, y *Rand* corresponde al mismo modelo con pesos iniciales aleatorios.

Dados estos resultados, el siguiente paso es profundizar en el propio HitsBERT y tratar de obtener algunas conclusiones sobre como este modelo interpreta estos datos.

Conjunto	Precisión				Macro-F1			
	PRand	PBert	NPRand	NPBert	PRand	PBert	NPRand	NPBert
Car	0.350	0.250	0.400	0.567	0.252	0.221	0.361	0.532
Computers	0.496	0.508	0.740	0.728	0.493	0.503	0.739	0.727
Earthquakes	0.817	0.817	0.820	0.817	0.450	0.492	0.451	0.450
Fish	0.211	0.126	0.303	0.349	0.189	0.114	0.257	0.315
InsectWing.	0.077	0.073	0.246	0.236	0.073	0.064	0.230	0.226
Refrigera.	0.363	0.309	0.507	0.531	0.360	0.307	0.495	0.521
ScreenType	0.344	0.272	0.405	0.379	0.344	0.270	0.400	0.377
ShapeletSim	0.600	0.200	0.250	0.400	0.500	0.180	0.220	0.387
ShapesAll	0.017	0.008	0.197	0.298	0.014	0.007	0.155	0.250
Strawberry	0.595	0.622	0.658	0.664	0.421	0.494	0.456	0.537
UWaveGestu.	0.113	0.113	0.518	0.609	0.108	0.105	0.496	0.594
Wine	0.453	0.597	0.471	0.438	0.442	0.588	0.422	0.432

Tabla 8.4.: Accuracy y macro-F1 promedio las para cuatro configuraciones: *PRand* = pesos iniciales aleatorios preentrenados, *PBert* = pesos BERT BASE preentrenados, *NPRand* = pesos aleatorios sin preentrenar, *NPBert* = pesos BERT BASE sin preentrenar.

Por ello se compararon los mismos modelos con preentrenamiento y sin preentrenamiento para tratar de identificar la causa del problema. Los resultados aparecen en la Tabla 8.4. Llama la atención que la configuración con pesos de BERT BASE sin preentrenar obtenga la mayor precisión en 8 de los 12 conjuntos y el mejor macro-F1 en 7 de ellos. Sacar conclusiones solo a partir de la tabla sería un análisis superficial, de modo que se analizarán con más detalle las filas más llamativas.

Al observar Earthquakes destaca la gran brecha entre precisión y macro-F1. Este patrón

suele indicar un fuerte desequilibrio de clases donde el modelo favorece la clase mayoritaria, logrando una exactitud global alta, pero apenas identifica la minoritaria que, en estos casos suele ser la más interesante.

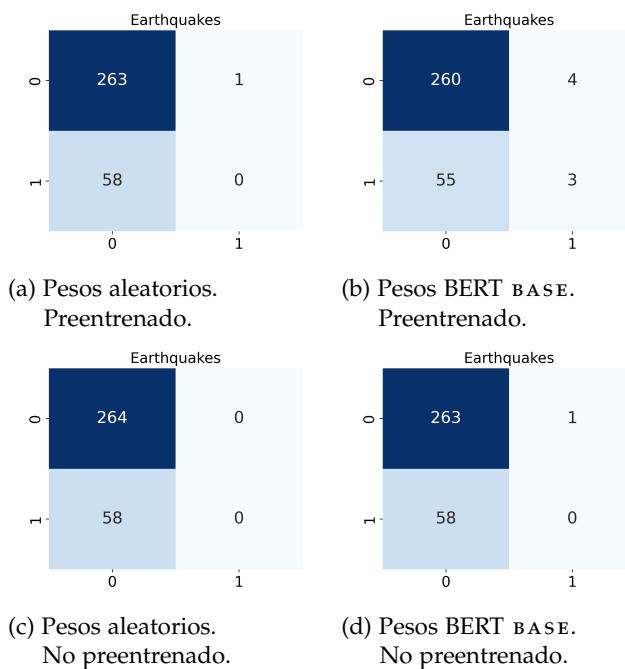


Figura 8.5.: Matrices de confusión de HitsBERT con las 4 configuraciones que se están estudiando. En las columnas se encuentran las etiquetas predichas y en las filas los valores reales.

Para clarificar la situación se calcula la matriz de confusión, que muestra cuántas instancias de cada clase se asignan a cada etiqueta predicha. Por columnas puede verse que clases predice el modelo y por filas se pueden ver las etiquetas reales. En la Figura 8.5 se comprueba lo siguiente:

- Con pesos aleatorios sin preentrenar (**c**) el modelo nunca predice la clase minoritaria. Como puede observarse la segunda columna está formada íntegramente por ceros.
- En (**a**) y (**d**) solo se arriesga una vez con la clase minoritaria y la predicción resulta errónea.
- La única variante que intenta etiquetar la clase minoritaria es la de pesos BERT BASE preentrenados (**b**), aunque su frecuencia sigue siendo muy baja y la precisión escasa.

En consecuencia, la configuración (**b**) muestra una precisión ligeramente menor, pero alcanza el mejor macro-F1, ya que toma mayor riesgo a la hora de determinar la clase.

Otro resultado llamativo de la Tabla 8.4 corresponde al conjunto ShapeletSim. Es el único caso en el que el modelo con pesos *aleatorios y preentrenados* alcanza simultáneamente la mejor precisión y el mejor macro-F1, además con una ventaja considerable sobre el resto. De nuevo,

8. Evaluación experimental

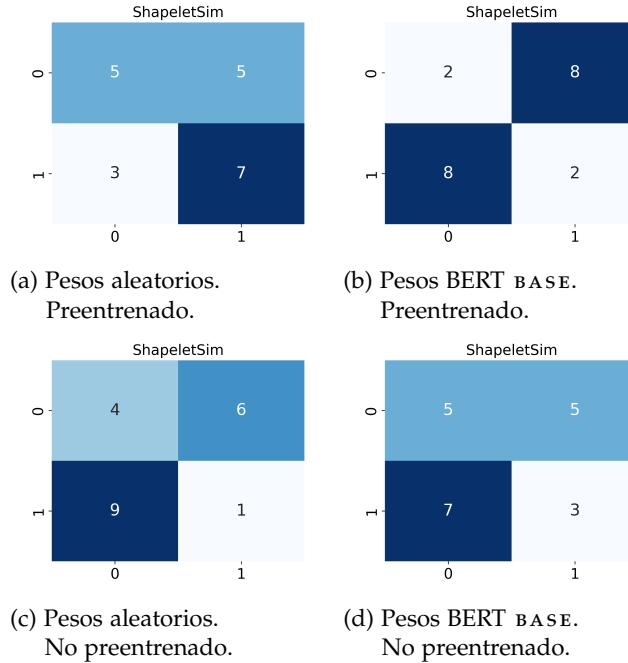


Figura 8.6.: Matrices de confusión de HitsBERT con las 4 configuraciones que se están estudiando. En las columnas se encuentran las etiquetas predichas y en las filas los valores reales.

para obtener una explicación más clara se pueden observar las matrices de confusión que se encuentran en la [Figura 8.6](#).

- El modelo con pesos de BERT BASE preentrenados (configuración **(b)**) presenta un comportamiento anómalo: en la mayoría de los casos asigna la etiqueta contraria a la correcta, lo que se traduce en métricas excesivamente bajas. La modularidad del proceso de extracción y evaluación, común a todos los conjuntos, descarta errores de implementación.
- Las únicas variantes que predicen ambas clases de forma balanceada son las configuraciones opuestas: pesos de BERT BASE sin preentrenar **(d)** y pesos aleatorios con preentrenamiento **(a)**. Esta última ofrece los mejores resultados, corroborando lo observado en la [Tabla 8.4](#).

Otro conjunto de datos que muestra resultados llamativos es UWaveGestureLibraryAll, porque los modelos *sin* preentrenamiento obtienen resultados muy superiores a los modelos preentrenados. Las dos variantes preentrenadas apenas superan el azar (precisión cercana a 0.11, Macro-F1 alrededor de 0.10), mientras que las versiones sin preentrenar mejoran drásticamente. En particular, la configuración con pesos BERT BASE sin preentrenamiento alcanza un 60 % de aciertos y un Macro-F1 de 0.594, superando claramente al modelo con pesos aleatorios.

La [Figura 8.7](#) enriquece este análisis mediante las matrices de confusión:

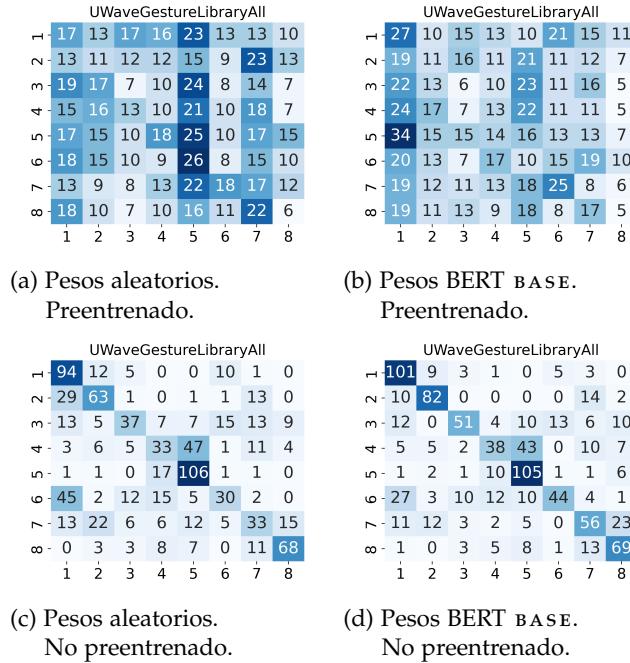


Figura 8.7.: Matrices de confusión de HitsBERT con las 4 configuraciones que se están estudiando. En las columnas se encuentran las etiquetas predichas y en las filas los valores reales.

1. Los modelos *preentrenados* (configuraciones **(a)** y **(b)**) presentan una distribución bastante uniforme tanto en filas como en columnas; es decir, parecen asignar etiquetas de forma casi aleatoria, lo que justifica los Macro-F1 cercanos a 0.10.
2. En los modelos *sin* preentrenar (configuraciones **(c)** y **(d)**) la mayoría de las predicciones se sitúan en la diagonal, lo que indica que aciertan la clase correcta con mucha mayor frecuencia.

El paso de **(a)/(b)** a **(c)/(d)** demuestra que el preentrenamiento no está dotando al modelo de características relevantes. Para estudiarlo con mayor detalle se empleará UMAP, una herramienta que proyecta datos de alta dimensionalidad en dimensiones bajas y facilita el análisis visual.

UMAP (Uniform Manifold Approximation and Projection)² [MHSG18] es una técnica de reducción de dimensionalidad similar a t-SNE, pero útil también para realizar reducciones de dimensionalidad no lineales. Este algoritmo parte sobre unas hipótesis donde destaca que los datos deben estar distribuidos uniformemente sobre una variedad de Riemann en un espacio de alta dimensionalidad. Bajo estas hipótesis consiguen modelar la variedad usando una estructura topológica difusa.

En la Figura 8.8 se muestran las proyecciones obtenidas con UMAP para las salidas de cada uno de los modelos estudiados sobre el conjunto UWAVEGESTURELIBRARYALL. Destaca la gran diferencia ya comentada entre **(a)/(b)** y **(c)/(d)**: cuando se preentrena el modelo, este

²<https://umap-learn.readthedocs.io/en/latest/>

8. Evaluación experimental

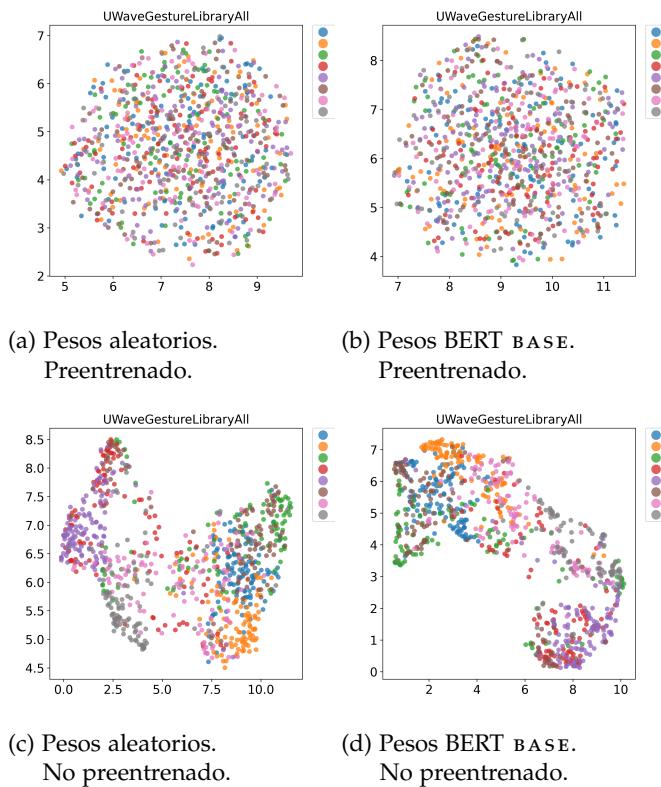


Figura 8.8.: Proyecciones UMAP con las distintas configuraciones del modelo HitsBERT sobre el dataset UWaveGestureLibraryAll.

parece perder la capacidad de extraer características temporales, de modo que los puntos se dispersan y las clases se mezclan.

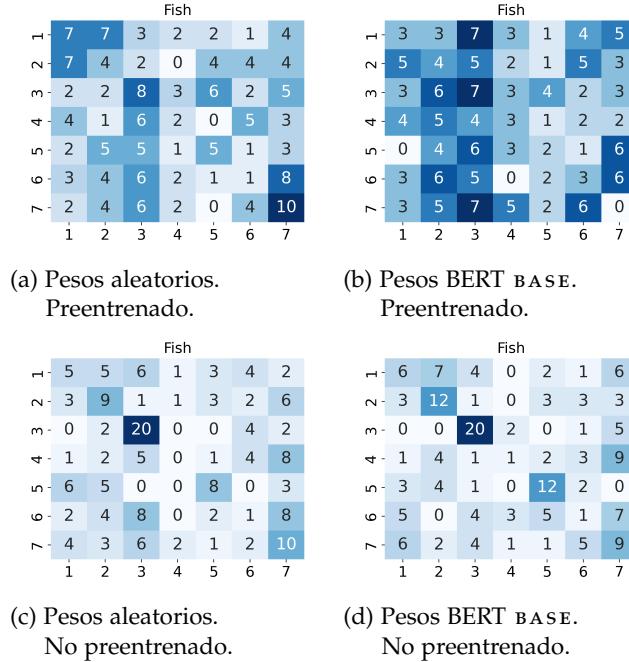


Figura 8.9.: Matrices de confusión de HitsBERT con las 4 configuraciones que se están estudiando. En las columnas se encuentran las etiquetas predichas y en las filas los valores reales.

Para completar el análisis se examina el conjunto Fish, donde todos los modelos obtienen métricas bajas, aunque siguen destacando las variantes sin preentrenar (véase Tabla 8.4). Las matrices de confusión se recogen en la Figura 8.9, y muestran un patrón muy similar al observado en UWaveGestureLibraryAll. Las proyecciones UMAP de la Figura 8.10 refuerzan todavía más esta conclusión.

Las proyecciones preentrenadas (a) y (b) muestran los datos distribuidos en un disco prácticamente homogéneo. Las siete clases se solapan entre ellas, algo coherente con el bajo macro-F1 medido. En contraste, las versiones sin preentrenar (c, d) forman un abanico donde pueden reconocerse ciertas agrupaciones que, a pesar de estar también mezcladas, muestran algo de orden o agrupación local entre ellas. En especial la clase 3 que, en la matriz de confusión muestra la mayor tasa de aciertos con diferencia y en la proyección de UMAP corresponde con la clase verde, la cual se localiza en la parte inferior en ambos casos.

Tras analizar todos estos resultados se pueden extraer algunas conclusiones y proponer nuevas hipótesis sobre cómo continuar este trabajo. La conclusión principal es la limitación del modelo a la hora de clasificar series temporales.

La causa más probable puede estar en el preentrenamiento: difumina las características específicas de cada clase, agrupando todas las series en un mismo vecindario. En cambio, los modelos aún sin preentrenar sí parecen extraer rasgos distintivos y logran cierta separación

8. Evaluación experimental

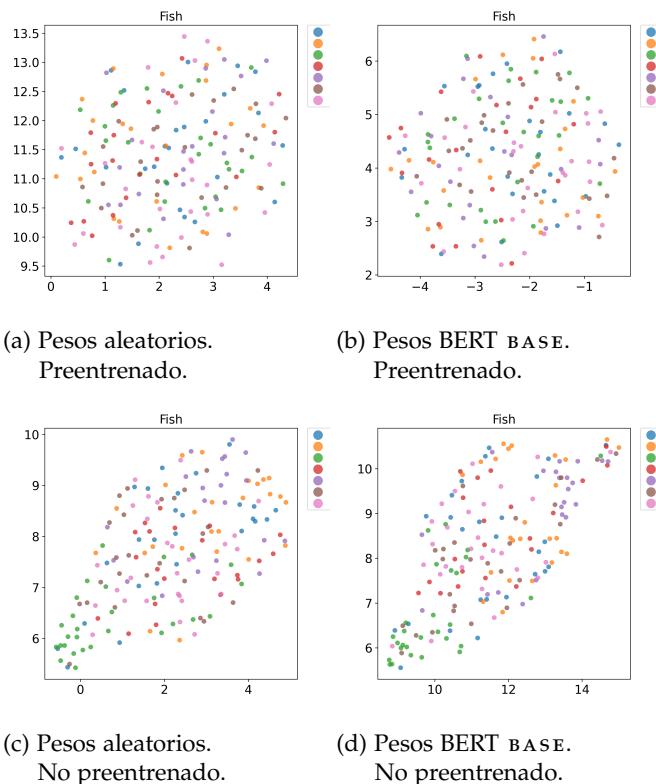


Figura 8.10.: Proyecciones UMAP con las distintas configuraciones del modelo HitsBERT sobre el dataset Fish.

entre clases. En la [Figura 8.11](#) y la [Figura 8.12](#) se muestran las distintas capas de HitsBERT con los pesos de BERT BASE sin preentrenar; se aprecia cómo el modelo va separando los datos progresivamente, lo que indica que la representación generada por HitsBE aporta información representativa incluso antes de aprender a procesar series temporales.

El bajo rendimiento del preentrenamiento puede deberse a distintas causas: tal vez el esquema de enmascaramiento, la generación de opciones o la evaluación de resultados sean inadecuados. Tres opciones pueden ser pocas, o el proceso podría requerir un entorno más controlado y no tan arbitrario. Por otra parte, quizás se necesite mayor aleatoriedad en la selección de los intervalos diádicos y no restringir las frecuencias ni la cantidad de estos.

Una línea de investigación igualmente prometedora pasa por revisar el *corpus* empleado. Es posible que el conjunto de datos no resulte lo bastante representativo ni ofrezca la diversidad necesaria para abarcar la heterogeneidad de las series temporales. Por otro lado podría ocurrir que, al igual que los LLM requieren entrenamientos específicos para cada idioma, de ahí que existan BERT en inglés o en español, tal vez sea conveniente entrenar modelos especializados por dominio. Por ejemplo, uno para series meteorológicas, otro para datos financieros, etc.

En definitiva, los avances de este capítulo abren la puerta a diversas líneas de trabajo sobre el modelo propuesto.

8. Evaluación experimental

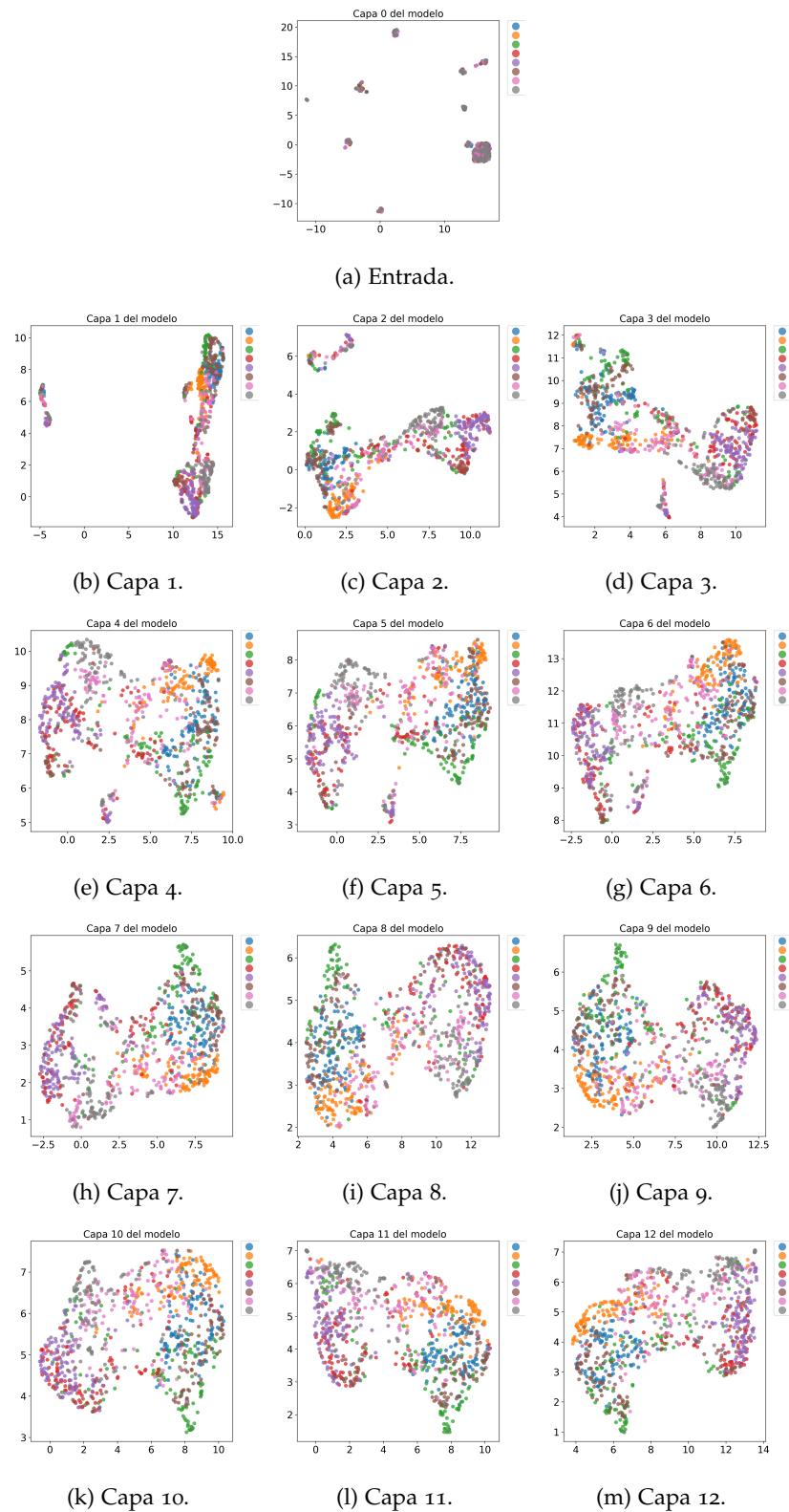


Figura 8.11.: Proyecciones de cada capa del modelo HitsBERT con pesos iniciales de BERT BASE sin preentrenar procesando el conjunto UWaveGestureLibraryAll.

8.2. Experimentación

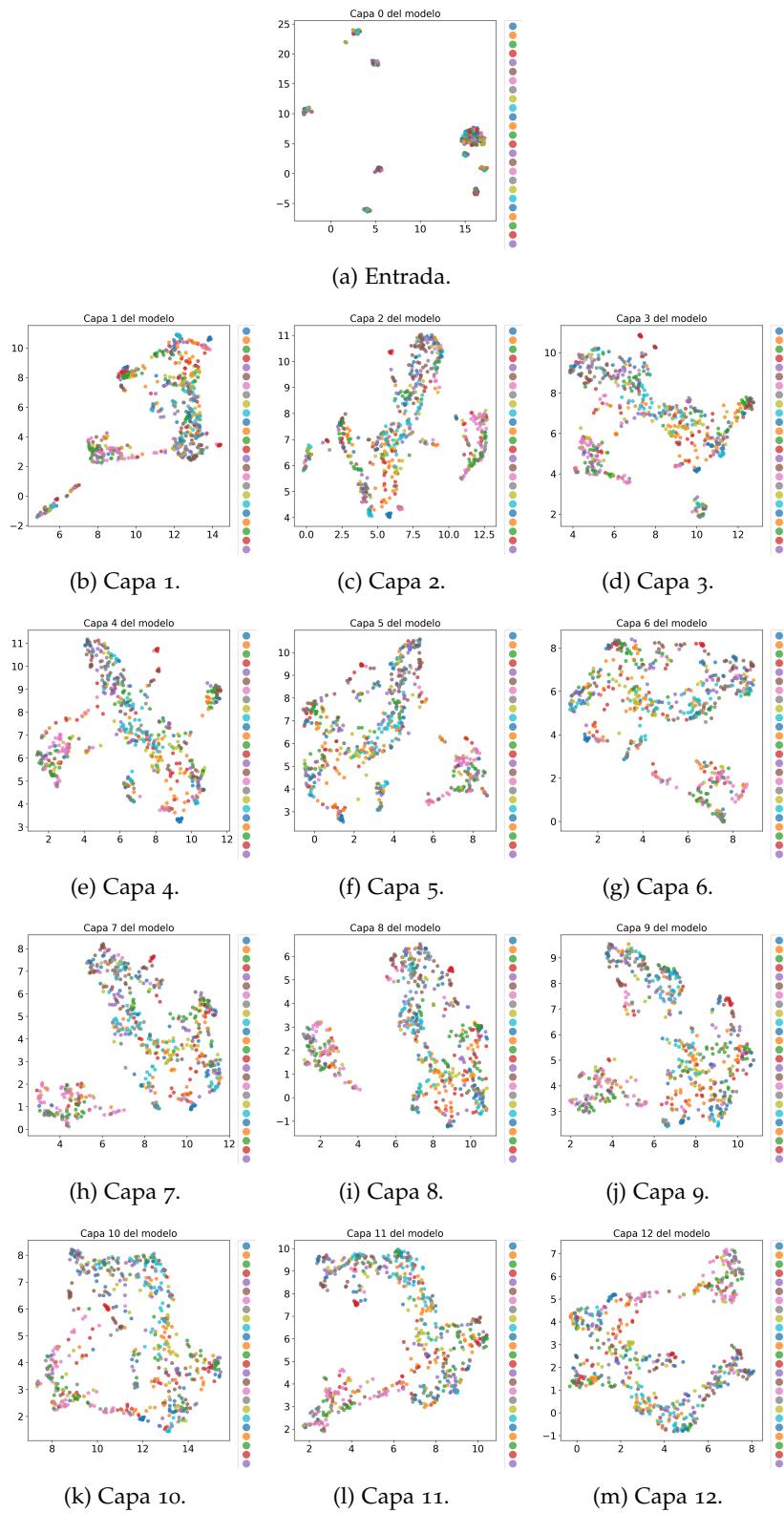


Figura 8.12.: Proyecciones de cada capa del modelo HitsBERT con pesos iniciales de BERT BASE sin preentrenar procesando el conjunto ShapesAll.

9. Conclusiones

Este capítulo constituye el cierre de este trabajo. En él se recogen las conclusiones finales sobre el trabajo realizado, su desarrollo a lo largo del tiempo y las líneas de investigación que quedan abiertas para el futuro.

9.1. Objetivos alcanzados

El proyecto ha cumplido con los objetivos propuestos, abordando de forma conjunta tanto la vertiente matemática como la informática. Desde el punto de vista teórico, se ha logrado profundizar en el campo del Análisis Funcional, comprendiendo y demostrando el teorema de caracterización de las bases *greedy* y mostrando, mediante razonamientos rigurosos, cómo el sistema de Haar cumple con esta propiedad. Este estudio ha permitido consolidar conocimientos adquiridos a lo largo del grado y ha proporcionado una base teórica sólida para fundamentar la propuesta informática.

En el ámbito informático, se ha diseñado y propuesto un esquema estructurado para representar series temporales de manera compatible con arquitecturas *Transformer* integrando el componente matemático de las bases de Haar. Además, dicha arquitectura se ha desarrollado y validado, cumpliendo el objetivo principal de esta parte.

Por otro lado, se ha planteado un preentrenamiento específico para este modelo inspirado en los métodos empleados en el procesamiento del lenguaje con el objetivo de estudiar su viabilidad para el problema de clasificación de series temporales. A pesar de los avances, los resultados obtenidos han puesto de manifiesto las limitaciones de la propuesta, siendo la causa más probable un preentrenamiento poco eficiente. Todo esto ha supuesto un esfuerzo importante de estudio, diseño e implementación evidenciando la complejidad y el desafío técnico que conlleva abordar un proyecto de este tipo.

Por último, este desarrollo ha permitido familiarizarse con tecnologías actuales del desarrollo de proyectos en inteligencia artificial, siguiendo buenas prácticas de reproducibilidad y organización del código. Herramientas como Poetry, PyTorch, DeepSpeed, GitHub o Docker han sido integradas correctamente en el flujo de trabajo.

9.2. Tiempo de desarrollo

El proyecto se ha desarrollado a lo largo de siete meses, como puede verse en el diagrama de Gantt de la Figura 9.1. El trabajo puede agruparse en cuatro grandes bloques: consolidación de conocimientos, desarrollo de la propuesta, experimentación y desarrollo de la memoria. A continuación se presenta una descripción resumida de cada bloque:

- **Consolidación de conocimientos:** este bloque agrupa tanto la parte de comprensión del problema como la formación previa a su resolución. Cabe destacar que la comprensión del problema no es solo entender la tarea de clasificación y las series temporales.

9. Conclusiones



Figura 9.1.: Diagrama de Gantt.

También consiste en comprender los objetivos de las dos partes, tanto la matemática como la informática, y analizar los posibles formas de unión ambas.

- **Desarrollo de la propuesta:** este bloque es el más denso de todos. Comienza con el estudio del estado del arte, una vez que se entiende el problema, cuales son los avances actuales sobre este. A esta etapa le sigue el diseño y desarrollo de la arquitectura propuesta, donde se ha invertido mayor cantidad de tiempo. Para finalizar este bloque se encuentra el diseño y la ejecución de los preentrenamientos. Se incluye en esta sección porque se considera una parte imprescindible para el tipo de arquitectura *Transformer*.
- **Experimentación:** comprende el diseño de los experimentos, la ejecución controlada de los mismos y la evaluación cuantitativa de los resultados. Incluye la preparación de los conjuntos de datos, la extracción de características, la validación cruzada y el análisis comparativo.
- **Desarrollo de la memoria:** Este trabajo se desarrolla junto con el proyecto en sí. Los primeros meses el foco se centraba en estructurar y desarrollar la parte matemática, así como tomar notaciones de lo que se iba avanzando en la parte informática. Los últimos meses cuando la parte matemática estaba casi desarrollada y empiezan a completarse los preentrenamientos, se empleaba la mayor parte del tiempo en desarrollar la parte informática. Aunque la escritura formal comenzó según indica el diagrama, el proceso de comprensión y adquisición de conocimientos matemáticos se ha realizado con mayor antelación de manera gradual para llegar al diseño de la propuesta con los conocimientos matemáticos necesarios.

9.3. Trabajo futuro

Las pruebas realizadas han confirmado que hitsBERT está limitado a la hora de extraer características útiles, debido posiblemente a que el preentrenamiento tiende a colapsar las representaciones de las series. No obstante, si se ha observado que HitsBE puede ser una opción viable para embeber estos datos en los modelos *Transformers*, lo que despeja el camino hacia futuras investigaciones.

Una de estas líneas futuras pasa por comprender mejor el subespacio que genera la **transformada de Haar**. Una variante a estudiar sería, en lugar de realizar una inyección en el espacio

mediante una transformación lineal, realizar la vectorización usando, por ejemplo, una red neuronal. Esto, aparte de eliminar la linealidad, permitiría aumentar las dimensiones de dicho subespacio. Otra opción podría ser conectar con mayor profundidad la propiedad del sistema de Haar de ser una base greedy y realizar un estudio sobre como afectan los distintos coeficientes de la transformada en la propia representación. Dando un paso más, cabe destacar que el esquema jerárquico que se presenta es propio de la transformada *wavelet*, es decir, que se podría buscar otro tipo de función madre que obtuviese mejores representaciones de los datos.

En un nivel paralelo, el **vocabulario** se presenta como una pieza clave. Aumentar el número de "palabras" promete mayor resolución, pero también implica posible mayor ruido. Realizar un estudio sobre la distribución de las distintas clases de monotonía, buscar alguna manera de medir el nivel de representación de estas y ser capaz de perfeccionar el vocabulario sería un punto fundamental para mejorar el rendimiento del propio modelo.

Otro enfoque para este sería considerar vocabularios específicos para cada tarea. Por ejemplo, a partir de un conjunto de datos de series financieras, buscar un vocabulario que minime lo máximo posible el error de representación. Otra alternativa sería un enfoque híbrido. Donde el vocabulario estuviese compuesto de una serie de palabras básicas y otro conjunto más específico para la tarea que se estudia.

El propio **HitsBE** también admite diversos refinamientos en sus parámetros como: variar la longitud admitida de la serie o de la palabra, variando así también el número de segmentos; ajustar el número de niveles de Haar que descomponen a las series o modificar la codificación posiciona.

Sin embargo, está claro que el problema radica en el **preentrenamiento** de HitsBERT. Esta fase es especialmente delicada, ya que requiere encontrar un algoritmo que permita que el modelo aprenda a distinguir entre diversos tipos de series temporales de forma eficaz. El enfoque basado en *Masked Language Modeling* resulta razonable, dado que es una estrategia ampliamente validada en el campo del procesamiento del lenguaje natural. No obstante, es posible que la adaptación realizada no haya sido la más adecuada o que, simplemente, este tipo de técnica no sea idónea para las series temporales debido a su alto componente estocástico y su diversa naturaleza. En cualquier caso, este punto abre la puerta a una gran cantidad de posibilidades que merecen ser exploradas en profundidad.

Por último, conviene abordar la inevitable diversidad de las series temporales. Cualquier dato que pueda ser medido a lo largo del tiempo puede considerarse como tal y sus características varían enormemente según su origen: no es lo mismo una serie financiera que una meteorológica, sanitaria o procedente de sensores industriales. De forma análoga a como los modelos lingüísticos se entrena específicamente para cada idioma, podría ser mejor opción diseñar variantes de HitsBERT centradas en dominios concretos, aplicando *fine-tuning* para transferir el conocimiento adquirido de forma general y refinarlo posteriormente con datos especializados y representativos de cada ámbito.

En definitiva, el camino abierto por este trabajo ofrece múltiples líneas de investigación para seguir avanzando en la construcción de modelos específicos para series temporales. Se ha demostrado que el diseño de HistBE y los fundamentos que lo sustentan son sólidos y ofrecen margen para futuras mejoras. Profundizar en la comprensión del subespacio generado por Haar, refinar el diseño del vocabulario, ajustar los parámetros del modelo y replantear el esquema de preentrenamiento son algunas de las muchas opciones a explorar. Todo ello con

9. Conclusiones

el objetivo de consolidar, en un futuro, un modelo fundacional sólido y especializado para abordar los retos que presentan las series temporales.

Bibliografía

- [Alba] Rafael Payá Albert. Notas de análisis funcional. Universidad de Granada, Granada.
- [Albb] Rafael Payá Albert. Notas de análisis matemático ii. Material proporcionado en clase. Universidad de Granada, Granada.
- [AM09] Yaser S. Abu-Mostafa. Lecture 9: The bias-variance tradeoff [slides], 2009. Learning From Data (CS156), California Institute of Technology.
- [AMMIL12] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, y Hsuan-Tien Lin. *Learning from Data: A Short Course*. AMLBook, 2012.
- [BFL⁺20] A. Bagnall, M. Flynn, J. Large, J. Lines, y M. Middlehurst. On the usage and performance of the hierarchical vote collective of transformation-based ensembles version 1.0 (hive-cote v1.0). En V. Lemaire, S. Malinowski, A. Bagnall, T. Guyet, R. Tavenard, y G. Ifrim, editores, *Advanced Analytics and Learning on Temporal Data. AALTD 2020*, volumen 12588 de *Lecture Notes in Computer Science*, páginas 1–22. Springer, Cham, 2020. Published 16 December 2020.
- [BLB⁺17] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, y Eamonn Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606—660, May 2017.
- [BLHB15] Anthony Bagnall, Jason Lines, Jon Hills, y Aaron Bostrom. Time-series classification with cote: The collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535, 2015.
- [BPC20] Iz Beltagy, Matthew E. Peters, y Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.
- [CBNKL18] Maximilian Christ, Nils Braun, Julius Neuffer, y Andreas W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, 307:72–77, 2018.
- [Cho18] François Chollet. *Deep Learning with Python*. Manning Publications, 2018.
- [CKH⁺15] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, y Gustavo Batista. The ucr time series classification archive, July 2015. www.cs.ucl.ac.uk/~eamonn/time_series_data/.
- [CNQK21] Nestor Cabello, Elham Naghizade, Jianzhong Qi, y Lars Kulik. Fast, accurate and interpretable time series classification through randomization, 2021. Submitted on 31 May 2021.
- [Dan21] Gustavo Dantas. Perceptrón 5 unidades [imagen]. Wikimedia Commons, 2021. Licencia CC BY-SA 4.0.
- [DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, y Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. En Jill Burstein, Christy Doran, y Thamar Solorio, editores, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, páginas 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [DRTV13] Houtao Deng, George Runger, Eugene Tuv, y Martyanov Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 2013.

Bibliografía

- [DSW20] Angus Dempster, Daniel F. Schmidt, y Geoffrey I. Webb. Rocket: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34:1454–1495, 2020.
- [DSW21] Angus Dempster, Daniel F. Schmidt, y Geoffrey I. Webb. Minirocket: A very fast (almost) deterministic transform for time series classification. En *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD ’21, página 248–257, New York, NY, USA, 2021. Association for Computing Machinery.
- [DSW23] A. Dempster, D.F. Schmidt, y G.I. Webb. Hydra: competing convolutional kernels for fast and accurate time series classification. *Data Mining and Knowledge Discovery*, 37(5):1779–1805, 2023. Published online 16 May 2023; Issue date: September 2023.
- [DSW24] A. Dempster, D.F. Schmidt, y G.I. Webb. Quant: a minimalist interval method for time series classification. *Data Mining and Knowledge Discovery*, 38(4):2377–2402, 2024. Published 22 May 2024; Issue date: July 2024.
- [fde17] fdeloche. Recurrent neural network unfold [imagen]. Wikimedia Commons, 2017. Licencia CC BY-SA 4.0.
- [GVE22] Antoine Guillaume, Christel Vrain, y Wael Elloumi. Random dilated shapelet transform: A new approach for time series shapelets. En Mounîm El Yacoubi, Eric Granger, Pong Chi Yuen, Umapada Pal, y Nicole Vincent, editores, *Pattern Recognition and Artificial Intelligence*, páginas 653–664, Cham, 2022. Springer International Publishing.
- [Hei04] Christopher Heil. *A Basis Theory Primer*. Birkhäuser, 2004.
- [HLB⁺14] J. Hills, J. Lines, E. Baranauskas, et al. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014. Published online 18 May 2013; Issue date: July 2014.
- [IBL21] Peter Izsak, Moshe Berchansky, y Omer Levy. How to train BERT with an academic budget. En Marie-Francine Moens, Xuanjing Huang, Lucia Specia, y Scott Wen-tau Yih, editores, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, páginas 10644–10652, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.
- [IFDB⁺23] Ali Ismail-Fawaz, Maxime Devanne, Stefano Berretti, Jonathan Weber, y Germain Forestier. Lite: Light inception with boosting techniques for time series classification. En *2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA)*, páginas 1–10, 2023.
- [IFDWF22] Ali Ismail-Fawaz, Maxime Devanne, Jonathan Weber, y Germain Forestier. Deep learning for time series classification using new hand-crafted convolution filters. En *2022 IEEE International Conference on Big Data (Big Data)*, páginas 972–981, 2022.
- [IFLF⁺20] H. Ismail Fawaz, B. Lucas, G. Forestier, et al. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020. Published 07 September 2020; Issue date: November 2020.
- [KB14] Diederik P. Kingma y Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [KSH12] Alex Krizhevsky, Ilya Sutskever, y Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. En F. Pereira, C.J. Burges, L. Bottou, y K.Q. Weinberger, editores, *Advances in Neural Information Processing Systems*, volumen 25. Curran Associates, Inc., 2012.
- [LNGI⁺19] T. Le Nguyen, S. Gsponer, I. Ilie, et al. Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. *Data Mining and Knowledge Discovery*, 33(5):1183–1222, 2019. Published 21 May 2019; Issue date: 01 July 2019.

- [LSP⁺19] B. Lucas, A. Shifaz, C. Pelletier, et al. Proximity forest: an effective and scalable distance-based classifier for time series. *Data Mining and Knowledge Discovery*, 33(3):607–635, 2019. Published 06 February 2019; Issue date: 15 May 2019.
- [MB22] Matthew Middlehurst y Anthony Bagnall. The freshprince: A simple transformation based pipeline time series classifier. En Mounim El Yacoubi, Eric Granger, Pong Chi Yuen, Umapada Pal, y Nicole Vincent, editores, *Pattern Recognition and Artificial Intelligence*, páginas 150–161, Cham, 2022. Springer International Publishing.
- [MHSG18] Leland McInnes, John Healy, Nathaniel Saul, y Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.
- [MLB20] Matthew Middlehurst, James Large, y Anthony Bagnall. The canonical interval forest (cif) classifier for time series classification. En *2020 IEEE International Conference on Big Data (Big Data)*, páginas 188–195, 2020.
- [MLF⁺21] M. Middlehurst, J. Large, M. Flynn, et al. Hive-cote 2.0: a new meta ensemble for time series classification. *Machine Learning*, 110(12):3211–3243, 2021. Published 24 September 2021; Issue date: December 2021.
- [MSB24] Matthew Middlehurst, Patrick Schäfer, y Anthony Bagnall. Bake off redux: a review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*, 38(4):1958—2031, April 2024.
- [MVB19] M. Middlehurst, W. Vickers, y A. Bagnall. Scalable dictionary classifiers for time series classification. En H. Yin, D. Camacho, P. Tino, A. Tallón-Ballesteros, R. Menezes, y R. Allmendinger, editores, *Intelligent Data Engineering and Automated Learning – IDEAL 2019*, volumen 11871 de *Lecture Notes in Computer Science*, páginas 348–360. Springer, Cham, 2019. Published 18 October 2019.
- [NI23] T.L. Nguyen y G. Ifrim. Fast time series classification with random symbolic subsequences. En T. Guyet, G. Ifrim, S. Malinowski, A. Bagnall, P. Shafer, y V. Lemaire, editores, *Advanced Analytics and Learning on Temporal Data. AALTD 2022*, volumen 13812 de *Lecture Notes in Computer Science*, páginas 59–72. Springer, Cham, 2023. Published 04 February 2023.
- [OL19] George Oastler y Jason Lines. A significantly faster elastic-ensemble for time-series classification. En *Intelligent Data Engineering and Automated Learning – IDEAL 2019: 20th International Conference, Manchester, UK, November 14–16, 2019, Proceedings, Part I*, página 446–453, Berlin, Heidelberg, 2019. Springer-Verlag.
- [RNSS18] Alec Radford, Karthik Narasimhan, Tim Salimans, y Ilya Sutskever. Improving language understanding by generative pre-training, 2018. Preprint. Work in progress.
- [Sal07] David Salomon. *Data Compression: The Complete Reference*. Springer-Verlag London Limited, London, 4 edición, 2007. Printed on acid-free paper.
- [Sana] Pablo Mesejo Santiago. Notas de la asignatura visión por computador. Material proporcionado en clase. Universidad de Granada, Granada.
- [Sanb] Pablo Mesejo Santiago. Notas del temario práctico de la asignatura visión por computador. Material proporcionado en clase. Universidad de Granada, Granada.
- [Sch15] P. Schäfer. The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6):1505–1530, 2015. Published 16 September 2014; Issue date: November 2015.
- [SL17] Patrick Schäfer y Ulf Leser. Fast and accurate time series classification with weasel. En *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17*, página 637–646, New York, NY, USA, 2017. Association for Computing Machinery.
- [SL23] P. Schäfer y U. Leser. Weasel 2.0: a random dilated dictionary transform for fast, accurate and memory constrained time series classification. *Machine Learning*, 112(12):4763–4788, 2023. Published 19 September 2023; Issue date: December 2023.

Bibliografía

- [TDB⁺22] C.W. Tan, A. Dempster, C. Bergmeir, et al. Multirocket: multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery*, 36(5):1623–1646, 2022. Published online 29 June 2022; Issue date: September 2022.
- [TLI⁺23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, y Guillaume Lample. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [VSP⁺17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, y Illia Polosukhin. Attention is all you need. En I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, y R. Garnett, editores, *Advances in Neural Information Processing Systems*, volumen 30. Curran Associates, Inc., 2017.
- [Wal02] David F. Walnut. *An Introduction to Wavelet Analysis*. Applied and Numerical Harmonic Analysis. Birkhäuser Boston, 2002. 1st printing, 2001. Corrected 2nd printing, 2003.
- [WLK⁺20] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, y Hao Ma. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768, 2020.
- [WYO17] Zhiguang Wang, Weizhong Yan, y Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. En *2017 International Joint Conference on Neural Networks (IJCNN)*, páginas 1578–1585, 2017.
- [WZZ⁺23] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, y Liang Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2304.00563*, April 2023.